Tampereen yliopisto

Valtteri Lahti

YRITYSOHJELMISTON INTEGRAATIO

Kandidaatintyö
Informaatioteknologian ja viestinnän tiedekunta
Elokuu 2020

TIIVISTELMÄ

Valtteri Lahti: Yritysohjelmiston integraatio
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan TkK-tutkinto-ohjelma, Tietotekniikka
Elokuu 2020

Ohjelmiston integraatio on yksi tärkeimmistä vaiheista ohjelmistoa kehittäessä. Viimeisten vuosikymmenten aikana integraatioon on luotu useita erilaisia tekniikoita, joista jokainen soveltuu erilaisiin käyttötapauksiin ja ohjelmistoihin. Yrityksien on tulevaisuuden kannalta elintärkeää valita oikea integraatioteknologia oikeaan tarkoitukseen, jotta yritykset säästyvät myöhemmiltä vaivoilta ja kuluilta. Integraatioteknologia ohjelmiston sisäisesti vaikuttaa suoraan ohjelmiston suorituskykyyn, muokattavuuteen, ylläpidettävyyteen ja joissain tapauksissa koko ohjelmiston toimivuuteen. Ulkoiseen integraatioon valittu teknologia vaikuttaa myös suorituskyvyn lisäksi siihen, kuinka helposti ja pienillä muutoksilla toisen yrityksen ohjelmisto pystytään kytkemään ohjelmistoon.

Työ on kirjallisuustutkielma, jossa ensin esitellään tunnettuja integraatioteknologioita ja -menetelmiä. Tekniikoiden esittelyn jälkeen tarkastellaan niiden ominaisuuksia, minkä jälkeen niille päätellään ominaisuuksiensa perusteella mahdollisia käyttökohteita.

Objektiivisesti parasta integraatioteknologiaa jokaiseen ohjelmistoon ei ole olemassa. Käytettävä teknologia on riippuvainen ohjelmiston arkkitehtuurista, mahdollisista integroitavista järjestelmistä ja tulevaisuuden suunnitelmista. Jos ohjelmisto on tällä hetkellä pieni ja muita integroitavia osia on vain vähän, ei välttämättä ole kannattavaa valita teknologiaa, jonka toimintakyky alenee huomattavasti integraatiovaatimusten lisääntyessä.

Tutkielman avulla voidaan päätellä, että viestintään tarkoitetut väliohjelmistot ovat helpoin ja toimiva ratkaisu hajautetun järjestelmän sisäiseen integraatioon. Viestintään tarkoitetulla väliohjelmistolla integroitavan ohjelmiston toimivuus on parhain mahdollinen ja luotettavuutta voidaan nostaa suorituskyvyn hinnalla. Objektipohjaiset väliohjelmistot ovat toimivia ratkaisuja silloin, kun ohjelmisto on pieni ja kommunikoinnin suorituskyky ja varmuus ei ole prioriteetti. Objektipohjaisten väliohjelmistojen etuna on niiden yksinkertaisuus ja helppokäyttöisyys. Jos tavoitteena on julkistaa ohjelmiston rajapinta kaikille saatavaksi, on suositulla REST-arkkitehtuurilla toteutettu verkkopalvelu hyvä vaihtoehto. REST-arkkitehtuurilla toteutettu verkkopalvelu mahdollistaa helpon, skaalautuvan ja nopean kommunikoinnin.

Avainsanat: Ohjelmiston integraatio, integraatio, väliohjelmisto, verkkopalvelu

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

CORBA	engl. Common object request broker architecture, objektipohjaisten väliohjelmistojen standardi
ESB	engl. Enterprise service bus, liikepalveluväylä
HTTP	engl. Hypertext transfer protocol, verkossa käytettävä tiedonsiirto-protokolla
IDL	engl. Interface description language, rajapintojen määrittelykieli
iPaaS	engl. Integration platform as a service, integraatioalusta palveluna
JSON	engl. JavaScript object notation, avoimen standardin tiedostomuoto
MOM	engl. Message-oriented middleware, viestintään tarkoitettu väliohjelmisto
ORB	engl. Object request broker, palvelupyynnön välittäjä
REST	engl. Representational state transfer, arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen
RPC	engl. Remote procedure call, etäproseduurikutsu
SOA	engl. Service oriented architecture, palvelukeskeinen arkkitehtuuri
SOAP	sovellusohjelmien välinen viestipohjainen tietoliikenneprotokolla
WS	engl. Web services, verkkopalvelut
XML	engl. Extensible markup language, merkintäkielien standardi, joka määrittää tietojen merkintämuodon

.

1. JOHDANTO

Yksi suurimmista modernien yritysohjelmistojen haasteista on integraatio. Sen lisäksi, että ohjelmiston tarvitsee sisäisesti integroida eri moduuleja ja järjestelmiä, on yritysten usein myös tarpeellista integroida ohjelmistonsa toisen yrityksen ohjelmiston kanssa. Integraatiota varten on luotu useita erilaisia ja eri käyttöön soveltuvia teknologioita ja standardeja. Eniten arvoa yritykselle tuottaa integraatioteknologia, joka vastaa vaadittuihin vaatimuksiin ja on helppo ja nopea ottaa käyttöön ilman suuria koodimuutoksia. Vaikka vanhimmat tekniikat ovat olleet olemassa vuosikymmeniä, uusia standardeja ja teknologioita luodaan edelleen parempien tulosten toivossa.

Jatkuvan teknologioiden kehityksen vuoksi tekniikoita ohjelmistojen integraatioon on erityyppisille ohjelmistoille suuri määrä. Tekniikoiden suuren määrän ja sovellusalueiden vuoksi ohjelmistoyritysten tulee suunnitella tarkasti ohjelmistojensa ja tuotteidensa integraatioarkkitehtuuri, jotta ne myöhemmin välttäisivät kalliit ja riskialttiit rakenteelliset muutokset. Ohjelmistojen suorituskykyä yleensä tarkastellaan maksimikuorman, vasteajan, skaalautuvuuden sekä suoritustehon suhteen. Koska ohjelmisto vaatii toimiakseen usein integraatiota, valitulla integraatiotekniikalla on suuri vaikutus suorituskyvyn ominaisuuksiin.

Tässä tutkielmassa käsitellään tunnettuja ohjelmistojen integraatioteknologioita ja -menetelmiä. Tutkielman tavoitteena on selvittää tunnettuja eri teknologioita integraatioon. Samalla selvitetään eri integraatioteknologioiden ominaisuuksia. Toisena tavoitteena tutkielmassa on selvittää, onko olemassa yleisesti oikeana pidettyä integraatioteknologiaa.

Tutkielma on luonteeltaan kirjallisuuskatsaus. Tutkielmassa tutustutaan luvussa 2 yritysohjelmiston käsitteeseen ja sen ominaisuuksiin, minkä jälkeen luvussa 3 esitellään integraation tarve ja tunnettuja teknologioita ja menetelmiä. Luvussa 4 esitellään eri teknologioiden ominaisuuksia. Luvussa 5 on päätelmiä eri integraatioteknologien mahdollisista käyttökohteista. Luvussa 6 on lopuksi tutkielman yhteenveto.

2. YRITYSOHJELMISTO

Yritysohjelmisto on tietyn organisaation tarpeisiin räätälöity tietokoneohjelmisto, jota käytetään organisaation sisäisesti. Yritysohjelmistoon kuuluu esimerkiksi organisaation sisäinen kommunikaatio, toiminnanohjausjärjestelmä, toimitusketjun hallinta ja asiakkuudenhallinta. Yritysohjelmiston tarkoitus on edesauttaa yrityksen tuottavuutta ja tehokkuutta. [1,2] Sisäisen käytön lisäksi, yritysohjelmistoja usein suunnitellaan integroitavaksi muiden ohjelmistojen kanssa [3]. Fowlerin mukaan yritysohjelmistojen keskiössä ovat suuren datamäärän hallinnointi, datan turvallinen rinnakkainen käsittely ja datan käsittelyyn vaadittavat käyttöliittymät. Ohjelmistojen suorituskykyä yleensä tarkastellaan maksimikuorman, vasteajan, skaalautuvuuden, suoritustehon suhteen. [4, Introduction]

Yritysohjelmisto koostuu useista eri hallintajärjestelmistä, jotka integroidaan ohjelmiston kanssa. Useimmat yritysohjelmistot toimivat yhteistyössä muiden yritysohjelmistojen kanssa, jolloin sisäisten järjestelmien lisäksi yritysohjelmistot integroidaan myös toisiin ohjelmistoihin. Yritysohjelmiston arkkitehtonisessa suunnittelussa tulee yleisesti integraation lisäksi useita muita ongelmia, rajoitteita ja vaatimuksia. Tyypillisimpien ongelmien ratkaisuiksi on kehitetty arkkitehtonisia malleja. Malleja käytetään tiettyjen ongelmien malliratkaisuina. Yhtä mallia käytetään yhden ongelman ratkaisuun ja useaa mallia voidaan käyttää yhdessä ohjelmistossa eri tarkoituksiin. [5, s. 5–6] Malli ei yleensä ikinä ole suoraan sopiva ohjelmistoon, vaan sitä pitää muokata ohjelmistoon ja ongelmaan sopivaksi [4, Introduction, kappale ”Once you need the pattern”].

3. INTEGRAATIO

Yritysohjelmistossa integraatio on tarpeellista yritysohjelmiston eri järjestelmien ja komponenttien välillä sekä usein myös toisen yritysohjelmiston kanssa. Ohjelmiston sisäistä integraatiota tarvitaan esimerkiksi, jos järjestelmä halutaan toteuttaa hajautettuna tai jos vanhaa järjestelmää ei ole järkevää päivittää, mutta sitä vaaditaan uudessa ohjelmistossa. Integraatio voidaan toteuttaa muun muassa käyttämällä yhteistä dataa, esimerkiksi tietokantaa, komponenttien välillä kommunikoivalla ohjelmistolla tai suoralla funktiokutsulla, jos komponentit ovat tietoisia toisistaan [6, s. 4]. Integraation toteuttamiseksi on saatavilla useita eri teknologioita ja arkkitehtuurisia malleja. Suurin osa tunnetuista teknologioista ovat väliohjelmistoja, minkä vuoksi suurin huomio on niissä.

Yritysohjelmiston järjestelmien ja komponenttien välinen integraatio mahdollistaa ohjelmiston kehityksen modulaarisesti, eli ohjelmiston isoja kokonaisuuksia voidaan kehittää pienempinä osina. Modulaarisuus johtaa ohjelmiston nopeampaan kehitykseen. Nopeasta kehityksestä seuraa pienemmät kulut. Lisäksi modulaarisuus mahdollistaa ohjelmiston helpomman ylläpidon ja toiminnallisuuden lisäämisen.

3.1 Väliohjelmisto

Väliohjelmisto on ohjelmisto, jolla voidaan liittää hajautettujen järjestelmien komponentteja toisiinsa. Hajautetuilla järjestelmillä tarkoitetaan järjestelmiä, joiden komponentit on hajautettu useille eri laitteille, jotka kommunikoivat keskenään intranetin tai internetin välityksellä [7, luku 1.1]. Väliohjelmiston yhteydet voivat olla yhdestä tai useammasta komponentista yhdelle tai useammalle komponentille [6, s. 40]. Eri ongelmia ratkaisevia ja eri käyttöön soveltuvia väliohjelmistoja on kehitetty useita.

Objektipohjaiset väliohjelmistot (*object-oriented middleware*) tulivat käyttöön 1990-luvun alussa. Nämä väliohjelmistot mahdollistavat hajautettujen järjestelmien komponenttien välisen kommunikoinnin ikään kuin komponentit olisivat samalla laitteella. Tällä tavoin kommunikointi on komponenttien välillä yleisimmin synkronista. Joissakin tapauksissa komponenttien välisten rajapintojen määrittelyyn käytetään rajapintojen määrittelykieltä (*interface description language, IDL*), joka määrittää komponentin saatavilla olevat metodit, niiden parametrit sekä paluuarvon tyypit. [6, s. 41–42]

Viestintään tarkoitettu väliohjelmisto (*message-oriented middleware, MOM*) on väliohjelmisto, jonka tehtävä on välittää viesti yhdestä tai useammasta komponentista yhdelle tai

useammalle vastaanottajalle. Suurta yritysohjelmistoa kehitettäessä MOM on yksi tärkeimmistä teknologioista. MOMilla voidaan integroida sovelluksia yhteen yhdeksi järjestelmäksi riippumatta niiden alustoista, teknologioista tai kielistä. Tämän vuoksi MOMilla on mahdollista integroida vanhempiakin ohjelmistoja uusien kanssa. MOM kytkee komponentit löyhästi toisiinsa, mistä seuraa, että koko ohjelmisto ei kaadu, jos jokin komponentti ei hetkellisesti ole saatavilla. Kyseisen ominaisuuden väliohjelmisto saa aikaan käyttämällä viestijonoa, johon lähetettävät viestit tallennetaan odottamaan lukua. [6, s. 43–44] Viestin lähettävä komponentti määrittää väliohjelmiston jonon, johon viesti halutaan lähettää. Viestin vastaanottava komponentti ilmoittaa väliohjelmistolle saaneensa viestin, jolloin viesti poistuu jonosta. Edellä mainitun varmistuksen vuoksi viestintään tarkoitetun väliohjelmiston viestien välitys on luotettavaa sekä asynkronista. [8, s. 43] Väliohjelmiston asynkronisuuden vuoksi viestin lähettävä komponentti voi jatkaa toimintaansa odottamatta viestin kuittaamista tai siihen vastaamista vastaanottajan toimesta [6, s. 45].

Useimmissa tapauksissa lähtökohtainen MOM ei ole riittävä ohjelmistojen tarpeisiin. Tärkeät tehtävät vaativat MOMilta enemmän tehokkuutta ja varmuutta viestin välittämiseen. Luotettavuutta, skaalautuvuutta ja käytettävyyttä parantaessa MOMin tehokkus luonnollisesti kärsii. Tavallisimmin MOM -palvelimet tarjoavat laadultaan eri tasoisia palveluita, joista on mahdollisuus valita tehokkuudeltaan ja luotettavuudeltaan sopiva. MOMin tapauksessa luotettavuudella tarkoitetaan ohjelmiston kykyä olla kadottamatta lähetettyä viestiä. MOM-teknologian luotettavuutta voidaan parantaa myös klusteroimalla MOM-palvelimia eli ajetaan useita instansseja eri palvelimilla, jolloin yhden instanssin kaatuminen ei haittaa ohjelman suoritusta [6, s. 45–47].

3.1.1 Etäproseduurikutsu

Etäproseduurikutsu (*remote procedure call, RPC*) on Linthicumin mukaan vanhin, helpoimmin ymmärrettävä ja helpoimmin käytettävä väliohjelmisto. Etäproseduurikutsun avulla voidaan kutsua tiettyä toiminnallisuutta yhden prosessin sisällä, jonka seurauksena toiminnallisuuden oikea suoritus tapahtuu täysin erillisessä prosessissa ja laitteessa. [16, luku 7]

RPC on synkroninen väliohjelmisto eli kutsuvan prosessin suoritus keskeytyy etäproseduurikutsun ja kutsuttavan toiminnallisuuden suorituksen ajaksi [16, luku 7]. Etäproseduurikutsussa asiakasohjelma antaa paikalliselle ”kannalle” (*stub*) kutsun parametrit. Kannan tehtävänä on välittää kutsu ja sen parametrit palvelimelle ja palvelimen vastaus takaisin asiakasohjelmalle. Koska etäproseduurikutsu on lähes samanlainen kuin tavan-

omainen funktiokutsu, on sen avulla hajautettujen järjestelmien ohjelmointi vaivattomampaa muihin väliohjelmistoihin verrattuna. Etäproseduurikutsussa ohjelmistokehittäjän ei tarvitse huolehtia komponenttien yhteyksien rajapinnoista. [21, s. 114–115]

3.1.2 CORBA

CORBA (*Common Object Request Broker Architecture*) on useimmiten esillä, kun on kyse integraatiosta tai väliohjelmistoista. CORBA on objektipohjaisten väliohjelmistojen standardi. CORBAn ensimmäinen versio (CORBA 1.0) julkaistiin vuoden 1991 lokakuussa [9]. CORBA on tunnettu, koska sillä oli menneisyydessä vahva asema ja on edelleen käytössä vanhoissa järjestelmissä.

CORBA-arkkitehtuurissa käytetään komponenttien väliseen kommunikointiin palvelupyynnön välittäjää (*Object Request Broker, ORB*) [6, s. 41]. ORB on vastuussa asiakasohjelman kutsun välittämisestä palvelinohjelmalle ja palvelimen toiminnon suorituksen vastauksen välittämisestä takaisin asiakkaalle [10, s. 255]. Rajapinnan määrittelykielellä määritellään CORBA-objektin rajapinta. Rajapinnan määrittelykielen kääntäjä luo objektin rungon määrittelystä halutulla ohjelmointikielellä. Käyttäjälle näkyvän objektin rungon metodeja kutsumalla käyttäjä pystyy kutsumaan palvelimella olevan objektin toteutuksen metodeja. [6, s. 41–42]

3.1.3 Publish/Subscribe

Pub/Sub eli Publish/Subscribe on MOM, joka laajentaa MOM-teknologiaa mahdollistamalla useamman vastaanottajan tai lähettäjän viestille. Toisin kuin tavanomaisissa MOMeissa, Pub/Sub -teknologiassa viestit eivät jää viestijonoon odottamaan lukua, vaan ne välittyvät heti kuunteleville komponenteille. [11]

Publisher eli ”julkaisija” lähettää viestin valitun aiheen (*topic*) alle. Aiheet ovat normaalien MOMien jonon vastine Pub/Subille. Subscriber eli ”tilaaja” seuraa sitä kiinnostaviin aiheisiin tulevia viestejä. Pub/Sub -ohjelmisto jakaa aiheeseen tulleen viestin eteenpäin jokaiselle aihetta kuunnelleelle tilaajalle. Aiheen julkaisijoiden ja tilaajien määrä voi vaihdella dynaamisesti ohjelman suorituksen aikana. [6, s. 50]

3.1.4 Viestinvälittäjä

Kun ohjelmiston keskenään kommunikoivat komponentit eivät ole täysin yksimielisiä viestien muotoilusta, MOMin käyttöönotosta tulee haasteellisempaa. Tällaisissa tilanteissa viestinvälittäjä (*message broker*) saattaa olla sopiva ratkaisu. Viestinvälittäjä laajentaa tavanomaista MOMia niin, että välittäjän sisäisesti on mahdollista käsitellä viestiä. [6, s. 81–84]

Viestinvälittäjään voidaan ohjelmoida viestin muokkaamista, jotta viesti vastaa vastaanottavan komponentin rajapintaa. Viestinvälittäjän tehtävä on siis muuntaa viesti oikeaan muotoon ja lähettää se vastaanottajalle. [6, s. 84–85; 21, s. 119] Muiden MOMien tavoin, viestinvälittäjiä voi klusteroida suuremman tehokkuuden ja skaalautuvuuden saavuttamiseksi [6, s. 87]. Viestin säilömisen ja lähettämisen luotettavuuden viestinvälittäjät mahdollistavat viestien säilömiseen tarkoitetulla viestijonolla [13]. Viestinvälittäjät voivat välittää viestin joko yhdeltä komponentilta yhdelle vastaanottajalle, tai Pub/Sub -mallin tyylisesti useilta komponenteilta useille vastaanottajille [13; 21, s. 119].

3.1.5 SOA ja WS

Palvelukeskeinen arkkitehtuuri (*Service-oriented architecture, SOA*) ja sitä varten räätälöidyt verkkopalvelut (*Web services, WS*) muodostavat yhdessä väliohjelmistoarkkitehtuurin. Niiden tavoitteina on mahdollistaa ohjelmistojen aikaista parempi yhteentoimivuus ja skaalautuvuus, samalla yrittäen asettaa yleiset standardit integraatioon. WS mahdollistaa aiemmin mainittujen hajautettujen järjestelmien väliohjelmistojen tavoin toiminnallisuuden käyttämistä verkon välityksellä SOA-järjestelmissä. Suurin ero muihin väliohjelmistoihin on järjestelmien yhteentoimivuus ja eri kielistä ja alustoista koituvien käytännön ongelmien ratkaisu. SOA ja WS eivät siis yritä tuloksetta poistaa ohjelmistojen välistä monimuotoisuutta, vaan ovat ratkaisu niiden väliseen integraatioon. [6, s. 65–66]

SOAlla luodaan autonomisista palveluista järjestelmä. SOA-järjestelmässä toiminnallisuudet ovat jaettu omiin kokonaisuuksiinsa, jotka vuorovaikuttavat suoraan toistensa kanssa alustoistaan riippumatta. SOAn arkkitehtuurisella paradigmalla mahdollistetaan palveluntarjoajan ja asiakkaan välinen vuorovaikutus palveluiden avulla. Palvelulla tarkoitetaan palveluntarjoajan palvelua, jolla tuotetaan asiakkaalle haluttu lopputulos tai toiminto. [12, s. 68–69]

Verkkopalvelut suunniteltiin integraatiostandardeiksi erityisesti SOA-järjestelmien vaatimuksia varten. Yhteentoimivuuteen ja yksinkertaisuuteen tavoittelemista huomioon ottamatta verkkopalvelut ei paljolti eroa muista väliohjelmistoista. [6, s. 71] Kaikkien verkkopalveluiden rajapinnat ovat määritelty XML-merkintäkielellä, jolloin ne ovat kuvaavia ja helppolukuisia. Verkkopalveluiden avulla löyhästi toisiinsa kytketyt komponentit kommunikoivat XML-pohjaisia rajapintoja käyttäen. Palveluntarjoaja siis vastaa palvelun asiakkaan lähettämään palvelun XML-kutsuun XML-viestillä. [12, s. 99]

Verkkopalveluita varten on useita standardeja. SOAP on verkkopalveluiden alkuperäinen standardi ja oli Gortonin mukaan ainakin vielä 2011 vuonna standardeista tärkein ja käytetyin. SOAP määrittelee XML-pohjaisen, yksinkertaisen ja kattavan kommunikoinnin

protokollan verkkopalveluiden kutsumiseksi. SOAP-viestit ovat XML-pohjaisia ja ne voidaan välittää millä tahansa tiedonsiirtoprotokollalla. [6, s. 73–74] Sittemmin suosioon on noussut REST (*Representational State Transfer*), joka joidenkin lähteiden mukaan on ollut käytössä SOAPia enemmän jo ennen vuotta 2011 [17]. REST API on verkkopalvelu, joka käyttää REST-arkkitehtuuria kutsun käsittelyyn. REST APIt käyttävät tiedonsiirtoon yleisimmin HTTP-protokollaa. [18] REST APIen siirtämä data on usein JSON- tai XML-muotoista. Aikaisemmin XML on ollut suositumpi dataformaatti, mutta lähivuosina JSONin käyttö on yleistynyt yli XML:n [19].

3.1.6 ESB

Liikepalveluväylä (*Enterprise service bus, ESB*) on standardeihin perustuva väliohjelmisto, jossa yhdistyy viestintä, verkkopalvelut, datan muokkaus ja reititys [15, luku 1]. ESB tarvitsee edellä mainitut ominaisuudet, koska se on käytössä SOA-järjestelmissä.

ESB tekee palvelinpuolen järjestelmistä palvelurajapintoja, joita muut ohjelmat voivat käyttää. Koska ESB on käytössä SOAn kanssa, rajapinnoilla on yleiset standardit kommunikointiin, mikä mahdollistaa uusien palveluiden nopean liittämisen osaksi ohjelmaa ilman ylimääräistä integraatiota. SOA-järjestelmän palvelut ovat usein vanhojen järjestelmien funktioita, joiden rajapinta on julkistettu. Usein vanhojen järjestelmien protokollat ja datan muodot vaativat muokkausta toimiakseen SOAn tiedonsiirtoprotokollilla. ESB:n avulla datan ja protokollan muutokset tapahtuvat ESB:n sisällä samalla, kun data välittyy sen kautta. [14]

3.2 Integraatioalusta palveluna

Integraatioalusta palveluna (*Integration platform as a service, iPaaS*) on pilvipalveluiden kokoelma, joka mahdollistaa komponenttien ja ohjelmistojen integraation ilman ylimääräistä laitteistoa tai väliohjelmistoa. iPaaSin avulla käyttäjä voi luoda ja hallita komponenttien välillä tapahtuvaa kommunikointia [20].

IPaaS mahdollistaa tapahtumapohjaisen synkronoinnin, datan formatoinnin komponenttien välillä sekä tarjoaa valmiita sovittimia erilaisten applikaatioiden kytkemiseen. Lisäksi iPaaS tarjoaa mahdollisuuden edellä mainittujen toiminnallisuuksien kehittämiseen. IPaaS-järjestelmät ovat suunniteltu helposti käytettäviksi ja valmiina saatavilla olevien toiminallisuuksien ja helppojen konfiguraatiomenetelmien vuoksi integraatioon kuluva aika ja vaiva on mahdollisimman pieni. Helppokäyttöisyys johtuu siitä, että iPaaS on teknologiana suhteellisen uusi ja sen kehityksessä on erityisesti panostettu käytettävyyteen. [22] Pilvipalvelujen tarjoaja hallitsee integraatioalustaa ja tarjoaa sitä asiakkaille palveluna. Asiakas tilaa alustan ja siihen haluamansa työkalut ja palvelut, jotka se tarvitsee

integraatioon. Palveluntarjoajan vastuulla on tietoturva, datan hallinnointi sekä laitteiston ja ohjelmistojen ylläpito ja päivitys. [23]

4. INTEGRAATIOTEKNOLOGIOIDEN OMINAISUUKSIA

Ennen kuin ohjelmistoon valitaan integraatioteknologia, tulee suunnitella tarkkaan, mitä ohjelmistolta haluaa. Tuleeko ohjelmistosta pienikokoinen pienelle määrälle yhtäaikaisia käyttäjiä vai onko kyseessä suuri ohjelmisto suurelle yleisölle? Onko näkyvyys sisäverkossa riittävä vai pitääkö ohjelmiston olla saatavilla verkon välityksellä mistä tahansa? Minkälaisia ovat ohjelmistot, joihin halutaan integroida nyt tai mahdollisesti tulevaisuudessa? Onko ohjelmistossa tärkeätä nopea vasteaika ja korkea tehokkuus vai luotettava kommunikointi ja virheensieto? Näitä ja monia muita kysymyksiä on mietittävä, kun valitsee sopivaa integraatioteknologiaa ohjelmistoon.

Jokainen tässä työssä esitelty teknologia on syntynyt jostakin syystä. Jokaisella teknologialla on yritetty korjata jokin tietty integraation haaste. Kaikilla teknologioilla on kuitenkin haittapuolensa. Haasteita ratkaistaessa on jouduttu uhraamaan muita ominaisuuksia; luotettavuuden hintana on usein suoritusteho ja yksinkertaisuuden hintana on yleensä luotettavuus ja suoritusteho.

Yritysohjelmisto on usein kooltaan suuri ja siihen on sijoitettu mittava määrä yrityksen pääomaa ja työtunteja. Mittavien sijoituksien vuoksi yrityksille on elintärkeää, että integraatioon valitaan oikea teknologia, koska valittavan integraatioteknologian ominaisuudet voivat vaikuttaa suuresti ohjelmiston ominaisuuksiin. Jo toteutetun, väärin perustein valitun, teknologian korjaaminen tai muuttaminen toiseksi on kallista, riskialtista ja aikaa vievää.

4.1 Väliohjelmistot

Väliohjelmistojen avulla lähde- ja kohdejärjestelmien monimutkaisuudet eivät näy ohjelmoijalle, jolloin hän voi keskittyä tiedon jakamiseen ja välittämiseen. Samaa väliohjelmiston rajapintaa saatetaan käyttää useassa erilaisessa ohjelmistossa sekä useilla eri alustoilla. Kyseisellä rajapinnalla pystytään tällöin piilottamaan yksityiskohtia yhdistettävistä osapuolista ja alustasta. [16, luku 7, ”Middleware: The engine of EAI”]

Edellä mainitun abstraktion lisääminen vähentää järjestelmän monimutkaisuutta huomattavasti. Toisaalta jokainen väliohjelmisto vaatii järjestelmältä resursseja, mikä voi vaikuttaa järjestelmän suorituskykyyn, skaalautuvuuteen ja muihin tehokkuuden ominaisuuksiin. [21, s. 107]

4.2 Viestintään tarkoitetut väliohjelmistot

Toisin kuin esimerkiksi etäproseduurikutsut, MOMit ovat yleensä asynkronisia, jolloin ne kytkevät komponentit vain löyhästi toisiinsa. Asynkronisuuden vuoksi MOMia käytettäessä, kutsuttavan kohteen suoritusta ei jäädä odottamaan. Kutsun jälkeen kutsuva järjestelmä siis jatkaa toimintaansa välittömästi. Lisäksi MOMien avulla voidaan varmistaa, että viesti saapuu kohteelle riippumatta kohteen tai verkon senhetkisestä tilasta. Viestien käyttö on myös etu, koska datamäärältään pieniä viestejä on helppo käsitellä ja hallinnoida. Tietynlaiset MOMit pystyvät jakamaan viestin myös useille vastaanottajille. MOM-tuotteita pystyy myös parantelemaan muun muassa priorisoinnilla, rinnakkaisuudella ja kuormituksen tasaamisella. [16, luku 7, ”Message-oriented”] MOMit pystyvät takaamaan viestin välittymisen vastaanottajalle, vaikka vastaanottaja ei olisi tavoiteltavissa välittömästi [16, luku 9, ”Message-Oriented Middleware”].

MOMit ovat alustariippumattomia ja toimivat yhdessä useiden erilaisten teknologioiden kanssa. MOMin käyttöönotossa komponentteja ei tarvitse kirjoittaa uudelleen eikä tehdä niihin suuria ja mahdollisesti riskialttiita muutoksia. MOM on usein toteutettu palvelimena, jolloin se voi käsitellä useita samanaikaisia viestejä. MOMeilla on mahdollisuus taata viestin perille vieminen järjestelmävirheistä huolimatta. Jos MOM palvelin kaatuu, komponentit eivät voi kommunikoida toistensa kanssa. Ratkaisuna tähän on MOM-palvelimien klusterointi, jolloin ohjelma toimii huolimatta yhden palvelimen kaatumisesta. Klusteroinnin hyötynä on myös viestien tasainen jakautuminen useille palvelimille, jolloin viestien välittäminen on nopeampaa. MOM-teknologiaa voidaan käyttää myös tiukemmin kytkevään, synkroniseen kommunikointiin. Jos yrityksellä on kokemusta väliohjelmista, MOM-teknologian käyttöönotto on edullista ja suurien virheiden riski on alhainen. [6, s. 43–49]

4.3 Etäproseduurikutsu

Etäproseduurikutsujen suurin ja yksi ainoista eduista on niiden helppokäyttöisyys ja selkeys. Helppokäyttöisyyden ja selkeyden vastapainoina ovat synkronisuus, heikko suoritusteho sekä suuri vaadittu kaistanleveys. Synkronisuuden vuoksi, kutsuvan järjestelmän on odotettava kutsuttavan järjestelmän vastausta etäproseduurikutsuun. [16, luku 7, ”RPCS”] Kutsun suorittamisen odottaminen luonnollisesti laskee ohjelmiston suorituskykyä huomattavasti. [16, luku 9, ”RPCs”] Yksi etäproseduurikutsu vaatii useita pyyntöjä verkon välityksellä, jolloin hidas internetyhteys on este kutsujen nopealle suorittamiselle. Yhden kutsun suorittaminen sisältää useita eri vaiheita verkon yli tehtyjen kutsujen lisäksi. [16, luku 7, ”RPCS”]

Etäproseduurikutsut ovat helppo toteuttaa ja ymmärtää, koska kutsu näyttää normaalilta funktiokutsulta. Kutsut ovat etäkutsuja, joten ne ovat suhteellisen hitaita kulkiessaan ORBin ja verkon välityksellä. Koska objektipohjaiset väliohjelmistot ovat synkronisia ja kytkevät komponentit toisiinsa tiukasti, hajautetuilla järjestelmillä on aina se riski, että jokin palvelin ei ole saatavilla, jolloin ohjelma ei toimi halutulla tavalla. [6, s. 43]

4.4 CORBA

Yksinkertaisuuden lisäksi objektipohjaisten väliohjelmistojen toinen suuri etu on kuuliaisuus yhteensopiville standardeille. Toimittajat noudattavat vakiintuneita standardeja, minkä vuoksi ne tarjoavat muiden ohjelmistojen kanssa yhteensopivia teknologioita. Standardien rajapintojen laadun ja avoimuuden vuoksi toimittajien on helppo tarjota yhteensopivia teknologioita. [16, luku 10, ”What's So Easy?”]

Yksinkertaisuuden hintana on tiukasta kytkennästä johtuva kommunikoinnin epävarmuus. Jos kohdeobjekti ei ole saatavilla tiukasti kytketyssä ohjelmistossa, väliohjelmisto ei jää odottamaan kohdetta, vaan suoritus palaa kutsuvaan komponenttiin. Objektipohjaiset väliohjelmistot myös skaalautuvat huonosti suurempiin ohjelmistoihin, koska niissä on harvoin roskienkeruuta, kuormituksen tasaamista tai rinnakkaisuutta. [16, luku 10, ”The Realities”] Objektipohjaisen väliohjelmiston onnistunut käyttöönotto vaatii ohjelmiston hajauttamista. Ohjelmiston lähdekoodia tarvitsisi muokata suurelta osin ja jokainen ohjelmiston osa pitää testata ja ottaa sen jälkeen uudelleen käyttöön. [16, luku 10, ”What's So Difficult?”]

4.5 Publish/Subscribe

Publish/Subscribe on MOM-teknologia, joka mahdollistaa viestinnän myös yhdeltä komponentilta monelle ja monelta komponentilta monelle [6, s. 50]. Pub/Sub vapauttaa ohjelman tarpeesta ymmärtää vastaanottajasta mitään. Ohjelman tarvitsee ainoastaan lähettää haluamansa data väliohjelmistolle ja väliohjelmisto välittää sen kaikille kiinnostuneille osapuolille. Pub/Sub tarjoaa vastaanottaville osapuolille saman vapauden. Vastaanottajat eivät tiedä lähettäjästä mitään, vaan ovat kiinnostuneita vain tulevasta informaatiosta. [16, luku 7, ”Publish/Subscribe”]

Samannimisiä aiheita voidaan monistaa eri palvelimille. Jos yksi palvelin ei ole saatavilla, julkaisija lähettää viestit monistetulle jonolle, jolloin viestin vieminen perille on luotettavampaa. Pub/Sub kytkee komponentit löyhästi toisiinsa, jolloin ohjelmiston muokattavuus säilyy. Uusia tilaajia ja julkaisijoita voidaan lisätä ohjelmistoon ilman arkkitehtonisia muutoksia tai konfiguraatioita. Viestien datan muodon muuttaminen saattaa vaatia myös

tilaajien toteutuksien muuttamista. [6, s. 105] Toisaalta, Pub/Sub yleensä sisältää kyvyn viestien muokkaukseen, jolloin komponentit, jotka eivät ole suunniteltu vuorovaikuttamaan toisiensa kanssa, pystyvät kommunikoimaan [21, s. 113].

4.6 Viestinvälittäjä

Viestinvälittäjä mahdollistaa integraatioon liittyvän liiketoiminnan logiikan suorittamista välittäjän sisäisesti. Viestinvälittäjän avulla komponentti voi antaa välittäjälle viestin haluamassaan muodossa ja välittäjä muokkaa viestin vastaanottajalle sopivaksi. Tavallisimmillaan viestinvälittäjät ovat tehokkuudeltaan optimoituja. Sen vuoksi välittäjät loistavat, kun muutokset ovat lyhytikäisiä ja nopeasti suoritettavia. Tehokkuutensa vuoksi viestinvälittäjät voivat nopeasti aloittaa viestin muokkaamisen alusta, jos välittäjä kaatuu. Koska muokkaus voidaan aloittaa alusta, viestinvälittäjien ei tarvitse turvautua resursseja vaativaan tilojen hallinnointiin. [6, s. 84–87]

Toisaalta useat ohjelmistot vaativat monia peräkkäisiä kutsuja, muutoksia ja päivityksiä. Tämänlaisissa ohjelmistoissa, joissa tapahtuu suuri määrä kommunikointia, viestinvälittäjät eivät ole toimiva ratkaisu, koska ne saattavat aiheuttaa pullonkaulan. Pullonkaulan välttämiseksi useita viestinvälittäjiä on mahdollisuus klusteroida, jolloin välittäjien aiheuttama pullonkaula pienenee tai häviää. Klusteroinnin seurauksena välittäjien kehittäminen monimutkaistuu ja ylläpitokustannukset kasvavat. Haittapuoli viestinvälittäjän viestin muokkaamisessa on se, että muutoslogiikka pitää itse määritellä. Yksinkertaisille muutoksille tämä ei ole ongelma, mutta on mahdollista, että viestien muuttaminen vaatii useita monimutkaisia toimenpiteitä, jolloin muutoslogiikan tuottaminen on haaste. [6, s. 85–93]

4.7 SOA ja WS

Suurimmat ohjelmistoyritykset ovat suostuneet tukemaan verkkopalveluiden yhteisiä standardeja. Koska standardeja noudatetaan, verkkopalvelut eivät ole riippuvaisia alustasta tai ohjelmointikielestä. Verkkopalveluiden ei tarvitse tietää mitään asiakassovelluksesta ja ne voivat vastaanottaa pyyntöjä mistä tahansa, jos pyynnöt vastaavat oikeata formaattia ja täyttävät turvallisuusvaatimukset. Verkkopalveluiden omistaja pystyy ottamaan käyttöön ja hallita palveluitaan täysin. Palveluiden omistaja voi muuttaa palveluidensa määrittelyjä, vaatimuksia ja toteutuksia milloin tahansa. [6, s. 65–69]

Versioiden yhteensopivuus on verkkopalveluiden yksi pitkäaikaisimpia vaikeuksia, jota vaikeuttaa edelleen palveluiden avoimuus. Palveluiden omistaja voi muuttaa palvelujaan

oman tahtonsa mukaan, kunhan vanhempien versioiden kutsut toimivat uudessa versiossa. Verkkopalvelut, jotka ovat käytettävissä verkon välityksellä, ovat vastuussa omasta tietoturvastaan ja palveluiden omistajan tarvitsee suojata palvelut haitallisten kutsujen varalta. [6, s. 69]

4.8 ESB

ESB valjastaa verkkopalvelut ja muut täydentävät standardit yhdistämällä ne viestinvälittäjien parhaiden ominaisuuksien kanssa. Tästä seuraa, että ESB pystyy viestinvälittäjän tavoin muokkaamaan dataa vastaanottavan järjestelmän hyväksymään muotoon. ESB kykenee muodostamaan laajan verkon ydinosan. Suurien maailmanlaajuisten yritysten toimipisteiden välisten integraatioiden lisäksi ESB on myös täysin kykenevä pienempiin, paikallisiin integraatiotehtäviin. ESB tarjoaa joustavat tuennat, joiden avulla sitä voidaan käyttää kaikenlaisissa ympäristöissä. ESB kykenee saavuttamaan korkean turvallisuuden tason autentikaation ja pääsytietojen ja pääsylupien hallinnoinnin avulla. ESB:n luotettavuus saavutetaan valitsemalla sen ytimeksi sopiva MOM, joka kykenee asynkroniseen kommunikaatioon, luotettavaan datan toimittamiseen ja eheään vuorovaikutukseen. [15, Luku 1] Kehittäjät voivat käyttää kommunikointiin ESB:n kanssa protokollaa, antaakseen sille komentoja, jotka ohjeistavat vuorovaikutuksia palveluiden välillä [14].

Monissa organisaatioissa ESB:n käyttöönotto sujui onnistuneesti, toisissa organisaatioissa ESB huomattiin pullonkaulaksi SOA-kehityksessä. Integraatioiden kokojen kasvaessa, palvelimen saatavuuden takaamisen ja onnettomuudesta palautumisen toteuttamisen kustannukset kasvavat. Kustannuksien kasvamisen ja keskitetyn ESB:n ylläpidon ja päivittämisen haasteiden vuoksi ESB:n lupaamien tuottavuustekijöiden toimittaminen viivästyi, mikä johti asiakkaiden turhautumiseen. [14]

4.9 Integraatioalusta palveluna

Palvelussa on palveluntarjoajan toimesta etukäteen kehitettyjä sovittimia muihin yleisiin järjestelmiin, esimerkiksi toiminnanohjausjärjestelmiin, integraatiota varten. IPaaS mahdollistaa viestinvälittäjän tavoin datan muokkaamisen vastaanottavan järjestelmän hyväksymään muotoon. IPaaS mahdollistaa synkronisen, asynkronisen sekä aikatauluun tai tapahtumaan sidotun kommunikoinnin. [22]

Haasteena IPaaS-järjestelmissä on tietoturva johtuen siitä, että kommunikointi tapahtuu verkon välityksellä. Toinen ongelma tietoturvallisuuteen liittyen on se, että metadata, esi-

merkiksi salasanat, jaetaan palveluntarjoajan lisäksi mahdollisesti pilvipalveluiden infrastruktuurin tarjoajan kanssa. Koska integraatio toimii verkon välityksellä, heikko tai hidas verkkoyhteys johtaa kommunikoinnin hitaampaan toimintaan. Palveluntarjoajat luonnollisesti haluavat pitää asiakkaat pelkästään itsellään, mistä johtuu eri toimittajien alustojen rajallinen yhteensopivuus. [22]

5. PÄÄTELMÄT

Kuten luvussa 4 mainitaan, valittava integraatioteknologia on riippuvainen siitä, minkälaisia ominaisuuksia siltä haluaa. Objektiivisesti parasta integraatioteknologiaa jokaiseen ohjelmistoon ei ole olemassa. Käytettävä teknologia on riippuvainen ohjelmiston arkkitehtuurista, mahdollisista integroitavista järjestelmistä ja tulevaisuuden suunnitelmista. Jos ohjelmisto on tällä hetkellä pieni ja muita integroitavia osia on vain vähän, ei välttämättä ole silti kannattavaa valita teknologiaa, jonka toimintakyky murenee integraatiovaatimusten lisääntyessä.

Jos ohjelmistolle on tarpeellista hyvä suorituskyky ja korkea kommunikaation varmuus, viestintään tarkoitettu väliohjelmisto saattaisi olla oikea ratkaisu. Jos komponenttien rajapintojen data ei ole samassa muodossa, sekä Pub/Sub-teknologia että viestinvälittäjä pystyvät molemmat muokkaamaan dataa ennen sen lähettämistä vastaanottajalle. Viestinvälittäjä loistaa, kun yhtäaikaista kommunikaatiota ei ole liikaa. Jos yhtäaikaisen kommunikaation määrän oletetaan olevan suhteellisen vähäistä ja kommunikaatio tapahtuu vain yhdeltä komponentilta yhdelle, viestinvälittäjä ei aiheuta pullonkaulaa. Tämän kaltaisissa ohjelmistoissa viestinvälittäjää voisi käyttää integraation toteuttamiseen. Kun kommunikaation määrä kasvaa ja mahdollisia viestien vastaanottajia ja lähettäjiä on enemmän kuin yksi tapahtumaa kohden, Pub/Sub-teknologia vastaa paremmin ohjelmiston vaatimuksiin.

RPC, CORBA ovat vanhoja teknologioita, joiden haittapuolet ylittävät väliohjelmistojen edut. Epävarma, synkroninen ja hidas kommunikaatio ei ole haluttu ominaisuus, vaikka väliohjelmistojen käyttö olisikin kehittäjille helppoa ja yksinkertaista. Näitä väliohjelmistoja on silti käytettävä silloin kun vanhan ohjelmiston arkkitehtuurin muuttaminen on liian kallista ja riskialtista.

Jos tavoitteena on julkistaa ohjelmiston rajapinta kaikille saatavaksi, on REST-arkkitehtuurilla toteutettu verkkopalvelu paras vaihtoehto. Edellä mainittu, todennäköisesti tällä hetkellä maailman käytetyin, integraatioteknologia mahdollistaa helpon, skaalautuvan, nopean ja turvallisen kommunikoinnin. Verkkopalveluiden kanssa käytettävä ESB on myös toimiva ratkaisu, jos ohjelmiston rajapinnan data vaatii muokkausta, ennen kuin integroitava ohjelmisto hyväksyy datan muodon.

Pilvipalvelun käyttäminen integraatioalustana on hyvä ratkaisu pienemmille organisaatioille, jotka eivät halua sijoittaa suuria summia laitteistoihin. Koska useimmat palveluntarjoajat tarjoavat ”on demand” -laskutusta, iPaaS on pienemmille organisaatioille myös

käytettäessä edullinen. Jos integraatiovirta ei ole liian monimutkainen, tämän uuden teknologian käyttöönotto ja konfigurointi sujuu kivutta palveluntarjoajan käyttöliittymien kautta.

6. YHTEENVETO

Tutkielmassa tutkittiin tunnettuja yritysohjelmistojen integraatioteknologioita. Keskeisenä aiheena olivat väliohjelmistot ja muutama tunnettu väliohjelmistojen alalajien paradigma. Valittuja teknologioita ja paradigmoja tutkittiin tutkielmassa sen vuoksi, että ne olivat esillä useimmissa tutkituissa lähteissä.

Teknologioista selvitettiin niiden yleinen toimintaperiaate. Toimintaperiaatteiden lisäksi integraatioteknologioista selvitettiin niille ominaisia piirteitä ja ominaisuuksia. Teknologioista siis selvitettiin niiden ominaisuudet ja käyttäytyminen integraatiotehtävässä sekä mahdollisesti syyt siihen, miksi teknologiat käyttäytyvät kyseisellä tavalla. Samalla selvisi eri teknologien kipukohtia ja niitä varten kehitettyjä ratkaisuja. Toimintaperiaatteiden, piirteiden ja ominaisuuksien pohjalta pohdittiin eri teknologioille mahdollisia käyttötarkoituksia.

Johdannon tavoitteista molemmat saavutettiin. Tutkielmassa saatiin selville tunnettuja ja laajasti käytössä olevia integraatioteknologioita sekä selvitettiin niiden toimintaa ja ominaisuuksia. Tutkielmassa myös selvisi, että yhtä oikeana pidettyä integraatioteknologiaa ei ole olemassa, vaan jokainen integraatioteknologia on olemassa tietyn ongelman ratkaisemista varten. Yhtä oikeana pidettyä integraatioteknologiaa ei ole, koska jokaisen ohjelmiston vaatimukset ovat erilaiset, jolloin myös vaatimukset integraatioteknologioilta ovat erilaiset.

LÄHTEET

[1] Enterprise Software, Techopedia, 24.1.2017, https://www.techopedia.com/definition/7045/enterprise-software (viitattu 2.4.2020)

[2] Stephen Watts, Enterprise Application Software Defined: How Is It Different from Other Software?, BMC, 16.8.2017, https://www.bmc.com/blogs/enterprise-application-software-defined-how-is-it-different-from-other-software/ (viitattu 2.4.2020)

[3] Vangie Beal, Enterprise application, Webopedia, https://www.webopedia.com/TERM/E/enterprise_application.html (viitattu 2.4.2020)

[4] Martin Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley Professional, 2002, https://learning.oreilly.com/library/view/patterns-of-enterprise/0321127420/?ar (viitattu 2.4.2020)

[5] Parameswaran Seshan, Process-Centric Architecture for Enterprise Software Systems, 2010

[6] Ian Gorton, Essential software architecture, Springer, 2011

[7] Garima Verma, Khushboo Saxena, Sandeep Saxena, Distributed Systems, BPB Publications, 2017, https://masterworkshop.skillport.com/skillportfe/main.action#summary/BOOKS/RW$5836:_ss_book:135269 (viitattu 24.4.2020)

[8] Dominic Duggan, Enterprise Software Architecture and Design : Entities, Services, and Resources, IEEE Computer Society Press, 2012

[9] Object Management Group, CORBA History, Object Management Group, https://www.corba.org/history_of_corba.htm (viitattu 23.4.2020)

[10] Kenneth Birman, Guide to Reliable Distributed Systems, Springer, 2012 (3.5.2020 saatavilla https://link-springer-com.libproxy.tuni.fi/book/10.1007%2F978-1-4471-2416-0#toc)

[11] Amazon Web Services, Pub/Sub messaging, Amazon, https://aws.amazon.com/pub-sub-messaging/ (viitattu 5.5.2020)

[12] S. Anandamurugan, T. Priyaa, Service Oriented Architecture, Nova Science Publishers, 2014

[13] IBM Cloud Education, Message Brokers, IBM, 23.1.2020, https://www.ibm.com/cloud/learn/message-brokers (viitattu 11.5.2020)

[14] IBM Cloud Education, Enterprise Service Bus, IBM, 18.7.2019, https://www.ibm.com/cloud/learn/esb (viitattu 12.5.2020)

[15] David Chappell, Enterprise Service Bus, O'Reilly Media, 2004, https://learning.oreilly.com/library/view/enterprise-service-bus/0596006756/?ar (viitattu 12.5.2020)

[16] David Linthicum, Enterprise Application Integration, Addison-Wesley Professional, 1999, https://learning.oreilly.com/library/view/enterprise-application-integration/0201615835/ (viitattu 13.5.2020)

[17] Adam DuVander, New Job Requirement: Experience Building RESTful APIs, ProgrammableWeb, 9.6.2010, https://www.programmableweb.com/news/new-job-requirement-experience-building-restful-apis/2010/06/09 (viitattu 15.5.2020)

[18] IBM Cloud Education, REST APIs, IBM, 11.6.2019, https://www.ibm.com/cloud/learn/rest-apis (viitattu 15.5.2020)

[19] Wendell Santos, JSON is Clearly the King of API Data Formats in 2020, ProgrammableWeb 3.4.2020, https://www.programmableweb.com/news/json-clearly-king-api-data-formats-2020/research/2020/04/03 (viitattu 15.5.2020)

[20] Nicolas Serrano, Josune Hernantes, Gorka Gallardo, Service-Oriented Architecture and Legacy Systems, IEEE, 2014 (https://ieeexplore-ieee-org.libproxy.tuni.fi/document/6898686) (viitattu 17.5.2020)

[21] Vivek Kale, Guide to Cloud Computing for Business and Technology Managers, CRC Press, 17.12.2014

[22] Nico Ebert, Kristin Weber, Stefan Koruna, Integration Platform as a Service, 19.7.2017, https://link-springer-com.libproxy.tuni.fi/article/10.1007/s12599-017-0486-0 (viitattu 20.5.2020)

[23] IBM Cloud Education, iPaaS, 17.8.2019, https://www.ibm.com/cloud/learn/ipaas (viitattu 20.5.2020)