Geneeristen integraatioiden kehittäminen

Satu Santamaa

2019 Laurea

Laurea-ammattikorkeakoulu

Geneeristen integraatioiden kehittäminen

Satu Santamaa
Tietojenkäsittely
Opinnäytetyö
Marraskuu, 2019

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittely
Tradenomi (AMK)

Satu Santamaa

Geneeristen integraatioiden kehittäminen

Vuosi 2019 Sivumäärä 39

Tämän opinnäytetyön tarkoitus oli selvittää integraatiokehitystyössä sekä valvonnassa esiintyviä yleisimpiä ongelmakohtia ja mahdollisuuksia niiden ratkaisuun geneeristen integraatioiden kehittämisellä. Työ on tehty Digia Oyj:n jatkuvien palveluiden Ratkaisutuki-tiimin näkökulmasta heidän laajalti käyttämäänsä integraatioalustaa ajatellen. Tavoitteena oli selvittää kehitystyön sekä jälkeenpäin tapahtuvan valvonnan ja mahdollisten virhetilanteiden selvityksen helpottaminen noudattaen ajatusta geneerisemmistä asiakasriippumattomista liittymistä.

Opinnäytetyön teoriaosuus koostuu keskeisimmistä käsitteistä sekä yleisestä teoriasta liittyen integraatiokehitystyöhön ja järjestelmäintegraatioihin. Teoriaosuuden koostamisen apuna on käytetty saatavilla olevaa kirjallisuutta sekä sähköisiä lähteitä. Teoriaosuudessa käydään läpi integraatioiden perusteita ja sitä voidaan käyttää tukena esimerkiksi integraatioihin ja niiden kehitystyöhön tutustumisessa.

Tutkimusmenetelmänä opinnäytetyössä on käytetty kvalitatiivista tutkimusta. Aineistoa kerättiin tekemällä puolistrukturoitu haastattelu Ratkaisutuki-tiimin neljälle jäsenelle. Aineistonkeruu tehtiin kahdenkeskisillä haastatteluilla työpaikalla. Tehtyjä haastatteluja ei niiden sisällön vuoksi voitu julkaista opinnäytetyön liitteenä.

Haastattelujen tuloksena saatiin selville kehittäjien näkemyksiä integraatiokehityksen ja -valvonnan tämän hetkisistä ongelmakohdista. Vastauksissa geneeristen integraatioiden katsottiin mahdollistavan ratkaisuja tai helpotuksia esiintyviin ongelmiin, mutta myös todettiin olevan mahdollisesti hankalia toteuttaa kattavasti kaikissa integraatioissa. Tutkimuksen pohjalta voitiin kehitysehdotuksena nostaa esille mahdollisten integraatiokirjastojen luominen yhteiseen käyttöön sekä toimintatapojen yhtenäistäminen uusia liittymiä kehitettäessä niiltä osin kuin se on mahdollista.

Asiasanat: integraatio, järjestelmäintegraatio, geneerinen, kehitystyö

Laurea University of Applied Sciences **Abstract**
Degree Programme in Business Information Technology
Bachelor's Thesis

Satu Santamaa

Developing generic integrations

Year 2019 Pages 39

The purpose of this thesis was to define the problems that occur with developing and monitoring integrations and the possibilities to resolve these problems with generic integration development. This research was made from the perspective of integration development at continuous services in Digia Oyj considering the integration platform which had been developed in the company. The objective was to examine the possibility of using more generic integrations to mitigate the amount of work, which is done in developing, monitoring and problem solving with integrations.

The theoretical part discussed the concepts and main theory of integration development and system integrations. It was written using the literature of the field and electronic sources. It contains the basics from integrations and can be used for example as an induction to integrations and integration development.

The method used in this thesis was qualitative research. The data was collected with semi-structured interviews from four of the members of continuous services. The interviews were conducted face-to-face for one of the interviewees at a time. Because of the content of these interviews they cannot be published as an attachment of the thesis.

The interviews showed the insights of the problems in integration development and monitoring. In the answers the genericity of integrations was seen as a possibility to resolve and ease some of the problems but also was recognized as probably impossible to implement in all of the integrations. As a conclusion based on this research are recommendations for creating libraries of more generic integrations for everyone's use and unifying the procedures of the development of new integrations as much as it is possible.

Keywords: integration, development, system integration, generic

Sisällys

1 Johdanto

Tutkimuksen aiheena oli geneeriset integraatiot toimeksiantajayrityksen integraatiokehityksessä ja tavoitteena oli selvittää mahdollisuudet suunnata integraatiokehitystyön kehittämistä geneeristen integraatioiden suuntaan toimeksiantajayrityksessä. Tutkimuksella pyrittiin selvittämään näiden eri ympäristöihin soveltuvien yleiskäyttöisten liittymien hyödyt ja mahdolliset haitat yrityksen integraatiokehitystyössä. Yleiskäyttöisillä liittymillä tarkoitetaan tässä työssä nimenomaan sellaisia geneerisiä integraatioliittymiä, joita voidaan käyttää useammassa, kuin yhdessä ympäristössä ja se on toiminnaltaan kaikissa samanlainen eikä vaadi käyttöön otettaessa muutoksia kuin konfiguraatioihin. Näiden liittymien kehitystyöllä ja käytöllä uusissa integraatioissa voitaisiin vaikuttaa positiivisesti tehtävän kehitystyön työmääriin. Tällä pystytään vaikuttamaan muun muassa toteutettavan palvelun kustannustehokkuuteen, toimitusaikaan sekä -varmuuteen. Lisäksi myöhemmin tehtävä integraatioiden ylläpito, valvonta ja virheenselvitys helpottuisi. Teoriataustaa tutkimukselle selvitetty aihealueen kirjallisuudesta ja muista julkaisuista. Tutkimustyön tueksi tehtiin puolistrukturoitu haastattelu toimeksiantajayrityksessä työskentelevälle neljälle eri osaamistaustan omaaville kehittäjille.

Toimeksiantajayrityksessä oli tutkimusta tehdessä jo käytössä muutamia geneerisiä integraatiomoduuleja yrityksen omassa integraatioalustassa yhdistämään tuoteperheen ohjelmat toisiinsa. Opinnäytetyötä tehdessä työskentelin itse toimeksiantajayrityksen Digia Oyj:n jatkuvien palveluiden Ratkaisutuki-tiimissä. Työni kohdistui pääasiassa integraatioiden kehittämiseen sekä valvontaan ja mahdollisten virhetilanteiden selvitykseen. Tutkimus on tehty selvitystyönä toimeksiantajayrityksen käyttöön integraatiokehityksen parantamista ajatellen. Opinnäytetyön merkitys integraatioiden kehitykselle on aikaansaada selvitys näiden moduulien kehityksen tarpeelle sekä mahdolliset puitteet, joiden mukaisesti jo alkanutta kehitystyötä ylläpidetään.

2 Tausta ja lähtökohdat

Tällä hetkellä integraatioiden ylläpitoa ja kehitystä hankaloittavat mm. ympäristöjen eroavaisuudet keskenään sekä dokumentaatioiden puutteellisuus tai puuttuminen. Eri ympäristöissä sanomaliikenne saattaa noudattaa keskenään teoriassa samaa kaavaa, mutta toteutukset poikkeavat toisistaan huomattavasti aiheuttaen kuormaa ympäristöjen ylläpidolle. Kuormitusta saattaa aiheuttaa myös ylläpitäjän tietämättömyys eri ympäristöjen integraatiototeutusten eroavaisuuksista. Tämän lisäksi on mahdollista, että ympäristön dokumentaatio on puutteellista tai sitä ei ole lainkaan. Nämä saattavat hidastaa ylläpitoprosesseja kehittäjälle ennalta tuntemattomassa asiakasympäristöissä ja mahdollisten virhetilanteiden selvitykset saattavat venyä tarpeettoman pitkiksi.

Opinnäytetyön aikaansaama ymmärrys geneeristen integraatioiden tuomille mahdollisuuksille kehitystyössä edesauttaa integraatioiden tulevaa kehitystä ja ohjaa nykyisiä sekä tulevia

kehittäjiä integraatioiden yhdenmukaiseen kehittämiseen. Yhdenmukaisella kehittämisellä voidaan saavuttaa pidemmällä tähtäimellä jopa asiakasriippumattomia integraatioratkaisukokonaisuuksia eli arkkitehtuureja, joiden käyttöönotto ja ylläpito on nykyistä huomattavasti kustannustehokkaampaa ja niitä ylläpitävälle osapuolelle suoraviivaisempaa.

Jatkuvan kehitystyön myötä mahdollisesti saavutettavat asiakasriippumattomat integraatioratkaisut edesauttaisivat uusien ympäristöjen käyttöönottoa, niiden jo sisältäessä yleiskäyttöiset, geneeriset integraatioratkaisut tuotteistettuna suoraan integraatioalustalle. Integraatioympäristön pystyttäminen ei siis enää vaatisi varsinaisten liittymien tekoa, vaan valmiiden ratkaisujen implementointia uusiin ympäristöihin. Tällä saataisiin vähennettyä kustannuksia niin palvelun tuottajalta kuin asiakkaalta integraatioiden osalta.

Viitekehyksenä opinnäytetyössä voidaan käyttää integraatioarkkitehtuuriin liittyvää aineistoa. Lisäksi erilaisiin tietomalleihin perustuvaa aineistoa on tarpeen käyttää asian selkeyttämiseksi.

2.1 Tutkimuskohteen kuvaus ja tutkimuksen tavoite

Tutkimuksen taustana on toimeksiantajayrityksessä laajassa käytössä olevan, heidän omaan käyttöönsä kehittämän, Java-pohjaisen integraatioalustan kehittäminen tukemaan laajemmin jo kehitteillä olevia geneerisiä integraatioratkaisuja. Tutkimusta tehdessä yrityksellä oli jo käytössä Ratkaisutukitiimissä kehitettyjä geneerisiä integraatiomoduuleja yhden sovellustuoteperheen integraatioissa. Tavoitteena on aikaansaada selvitys näitä vastaavien ratkaisujen kehityksen tarpeelle yleisempään käyttöön integraatioissa ja jo alkaneen kehitystyön jatkaminen ja ylläpito. Tutkimuksen avulla selvitetään mahdollisuutta integraatiokehitystyön standardisoinnille ja tehostamiselle.

Tutkimuksen kohteena on geneeristen integraatioiden kehityksen tarve sekä niiden mahdollisen käytön tuomat edut ja mahdollisuudet toimeksiantajayrityksessä. Tutkimuksella pyritään selvittämään näiden eri ympäristöihin soveltuvien yleiskäyttöisten liittymien kehitystyöhön liittyviä käsitteitä ja lainalaisuuksia järjestelmäintegraatioiden kehitystyössä yleisesti. Tavoitteena on myös tuoda toimeksiantajayrityksessä esille nämä edut ja mahdollisuudet sekä löytää ratkaisuja geneeristen integraatioiden kehitykseen ja käyttöön yleisesti kehitystyössä.

2.2 Tutkimuskysymykset

Tällä opinnäytetyöllä on tarkoitus selvittää, voidaanko geneerisiä integraatioita käyttää toimeksiantajayrityksen integraatiokehitystyössä apuna ja onko nykyistä kehitystyötä sekä myöhemmin tehtävää valvontaa ja virheenkorjaustilanteita mahdollista parantaa geneeristen integraatioiden avulla. Tämän lisäksi pyrittiin selvittämään, onko geneerisiä integraatioita mahdollista käyttää yleisesti toimeksiantajayrityksen asiakkaiden liittymissä vai onko tällaiset yleiskäyttöiset liittymät mahdotonta toteuttaa.

Ongelman selvittämiseksi geneeristen integraatioiden teoriataustaa on selvitetty alan kirjallisuudesta ja muista julkaisuista. Tutkimustyön tueksi tehtiin puolistrukturoitu haastattelu eri osaamistaustan omaaville kehittäjille koskien integraatioiden kehittämistä yleensä ja heidän näkemyksiään geneerisistä integraatioista ja niiden mahdollisuuksista.

2.3 Aihealueen rajaus

Aiheen valintaan vaikutti mielenkiintoni nopeasti pystytettävistä ja helposti ylläpidettävistä integraatioympäristöistä. Integraatioiden ylläpito saattaa vaatia jatkuvaa kehitystyötä, joka hankaloituu, mikäli eri asiakkailla on toisistaan poikkeavia ympäristöjä, joissa kuitenkin sanomaliikenne saattaa toimia samoilla periaatteilla keskenään. Tutkimuksessa keskitytään sanomapohjaiseen integraatioalustaan, sen ollessa keskeinen kehitystyön kannalta.

Aihealue on pyritty rajaamaan toimeksiantajayrityksen jatkuvissa palveluissa tehtyihin integraatiokehitystöihin ja myöhemmin tehtävään valvontaan. Jatkuvien palveluiden piiriin kuuluu lukuisia erilaisia asiakasympäristöjä ja useita integraatioteknologioita, joista opinnäytetyön kannalta oleellisin on yrityksessä kehitetty Java-pohjainen integraatioalusta.

2.4 Keskeiset käsitteet

Järjestelmäintegraatio: Järjestelmäintegraatio on valikoima toimintatapoja ja teknologioita, joiden avulla muutoin keskenään yhteensopimattomat järjestelmät saadaan kommunikoimaan keskenään.

Väliohjelmisto: Järjestelmien välissä toimivaa integraatioita tarjoavaa ohjelmistoa (middleware) käytetään liittymien konfigurointiin, ylläpitoon ja valvontaan.

Sanoma: Sanoma on atominen datapaketti, informaatiota sisältävä itsenäinen kokonaisuus, jota voidaan välittää eteenpäin.

Rajapinta: Rajapinnat eli API:t (Application Programming Interface) ovat kehittäjien sovellukseen luomia väyliä, joiden kautta saadaan yhteys eri tasoille tai palveluihin sovelluksessa.

Adapterit ja konnektorit: Adapterit ja konnektorit tarjoavat yksinkertaisen rajapinnan integroitavan ohjelmiston tarjoamaa rajapintaa vasten.

Muuntimet: Muuntimilla lähettävän järjestelmän data saadaan muutettua vastaanottavan järjestelmän ymmärtämään muotoon.

Tietomallit: Tietomallilla tarkoitetaan datan rakenteen ja datan suoritettavan käsittelyn määrittelevää abstraktia mallia.

Kanoninen tietomalli: Kanoninen tietomalli (canonical data model, CDM) on tietomallityyppi, joka esittää datakokonaisuudet ja yhteydet yksinkertaisimmalla mahdollisella tavalla. Tunnetaan myös yleisenä tietomallina.

JSON: JSON (JavaScript Object Notation) on tekstipohjainen tiedon siirtoon ja säilytykseen soveltuva informaation esitystapa.

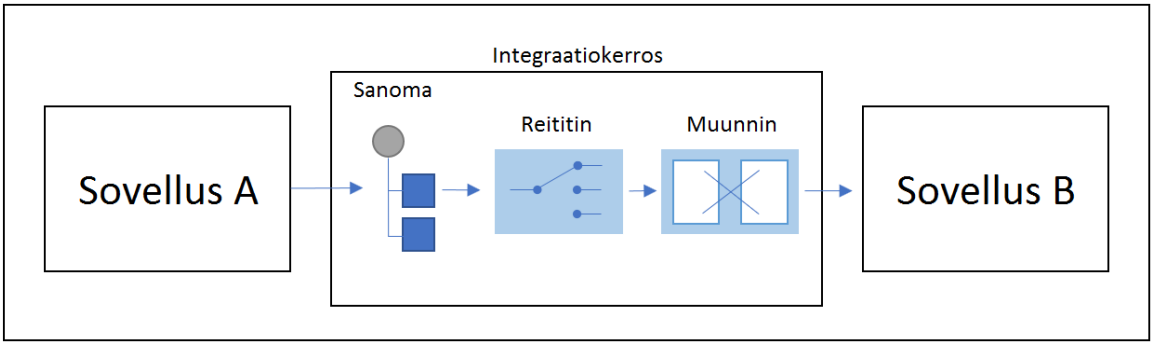
3 Järjestelmäintegraatiot

Integraatiot ovat yritysmaailmassa yleensä näkymättömiä, mutta tärkeitä tekijöitä. Järjestelmien välille rakennettujen liittymien avulla voidaan yhdistää useita sovelluksia keskenään yhtenäisiksi kokonaisuuksiksi, vaikka sovellukset itsessään eivät olisi yhteensopivia. Integraatioiden avulla yritykset voivat käyttää juuri heidän tarpeisiinsa soveltuvia sovelluksia, kuitenkin välittämättä niiden yhteensopivuudesta, mikäli sovelluksissa on integroimiseen soveltuvat rajapinnat.

Yrityksen tietojärjestelmä on looginen kokonaisuus, joka on rakennettu vastaanottamaan, prosessoimaan ja tuottamaan yrityksen toiminnalle tarpeellista tulosinformaatiota. Yrityksillä on nykyään päivittäisessä käytössä useita eri ohjelmistoja, joista rakentuu yrityksen tietojärjestelmäkokonaisuus. Tämä kokonaisuus voi koostua vanhemmista perinne- eli legacy-järjestelmistä, uudemmista järjestelmistä ja useista pienistä ohjelmistoista. Integraatioiden avulla toisistaan erillään olevat järjestelmät kyetään yhdistämään yhtenäiseksi tietojärjestelmäksi, jossa eri järjestelmät kommunikoivat yrityksen liiketoiminnalle oleellisella tavalla. Integraatiot voivat olla reaaliajassa yrityksen sisäisesti toimivia ja eri yritysten välisiä järjestelmäintegraatioita. (Tähtinen 2005, 14.)

Suppeasti määriteltynä järjestelmäintegraatiosta puhuttaessa teknisesti kyseessä on valikoima toimintatapoja ja teknologioita, joiden avulla muutoin keskenään yhteensopimattomat järjestelmät saadaan kommunikoimaan keskenään, jolloin ne muodostavat käyttäjän näkökulmasta yhtenäisen kokonaisuuden. Järjestelmäintegraatiolla saadaan sovellus kommunikoimaan esimerkiksi sovelluksen ulkoisen tietokannan kanssa tai yhdistetään toisistaan erilliset sovellukset yhtenäiseksi, keskenään reaaliaikaiseksi kokonaisuudeksi, jossa esimerkiksi toiseen

sovellukseen tehdyt muutokset informaatioon ovat käytettävissä reaaliaikaisesti myös toisessa sovelluksessa. (Kuvio 1.)

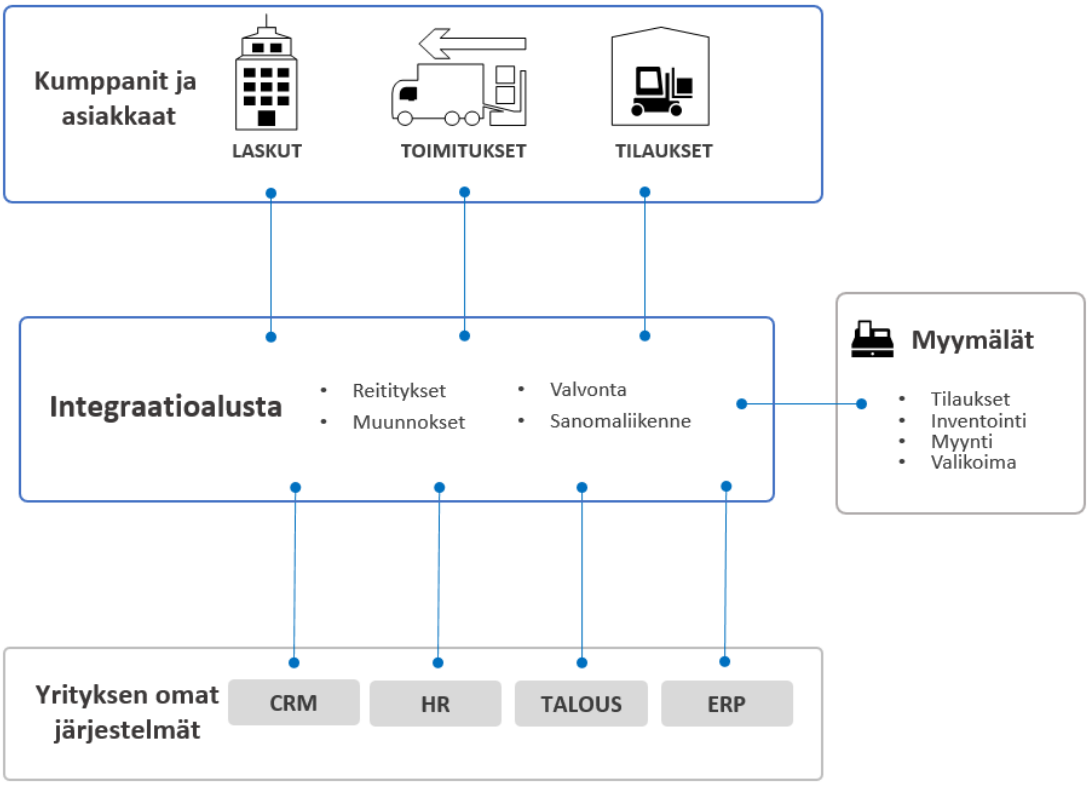


Kuvio 1: Yksinkertaistettu kahden sovelluksen integraatioesimerkki

Järjestelmäintegraatio voidaan rinnastaa hajautettuun sovelluskehitykseen, jonka päämääränä on rakentaa laajoja ja hajautettuja, mutta toisaalta yleensä varsin rajatun ongelman ratkaisuun suunnattuja tietoteknisiä sovelluksia. Tietotekniikan avulla tapahtuvalla integraatiolla tarkoitetaan informaation siirtämistä hallitusti erilaisten loogisia operaatioita sisältävien komponenttien välillä. (Tähtinen 2005, 16.)

3.1 Integraatiot yrityksissä

Integraation voidaan sanoa olevan se liima, jonka avulla yrityksen erilaisten sovellusten toiminnallisuudet kootaan yhdeksi kiinteäksi, hallittavaksi, kestäväksi ja tarvittaessa helposti muokattavaksi kokonaisuudeksi. Järjestelmäintegraatio on liima, mutta ei kuitenkaan yksittäinen tuote tai teknologia. Se on kokoelma erilaisia ajatus- ja suunnittelumalleja sekä käytäntöjä. Sen avulla useista yksittäisistä sovelluksista voidaan rakentaa yritykselle sen liiketoiminnan kannalta mahdollisimman käytännöllinen ja yrityksen tarkoitusperiin soveltuva kokonaisuus (Kuvio 2). (Tähtinen 2005, 14-15.)



Kuvio 2: Esimerkki useamman järjestelmän muodostamasta kokonaisuudesta

Yleisimmin integraatioita tehdään jo olemassa olevien järjestelmien väliin, jolloin saadaan rakennettua toimivia kokonaisuuksia joutumatta uusimaan järjestelmiä. Yrityksen nykyisten järjestelmien käyttöiän jatkaminen esimerkiksi integroimalla vanhempaan järjestelmään uudempi tai muutoin nykyisestä kokonaisuudesta puuttuvia ominaisuuksia omaava ohjelmisto, vähentää kustannuksia kehitystyöstä. Integroimalla vanhaan tietojärjestelmäkokonaisuuteen uusia ohjelmia, saadaan aikaan esimerkiksi yrityksen liiketoiminalle lisäarvoa sen kyetessä tuottamaan uuden ohjelmiston avulla vaikkapa uudenlaisia liiketoimintaan liittyviä suunnitelmia, raportteja tai arvioita, joilla voidaan vaikuttaa liiketoimintaan ja -tulokseen positiivisesti. Uuden ohjelmiston lisääminen yrityksellä käytössä olevaan tietojärjestelmäkokonaisuuteen on huomattavasti kustannustehokkaampaa, kuin mahdollisesti olemassa olevan järjestelmän osa-alueen kehittäminen ja räätälöinti yrityksen käyttöön soveltuvaksi. Uuden ohjelmiston käyttöönotto vaatii ohjelmallisesti minimissään vain ohjelmiston integroimisen aiemmin käytössä olevien järjestelmien kanssa. Näiden eri järjestelmien väliset integraatiot voivat olla yrityksen sisäisiä ja kahden tai useamman toistensa kanssa yhteistyötä tekevän organisaation välisiä järjestelmäintegraatioita. (Tähtinen 2005, 65-67.)

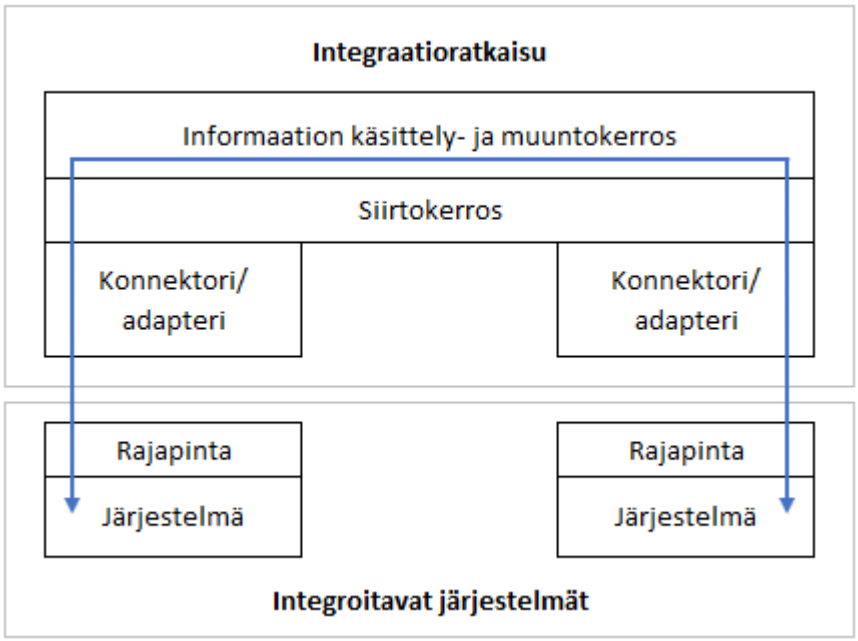
Järjestelmien väliset integraatiot rakennetaan niin, ettei integraatiossa kulkevan tiedon vuoksi tarvitse muuttaa järjestelmiä tai järjestelmien käyttämiä olemassa olevia tietorakenteita. Tämä edellyttää integraatioissa kyseiselle tietovirralle soveltuvaa muunninta, joka

muuntaa kahden eri järjestelmän välissä kulkevan datan lähettävän järjestelmän käyttämästä muodosta vastaanottavalle järjestelmälle käyttökelpoiseksi. Nämä muuntimet poistavat tarpeen tehdä muutoksia kumpaankaan järjestelmään. (Linthicum 2001, 10, 11)

3.2 Integraatioratkaisu

Järjestelmäintegraatio koostuu erilaisista toimintatavoista ja teknologioista, joiden avulla muutoin keskenään yhteensopimattomat järjestelmät saadaan kommunikoimaan keskenään. Tällöin ne muodostavan käyttäjän näkökulmasta yhtenäisen, jopa reaaliaikaisen kokonaisuuden, jossa toiseen sovellukseen tehdyt muutokset ovat käytettävissä toisessa, alkuperäisestä eriävässä sovelluksessa. Tietotekniikan avulla tapahtuvien integraatioiden avulla informaatiota voidaan siirtää hallitusti erilaisten loogisia operaatioita sisältävien komponenttien välillä. (Tähtinen 2005, 14, 16.)

Integraatioarkkitehtuurikokonaisuus voidaan koostaa lukemattomilla eri tavoilla. Joissain integraatioratkaisuissa saatetaan käyttää integraatiokerroksessa useita erilaisia muuntimia, dataa voidaan rikastaa lukemattomilla eri tavoilla ja lopputuloksena syntynyttä dataa tai sen osia voidaan toimittaa useammillekin eri vastaanottavalle järjestelmälle. Yksinkertaisuudessaan integraatioarkkitehtuuri sisältää integroitavat järjestelmät ja integraatiokerroksen sekä nämä yhdistävät rajapinnat ja konnektorit tai adapterit (Kuvio 3).



Kuvio 3: Yksinkertaistettu integraatioratkaisun malli

3.3 Väliohjelmisto

Järjestelmien välissä toimiva integraatioita tarjoava ohjelmisto (middleware) helpottaa liittymien sujuvaa ylläpitoa ja valvontaa. Tähän integraatioalustaan voidaan sisällyttää kaikki integraatioihin liittyvät monimutkaiset toimenpiteet ja sen avulla järjestelmien välisiä liittymiä ja niiden toimivuutta pystytään fasilitoimaan. (Linthicum 2001, 18.)

Alustaohjelmiston avulla voidaan yhdistää useita järjestelmiä toisiinsa ja välittää haluttua tietoa näiden toisistaan erillään olevien järjestelmien välillä käyttäen järjestelmien omia rajapintoja. Integraatioalusta tarjoaa kehittäjille helpon tavan integroida järjestelmiä käyttäen erilaisia ulkoisia resursseja, kuten tietokantapalvelimia, jonomanagereita ja API:a. (Linthicum 2001, 18.)

3.4 Sanoma

Sanoma on atominen datapaketti eli informaatiota sisältävä itsenäinen kokonaisuus, jota voidaan välittää eteenpäin. Jotta sanomapohjainen järjestelmä pystyy välittämään dataa, täytyy sen jakaa data yhdeksi tai useammaksi paketiksi, muodostaa jokaisesta paketista sanoma, viestiolio, ja lähettää sanoma eteenpäin. Sanomanvälittäjä ei kykene välittämään raakaa dataa, jolloin sanoman muotoon paketoitu data on välttämätön. (Enterprise Integration Patterns, 2017.)

Sanomalla on rakenne ja sisältö. Se koostuu otsikkotiedoista (headers) sekä tietosisällöstä eli varsinaisesta datasta (payload). Sanoman otsikkotiedot eli attribuutit sisältävät sanoman informaatiosisällön käsittelylle oleellista tietoa, kuten lähettäjän, vastaanottajan, sanoman sisällön tarkoitukseen, esitysmuotoon ja mahdolliseen salaukseen liittyviä parametrejä. Sanomasisältö eli varsinainen data sisältää järjestelmien välillä siirrettävän informaation. (Tähtinen 2005, 123-124.)

Tarvittaessa integraatiokerros voi esimerkiksi muuntimien avulla koostaa sanoman myös useista eri lähteistä. Sanomaa voidaan muokata integraatiossa esimerkiksi rikastamalla lähettävän järjestelmän dataa joko mahdollisesti noutamalla tarvittavaa informaatiota sekundäärisestä järjestelmästä tai vaikka lisäämällä sanoman dataan esimerkiksi integraation sisältämässä sanoman käsittelyssä siihen valmiiksi määriteltyjä vakioarvoja ennen valmiin sanoman lähetystä vastaanottavalle järjestelmälle.

3.5 Rajapinta ja API

Informaation siirto eri järjestelmien välillä on mahdollista ainoastaan, mikäli järjestelmät tarjoavat jonkinlaiset rajapinnat, joiden välityksellä järjestelmästä voidaan hakea ja jota kautta järjestelmiin voidaan syöttää informaatiota (Tähtinen 2005. 49). Nämä sovellusrajapinnat ovat kehittäjien luomia rajapintoja sovellukseen, joiden kautta saadaan yhteys eri tasoille tai palveluihin sovelluksessa (Linthicum 2001, 38). Rajapinta voi olla pelkästään datarajapinta,

jota käytetään sovelluksen sisältämän datan lukemiseen toisiin järjestelmiin. Rajapinta voi myös olla toiminnallinen rajapinta, jolloin se tarjoaa erilaisia toiminnallisuuksia, kuten laskenta-algoritmeja tai mahdollisuuden vaikuttaa järjestelmän tietoihin rajapinnan kautta. (Avoimen rajapinnan määritelmä 2014.)

API on lyhenne englanninkielisistä sanoista Application Programming Interface ja tarkoittaa ohjelmointirajapintaa. API:t on tarkkaan määriteltyjä mekanismeja, jotka on rakennettu yhdistämään jokin resurssi, kuten sovelluspalvelin, väliohjelmisto tai tietokanta (Linthicum 2001, 39).

Näitä sovellusten tarjoamia rajapintoja vasten tarvitaan jokin fyysinen siirtotie, siirtokerros. Siirtokerroksen kautta rajapinnoilta välitettyä informaatiota voidaan siirtää sovellusten välillä käyttäen eri siirtotapoja. (Tähtinen 2005. 50.)

3.6 Adapterit ja konnektorit

Rajapintakomponenteista käytetään nimityksiä konnektori tai adapteri. Konnektori tarjoaa yksinkertaisen rajapinnan jotain tiettyä integroitavan ohjelmiston tarjoamaa rajapintaa vasten. Konnektori välittää rajapinnasta lukemansa tiedoston informaation käsittely ja muuntokerrokselle, ottamatta kantaa tiedoston sisältöön. Samoin vietäessä informaatiota samaiselle integroidulle ohjelmistolle, konnektorin tehtävänä on vain kirjoittaa tiedoston integraatiossa määriteltyyn paikkaan. Adapteri puolestaan kykenee tulkitsemaan esimerkiksi integroitavan järjestelmän tietorakenteita niin, että se kykenee käymään itsenäisesti monimutkaisiakin keskusteluja integroitavan järjestelmän kanssa. Adapterille pystyy esimerkiksi toimittamaan komennon etsiä järjestelmän tietyltä osa-alueelta informaatiota, sille annetuilla parametreillä. (Tähtinen 2005, 120-121.)

3.7 Muuntimet

Mikäli integroitavat järjestelmät eivät tuota ja ymmärrä keskenään yhtäläistä informaatiota, tarvitaan kahden eri järjestelmän välille muunnin, jolla lähettävän järjestelmän data saadaan muutettua vastaanottavan järjestelmän ymmärtämään muotoon. Nämä muuntimet sijaitsevat integraation muunnoskerroksessa. Informaatio lähetetään lähettävästä järjestelmästä ja kulkee rajapinnan sekä siirtokerroksen kautta muunnoskerroksen käsiteltäväksi. Kun informaatio on muunnettu vastaanottajan ymmärtämään muotoon, se lähetetään edelleen siirtokerrosta ja vastaanottavan järjestelmän rajapintaa pitkin perille kohteeseensa. Muuntokerroksen tulee ymmärtää molempien järjestelmien tuottamaa informaatiota sekä muuntamaan toimitetusta informaatiosta vastaanottavalle järjestelmälle soveltuvaa informaatiota. Tämä edellyttää, että muuntokerros ymmärtää molempien järjestelmien informaation muodon tai tietomallin eli formaatin. (Tähtinen 2005, 57.)

Muunnoskerroksen tulee kyetä muodostamaan vastaanottavan järjestelmän ymmärtämä kokonaisuus joko kokonaisuudessaan siitä informaatiosta, jonka lähettävä järjestelmä lähettää, tai vaihtoehtoisesti muunnoskerros voi koota tämän kokonaisuuden useasta eri informaatiolähteestä. Muuntokerroksessa voidaan myös automaattisesti korjata ja täydentää läpikulkevaa informaatiota joillain muuntimeen annetuilla vakioasetuksilla. Esimerkiksi mikäli lähettävästä järjestelmästä toimitettava informaatio sisältää dataa eri mittayksiköissä kuin vastaanottava järjestelmä, voidaan tämä muuntaa muuntokerroksessa vakioasetuksiin asetetuilla muuntimilla vastaanottavan järjestelmän ymmärtämään mittayksikköön. Tai läpi kulkevaa informaatiota voidaan rikastaa muuntokerroksessa esimerkiksi jollain integraation tai vastaanottavan järjestelmän kannalta oleellisella datalla.

3.8 JSON

JSON lyhenne muodostuu englanninkielisistä sanoista JavaScript Object Notation. JSON on kevyt, tekstipohjainen tiedon siirtoon ja säilytykseen soveltuva ohjelmointikieleen sitoutumaton informaation esitystapa. JSON on johdettu ECMAScript-ohjelmointikielestä, mutta on kuitenkin riippumaton ohjelmointikielestä. JSON koostuu objekteista, jotka voivat sisältää järjestelemättömiä avain - arvo pareja sekä listoja. Objektin sisältämään avaimeen voidaan asettaa merkkijono ja arvoon voidaan sijoittaa dataa, joka on tyypiltään merkkijono, numero, boolean, null, lista tai toinen objekti. Lista on rakenteeltaan toisistaan pilkulla erotettuja arvoja. JSON syntaksi ei määrittele mitään erityistä tarkoitusta listamuodossa esitettyjen arvojen järjestykselle. (Standard ECMA-404 2017.) (Kuvio 4.)

```
{
    "family": {
        "name": "Ankka",
        "town": "Ankkalinna",
        "members": {
            "Aku": [
                "Tupu", "Hupu", "Lupu"
            ]
        }
    }
}
```

Kuvio 4: Esimerkki JSON muotoisesta informaatiosta

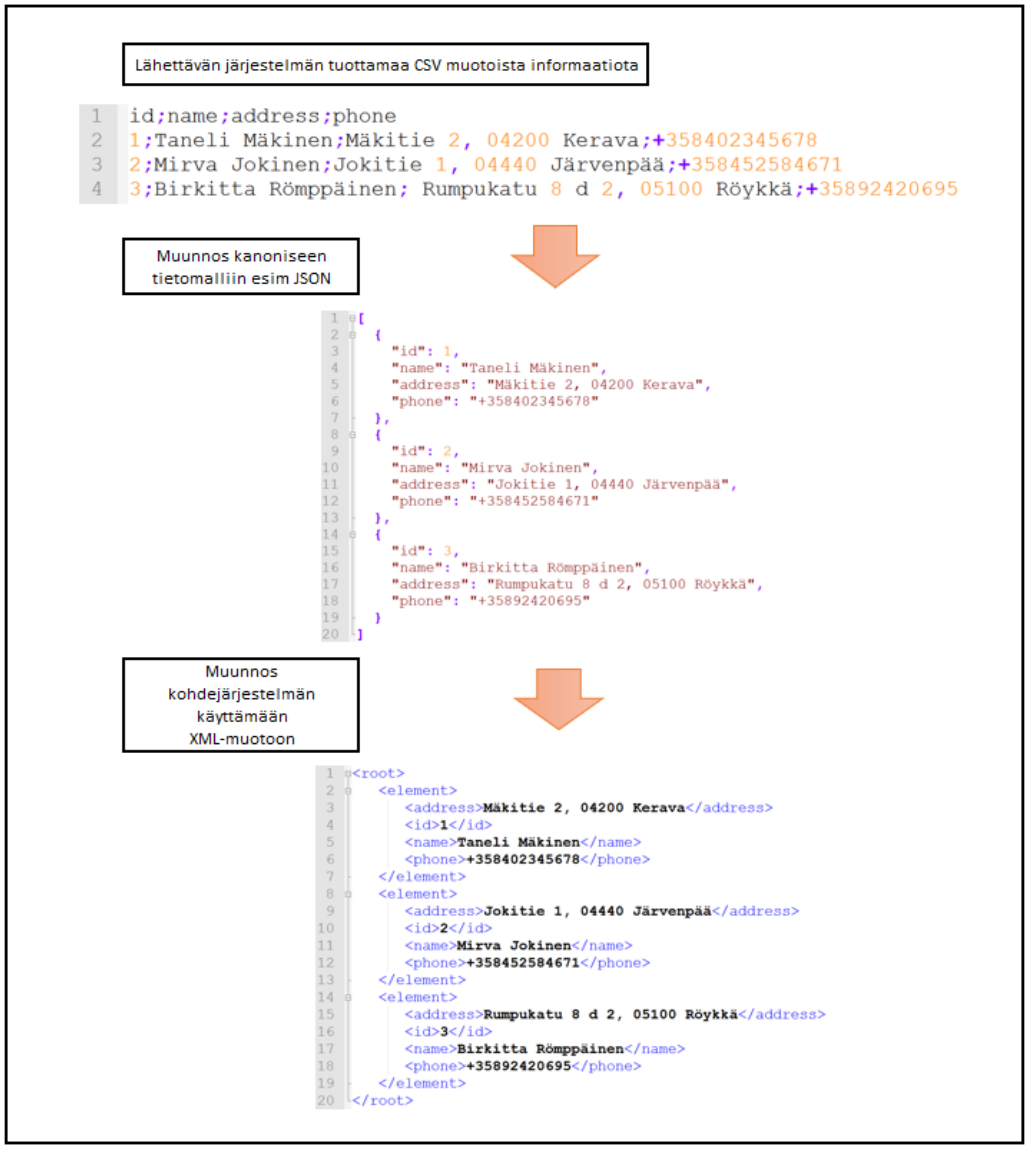
3.9 Kanoninen tietomalli

Tietomallilla tarkoitetaan datan rakenteen ja datan suoritettavan käsittelyn määrittelevää abstraktia mallia. Tietomalli järjestää dataelementit sekä standardoi niiden relaatiot toisiinsa. Tietomallilla dokumentoidaan, miten dataa säilytetään ja kuinka dataa voidaan hakea järjestelmästä. (Data Model 2018.)

Kanoninen tietomalli (canonical data model, CDM) on tietomallityyppi, joka esittää datakokonaisuudet ja yhteydet yksinkertaisimmalla mahdollisella tavalla. Kanonista tietomallia käytetään yleisimmin integroidessa kahta toisistaan poikkeavaa järjestelmää tai tietokantaa, niiden käyttämästä teknologiasta riippumattomasti. Kanoninen tietomalli tunnetaan myös yleisenä tietomallina, joka kattaa kaiken datan lähettävistä järjestelmästä vastaanottaviin järjestelmiin. (AMIS Technology Blog 2016.)

Kanoninen tietomalli ei ole yhdistelmä kaikista tietomalleista. Se on uusi tapa mallintaa toisistaan erillään olevien järjestelmien dataa. Tämän mallin täytyy kyetä kääntämään muita datan esitystyyppejä kuten kun järjestelmän tarvitsee ottaa yhteyttä toiseen, käännetään lähetettävä data ensin kanoniseen muotoon. Tämän jälkeen kanonisessa muodossa oleva informaatio käännetään vastaanottavan järjestelmän käyttämään muotoon ja lähetetään vastaanottavalle järjestelmälle käsittelyyn. (Canonical Data Model 2018.)

Esimerkkinä lähettävä järjestelmä A tuottaa CSV-muotoista informaatiota, joka muunnetaan integraatiossa noudattamaan esimerkiksi JSON-muotoista kanonista tietomallia. Tämän jälkeen informaatio tulisi lähettää vastaanottavalle järjestelmälle B, joka tukee XML-muotoista informaatiota, jolloin informaatio vaatii uuden muunnoksen ennen lähetystä järjestelmälle B. (Kuvio 5.)



Kuvio 5: Yksinkertaistettu esimerkki järjestelmien välisistä datan esitystapojen muutoksista

4 Geneeriset integraatiot integraatiokehityksessä

Integraatioiden geneerisyydellä tarkoitetaan tämän opinnäytetyön viitekehyksessä integraatiota, joka toimisi yleispätevänä ratkaisuna useamman kuin yhden asiakkuuden kahden sille integraatiolle spesifin eri järjestelmän väliseen integraatioon. Optimitilanteessa integraatio kehitettäisiin niin geneeriseksi, että se ei vaadi uuteen ympäristöön vietäessä muita muutoksia kuin sen ympäristön konfiguraatiot.

Geneeristen integraatioiden kehittämisellä pyritään saavuttamaan integraatioympäristöjen nopea käyttöönotto ja yhdenmukainen ylläpito. Valtaosa integraatioista luodaan asiakaskohtaisesti, joka saattaa hankaloittaa niiden ylläpitoa ja jatkokehitystä jokaisen ympäristön

mahdollisesti noudattaessa eri ratkaisuja sanomien välityksessä. Integraatioratkaisusta tulee ajan kanssa herkästi hankalia ylläpitää ja kehittää.

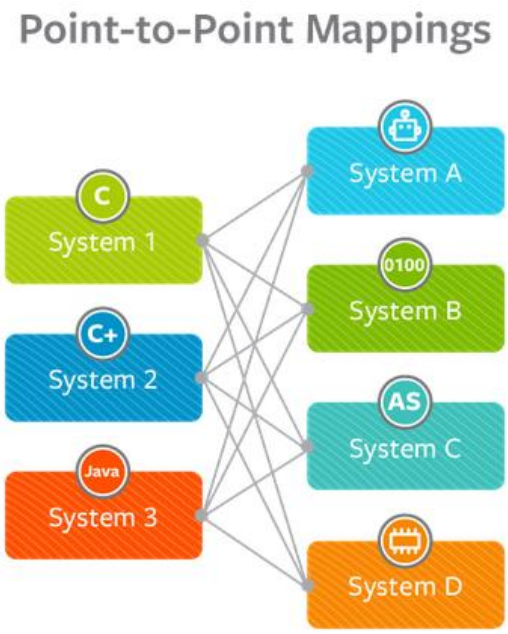
Kanoniset tietomallit ovat merkittävässä asemassa yhteiskäyttöisten integraatioiden käytössä, niiden määrittäessä sisään tulevien sanomien sisällön tietorakenteen ja niiden avulla varmistetaan, että sanomat sisältävät sanomanvälitykselle ja sanoman vastaanottajalle merkitykselliset tiedot.

4.1 Kanonisen tietomallin ja muuntimien käyttö

Suunnitellessa useiden ohjelmistojen välistä kommunikaatiota sanomavälityksen kautta, on syytä huomioida, että jokaisella ohjelmistolla on mahdollisesti oma sisäinen tietomalli. Tällöin on tarve selvittää, miten minimoidaan riippuvuussuhteiden eli dependenssien määrä, kun integroidaan ohjelmistoja, joilla on käytössä toisistaan eroavat tietomallit. Toisistaan erillään kehitetyt ohjelmistot käyttävät yleensä eri tietomalleja, sillä jokaisen ohjelmiston tietomalli on suunniteltu juuri kyseisen ohjelmiston käyttötarkoitukseen soveltuvaksi. Kun ohjelmisto suunnitellaan lähettämään tai vastaanottamaan viestejä jostain toisesta, sille tuntemattomasta ohjelmistosta, käyttää ohjelmisto luonnollisesti sille parhaiten soveltuvaa tietomallia. (Hohpe 2004, 355.)

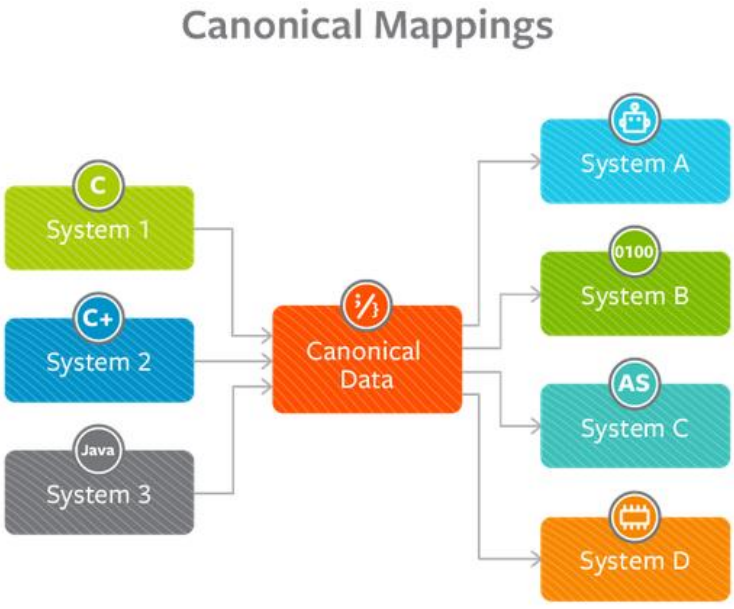
Sanomien käsittelyssä käytettävä muunnin ratkoo sanomien välisiä eroavaisuuksia tietomallissa muuttamatta itse ohjelmistoja tai paljastamatta keskenään viestivien ohjelmien käyttämiä sisäisiä tietomalleja. Kuitenkin, jos keskenään kommunikoivia ohjelmistoja on useita, saattaa jokaisen kahden keskenään keskustelevan ohjelmiston väliin olla tarve laittaa yksi ulkoinen muunnin. Tämä lähestymistapa vaatii ison määrän ulkoisia muuntimia, varsinkin jos jokainen integroitu ohjelmisto tuottaa tai vastaanottaa useita eri sanomatyyppejä. Tarvittavien ulkoisten muuntimien määrä kasvaa eksponentiaalisesti suhteessa integroituihin ohjelmistoihin, mikä taas muuttuu nopeasti hallitsemattomaksi. (Hohpe 2004, 355-356.)

Vaikka ulkoinen muunnin tarjoaa kahden eri tietomallia käyttävän järjestelmän välille yhteyden, se on silti riippuvainen eri tietomalleista, joita molemmat järjestelmät käyttävät. Tämän takia, jos point-to-point integraatioilla muihin järjestelmiin integroidun järjestelmän käyttämä tietomalli muuttuu, täytyy kaikki kyseisestä järjestelmästä muihin järjestelmiin olevien yhteyksien muuntimet muuttaa vastaamaan uutta tietomallia. Tämän lisäksi, jos järjestelmäkokonaisuuteen lisätään jokin uusi järjestelmä, jolla on oma yksilöllinen tietomallinsa, täytyy tämän uuden järjestelmän ja kaikkien siihen yhteydessä olevien välille luoda omat ulkoiset muuntimet, jotta järjestelmät voivat viestiä toistensa kanssa. Huomioitavaa on myös se, että jokainen lisävaihe muunnoksiin lisää viivettä liittymässä ja vähentää suoritustehoa (Kuvio 6). (Hohpe 2004, 356.)



Kuvio 6: Point-to-point integraatiot usean järjestelmän välillä (Chanonical Data Model, 2018).

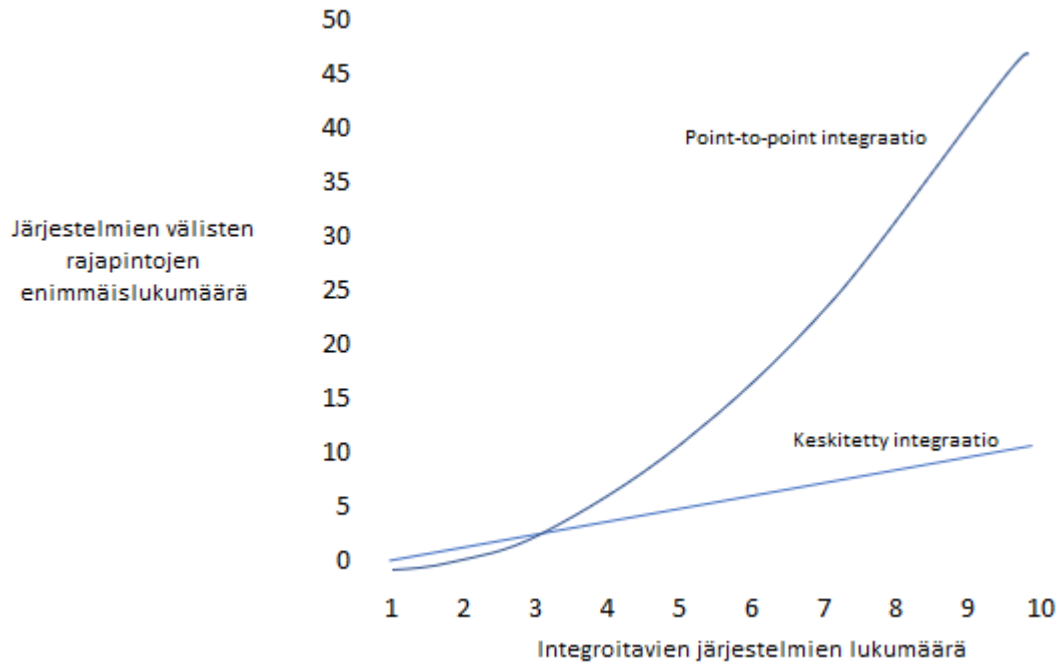
Kanoninen tietomalli tarjoaa uudentasoisen yhteyden järjestelmien yksilöllisten tietomallien välille. Mikäli järjestelmäkokonaisuuteen on tarve lisätä uusi järjestelmä, jolla on oma yksilöllinen tietomallinsa, tarvitaan ainoastaan muunnin, joka muuntaa uuden järjestelmän tietomallin kanoniseen tietomalliin, huolimatta siitä kuinka monta eri järjestelmää on osallisena (Kuvio 7). (Hohpe 2004, 356.)



Kuvio 7: Kanonisen tietomallin käyttö useamman järjestelmän integraatioissa (Chanonical Data Model, 2018).

Kanonisen tietomallin käyttö pienempien järjestelmäkokonaisuuksien integraatioissa saattaa vaikuttaa ylimitoitetulta toimenpiteeltä, mutta sen käyttö on eduksi, kun kokonaisuuteen tulee tarve lisätä järjestelmiä. Järjestelmäkokonaisuuteen liitettyjen järjestelmien käyttäessä suoria eli point-to-point integraatioita, ei tarvittavien ulkoisten muuntimien määrä kasva vielä hallitsemattomaksi, mikäli kokonaisuus sisältää vain kaksi toisiinsa liitettyä järjestelmää. Kuitenkin kun tähän kokonaisuuteen lisätään yksi uusi järjestelmä, tarvitaan muuntimia jo kuusi kappaletta.

Mikäli järjestelmäkokonaisuus koostuu esimerkiksi kuudesta erillisestä järjestelmästä, kasvaa tarvittavien järjestelmien ulkoisten muuntimien määrä kolmeenkymmeneen ilman kanonista tietomallia, kun taas kanonista tietomallia käytettäessä, ulkoisia muuntimia tarvitaan vain kaksitoista (Kuvio 8).

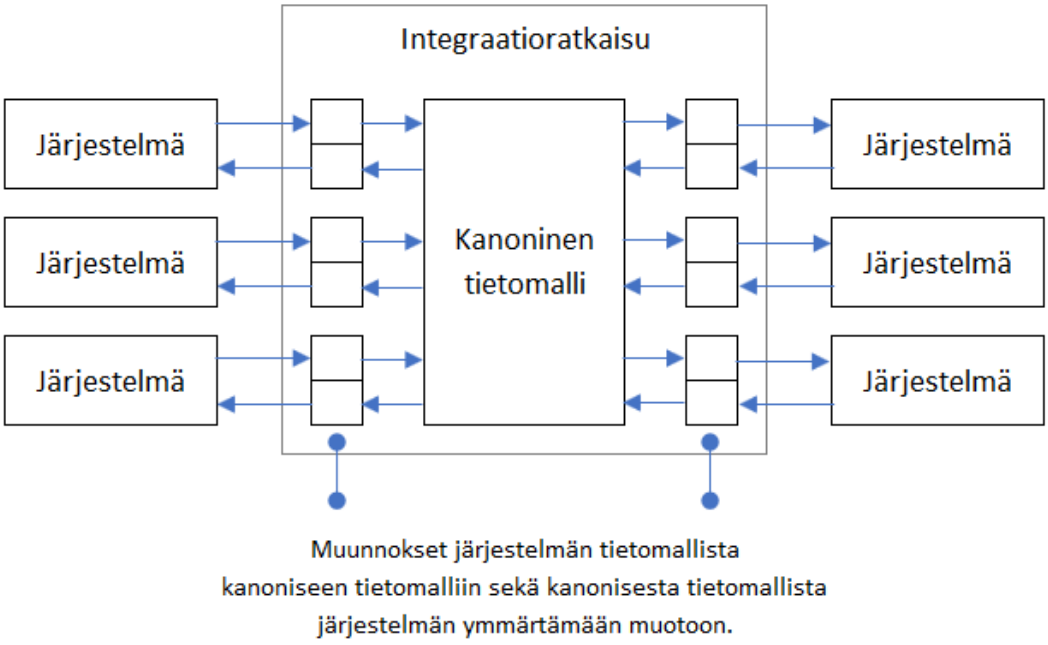


Kuvio 8: Integraatioiden enimmäismäärä point-to-point integraatioissa ja kanonista tietomallia käyttävissä keskitetyissä integraatioissa.

Kanonisen tietomallin käyttäminen järjestelmäintegraatioissa tuo etua myös tilanteeseen, jossa järjestelmäkokonaisuuden järjestelmistä joudutaan joku vaihtamaan toiseen järjestelmään. Esimerkiksi vanhemman järjestelmän tilalle vaihdetaan uusi, edelliseen verrattuna täysin erilainen järjestelmä. Tällöin integraatioista tarvitsee vain korvata vanhemman järjestelmän muunnin uudemman järjestelmän tietomallin mukaiseen muuntimeen.

Kanonisen tietomallin käyttö aiheuttaa kuitenkin tietynlaista yleisrasitetta sanomavirtaan. Jokaisen sanoman tarvitsee mennä kahden eri muunnosvaiheen läpi yhden sijasta; muunnos

lähettävän järjestelmän tietomallista kanoniseen tietomalliin sekä kanonisesta tietomallista vastaanottavan järjestelmän tietomalliin. Tästä johtuen kanonisen tietomallin käyttöä kutsutaan joskus kaksoismuunnokseksi (double translation). Suoraa muunnosta yhdestä tietomallista toiseen kutsutaan suoraksi muunnokseksi (direct translation). Jokainen erillinen muunnosvaihe sanoman käsittelyssä aiheuttaa viivettä sanomien kulussa, jolloin erittäin korkeaa suorituskykyä vaativat järjestelmät saattavat vaatia pelkästään suoria muunnoksia (Kuvio 9). (Hohpe 2004, 358.)



Kuvio 9: Kanonisen tietomallin kautta tehdyt muunnokset.

Sanomapohjaisissa integraatioissa, joissa kulkee esimerkiksi merkkipohjaista jäsenneltyä dataa järjestelmien välillä, lähtevän datan tietomallin muuttaminen kanoniseen muotoon on merkittävä yritettäessä saavuttaa nykyistä geneerisempiä integraatioratkaisuja. Muutettaessa data kanoniseen muotoon, voidaan sama data toimittaa useammalle eri vastaanottavalle järjestelmälle integraatioissa. Kanonista tietomallia noudattava data on tällöin monikäyttöistä integraatioympäristössä.

4.2 Integraatioiden geneerisyys toimeksiantajayrityksen integraatiotuotteessa

Tässä opinnäytetyössä keskitytään toimeksiantajayrityksen omaan integraatiotuotteeseen eli väliohjelmistoon ja sen ominaisuuksiin. Tämän integraatiotuotteen ollessa yrityksen omistuksessa, voidaan tuotetta muokata, parannella ja päivittää sen hetkiselle tarpeelle soveltuvammaksi. Tuotteen lähdekoodin ollessa yrityksen vapaassa käytössä, on tuotteelle kyetty tuottamaan lisäarvoa kehittämällä siihen niin kutsuttuja lisäosia.

Tässä opinnäytetyössä keskitymme niihin lisäosiin, jotka on kehitetty palvelemaan tiettyä käyttötarkoitusta, mutta ovat käyttöympäristössään geneerisiä. Näiden geneeristen integraatiomoduulien ansiosta on aikaan saatu yrityksen toiselle tuotteelle geneerinen integraatioympäristö, jonka ansiosta tämä tuote vaikuttaa yhdeltä reaaliaikaiselta kokonaisuudelta sen sijaan, että se koostuisi kolmesta irrallisesta ohjelmasta. Integraatioympäristö on myös suunniteltu siirrettäväksi sellaisenaan toiseen ympäristöön, mikäli tuote myytäisiin jollekin uudelle asiakkaalle. Tämä siirto vaatisi käytännössä vain integraatiotuotteen konfiguraatioihin uuden ympäristön yhteyksien tietojen tuonnin.

5 Tutkimusmenetelmä

Tutkimuksessa on käytetty laadullista eli kvalitatiivista tutkimusmenetelmää ja sen kohteena ollessa yksittäinen kehitysosa-alue katsotaan sen olevan tapaustutkimus. Tutkimuksessa on käytetty teoriapohjana alan kirjallisuutta ja tätä tukemaan tehtiin lyhyt puolistrukturoitu haastattelututkimus neljälle eri kokemuspohjan omaavalle integraatiokehittäjälle.

5.1 Kvalitatiivinen tutkimus

Laadullisella eli kvalitatiivisella tutkimusmenetelmällä on tavoitteena ymmärtää tutkittavan kohteen laatua, ominaisuuksia ja merkitystä kokonaisvaltaisesti. Otoksena kvalitatiivisessa tutkimuksessa on yleensä pieni määrä tapauksia, joita pyritään analysoimaan tarkasti. (Hirsjärvi 1997, 156-157.)

Kvalitatiivista tutkimusta voidaan toteuttaa usealla erilaisella menetelmällä. Yhteisenä piirteenä näissä menetelmissä korostuu näkökulmat, jotka liittyvät muun muassa tutkimuskohteen esiintymisympäristöön, taustaan, tarkoitukseen ja merkitykseen. Laadullinen tutkimus soveltuu toiminnan kehittämiseen, vaihtoehtojen etsimiseen sekä sosiaalisten ongelmien tutkimiseen. (Hirsjärvi 1997, 156-157.)

5.2 Tapaustutkimus

Tapaustutkimuksessa (case study) tutkitaan yksittäistä tapahtumaa, rajattua kokonaisuutta tai yksilöä. Olennaista tapaustutkimukselle on, että tutkittavaksi valittu kohde on yksittäinen tapaus, tilanne, tapahtuma tai joukko tapauksia, joka muodostaa tutkimuksen kohteeksi soveltuvan kokonaisuuden. (Saaranen-Kauppinen 2006.)

Tapaustutkimuksella pyritään selvittämään jotain ennalta tuntematonta, josta halutaan tarkempaa tietoa. Tapaustutkimuksen tarkastellessa usein monimutkaisia ja pitkäikäisä ilmiöitä, se soveltuu vastaamaan kysymyksiin *miten* ja *miksi*. Tutkimustavan tavoitteena on lisätä ymmärrystä tutkimuksen kohteesta. (Laine 2007, 9-12.)

Tutkimuskohteeksi valitusta tapauksesta pyritään tapaustutkimuksessa aikaansaamaan yksityiskohtaista tietoa. Tapaustutkimus ei pyri yleistettävyyteen, mutta kun sillä pyritään

ymmärtämään ja tulkitsemaan perusteellisesti yksittäisiä tapauksia niiden omassa kontekstissa. Tapaustutkimuksella haetaan tietoa ilmiöön sen kaikilta osa-alueilta niin kattavasti, että tutkimuksen tuloksilla voidaan osoittaa olevan jonkinlaista yleistettävyyttä. (Hirsjärvi 1997, 130-131.)

5.3 Haastattelututkimus

Haastattelu on yksi käytetyimmistä tiedonkeruutavoista. Haastattelutilanteessa tutkimusta tekevä ja haastateltava keskustelevat, haastattelutyypin mukaan, enemmän tai vähemmän järjestelmällisesti eli strukturoidusti tai laveasti niistä asioista, jotka kuuluvat tutkimuksen aiheeseen. Tutkimushaastattelu poikkeaa arkisesta keskustelusta tai esimerkiksi sanomalehteen tehdystä haastattelusta, sillä tutkimushaastattelulla on selkeä päämäärä, joka on tutkimustehtävän suorittaminen. Haastattelu tehdään tutkimusaineiston keräämiseksi ja aineistoa analysoidaan ja tulkitaan tieteellisen tutkimustehtävän selvittämiseksi. (Hirsjärvi 2000, 34, 42.)

Haastattelutapoja ja -tyyppejä on useita ja haastatteluja voidaan kategorisoida monin eri tavoin. Useimmiten käytetyssä luokittelutavassa haastattelun tyyppi jaotellaan sen mukaan, kuinka paljon haastattelija antaa kysymyksissään liikkumatilaa haastateltavalle. Esimerkiksi lomake- eli strukturoidussa haastattelussa haastattelun kulku on tarkoin suunniteltu ja kysymykset vastausvaihtoehtoineen ovat valmiina. Kun taas puolistrukturoiduissa ja strukturoimattomissa haastatteluissa kysymysten esittämistavat vaihtelevat eikä valmiita vastausvaihtoehtoja ole annettu. (Hirsjärvi 2000, 34, 43-44.)

Puolistrukturoitu tai puolistandardoitu haastattelu on välimuoto lomakehaastattelulle ja strukturoimattomalle haastattelulle. Tästä haastattelumuodosta ei ole yhtä selkeää määritelmää. Fieldingin (2000, 47) mukaan puolistandardoidussa haastattelussa kysymysten muoto on kaikille sama, mutta järjestys saattaa vaihdella haastattelutilanteessa. (Hirsjärvi 2000, 47.)

5.4 Tutkimuksessa käytetyt menetelmät

Tämän tutkimuksen aineistonkeruussa on käytetty puolistrukturoitua haastattelua. Aineistoa on kerätty tutkimuksen aiheen vuoksi integraatioasiantuntijoilta toimeksiantajayrityksessä muutamilla aiheeseen liittyvällä ennen haastattelutilannetta valmiiksi laaditulla kysymyksellä. Haastattelujen edetessä haastattelija on esittänyt tarpeen vaatiessa tarkentavia kysymyksiä aiheesta haastateltavalle.

Tutkimusmenetelmänä on tutkimuksen aihepiirin vuoksi käytetty laadullista eli kvalitatiivista tutkimusta. Tutkimusta tehdessä ei ole katsottu aiheelliseksi tai edes mahdolliseksi kerätä tutkimusaineisoa kvantitatiivisin menetelmin, johtuen tutkimuksen aiheesta. Aineistoa kerätessä on katsottu tarpeelliseksi haastatella sellaisia henkilöitä toimeksiantajayrityksessä,

joilla katsotaan olevan aiheeseen liittyvää riittävää asiantuntijuutta ja kokemusta integraatiokehittämisestä.

Tutkimuksen kohdistuessa toimeksiantajayrityksessä käytössä olevaan yrityksessä kehitettyyn integraatioalustaan ja mahdollisuuksiin kehittää kyseistä integraatioalustaa yrityksen omien kehittäjien toimesta, sekä tähän integraatioalustaan jo kehitettyihin ominaisuuksiin, voidaan katsoa tutkimuksen olevan case study eli tapaustutkimus.

5.5 Reliabiliteetti ja validiteetti

Tutkimuksen reliaabeliudella eli luotettavuudella tarkoitetaan saatujen mittaustulosten toistettavuutta. Mikäli tulokset eivät ole sattumanvaraisia voidaan tutkimus todeta olevan reliaabeli. Tutkimusta uusittaessa alkuperäistä vastaavissa olosuhteissa, pitäisi tulosten olla vastaavat. Tutkimustilanteessa esitettyjen kysymysten tulee olla yksiselitteisiä sekä helposti ymmärrettävissä ja haastattelujen tulee olla huolellisesti suoritettuja. (Hiltunen 2009.)

Tutkimuksen validiudella eli pätevyydellä tarkoitetaan tutkimuksessa käytetyn menetelmän kykyä mitata tutkimusta tehdessä sitä, mitä sen on tarkoituskin mitata (Hirsjärvi 1997, 226). Mikäli mittausten tulokset osoittavat saadun tiedon vastaavan teoriaa tai sillä voidaan sitä tarkentamaan ja parantamaan, voidaan tuloksen katsoa olevan validi (Hiltunen 2009).

Kaiken tutkimuksen luotettavuutta ja pätevyyttä tulisi arvioida jollakin tavoin. Laadullisessa tutkimuksessa luotettavuutta voidaan parantaa tarkalla selostuksella tutkimuksen toteutuksesta. Selostuksessa tulisi selvästi ja totuudenmukaisesti kertoa aineiston tuottamisen olosuhteet. Haastattelututkimuksessa tulisi kertoa olosuhteet ja paikat sekä haastatteluun käytetty aika, mahdolliset häiriötekijät ja virhetulkinnat haastattelutilanteessa. (Hirsjärvi 1997, 227).

Tulosten tulkinnassa tulisi kertoa millä perusteella tulkintoja esitetään ja mihin päätelmät perustetaan. Esimerkiksi haastattelututkimuksessa on hyvä käyttää suoria lainauksia tekstin seassa työhön tehtyjen päätelmien ja tulkintojen tukena. (Hirsjärvi 1997, 228.)

6 Haastattelun tulokset

Tutkimuksen teoriapohjaa tukemaan tehtiin lyhyt haastattelututkimus neljälle eri kokemustaustan omaavalle integraatiokehittäjälle jatkuvissa palveluissa. Haastattelututkimuksessa selvitettiin vastaajien oma kokemustausta integraatiokehitystyöstä ja siihen käytetyistä työkaluista sekä haastateltavien näkemyksiä integraatioiden suunnittelusta, kehityksestä ja valvonnasta (Liite 1). Haastattelu tehtiin yhdelle haastateltavalle kerrallaan. Haastatteluun osallistuneiden henkilöiden kokemustaustat vaihtelivat aloittelevasta integraatiokehittäjästä 40 vuoden ajan integraatiokehitystyötä tehneeseen. Haastattelutilanteet kestivät haastateltavasta riippuen 10 minuutista reiluun tuntiin.

Vastaajien keskuudessa oli käytetty lukuisia erilaisia tuotteita integraatioiden toteuttamiseen. Integraatiotuotteista ja niihin liitetyistä ulkoisista ohjelmistoista esiin nousivat muutamana yleisimpänä Dell Boomi, Python, Logic Apps, Friends, BizTalk, AWS sekä yrityksen oma tuote iSuite. Näiden lisäksi sanomien käsittelyssä on käytetty erilaisia jonotyökaluja ja ulkoisia ohjelmia erilaisten skriptien ajoon.

6.1 Integraatiokehityksessä ja valvonnassa esiintyneet ongelmat

Integraatiokehityksen ongelmina vastaajien keskuudessa pidettiin muun muassa eri asiakasympäristöjen keskinäisiä eroavaisuuksia ja puutteellista dokumentointia sekä integraatioista itsestään, että siitä ympäristöistä, joissa integraatiot toimivat. Yhtenä ongelmana tuli myös ilmi, että ohjelmistokehittäjillä saattaa olla hankaluuksia ymmärtää jatkuvien palveluiden näkökulmaa kehitettäessä uutta sovellusta ja sen rajapintoja.

”Mitä ongelmia itselle on tullut, liittyy siihen, että miten erilaisia kaikki ympäristöt on.” (Liite 2.)

”-- kehittäjäpuoli ei ymmärrä jatkuvien palveluiden näkökulmaa. On erikseen kehittäjät ja erikseen jatkuvat palvelut -- .” (Liite 2.)

Näissä tapauksissa rajapinnat eivät ole olleet kunnollisia tai niitä ei ole ollut ollenkaan integroitavassa tuotteessa, jolloin tämän sovelluksen integroiminen toiseen sovellukseen hankaloituu tai saattaa olla mahdotonta. Haastatteluissa tuli myös esille, että sovellusta kehitettäessä ei välttämättä pidetä tärkeänä, että olisi integraatioiden kannalta tärkeitä aineistotason rajapintoja vaan sovellukseen kehitetään mahdollisesti vain toiminnallisia- ja käyttöliittymärajapintoja.

”Saattaa olla tuote, jossa ei ole kunnollisia rajapintoja tai, jos rajapinnat on tehtynä, niin ne on vain ja ainoastaan tarkoitettu käyttöliittymää varten eli tieto johon pitäisi päästä käsiksi niin kykyä tuottaa sitä ei oikeasti ole. Eikä sitä koeta tärkeänä, että olisi aineistotason rajapintoja.” (Liite 2.)

Koettiin myös, että sovellusta kehitettäessä pyritään yleensä mahdollisimman yksinkertaisiin ratkaisuihin tehdä asiat, jolloin esimerkiksi käyttöliittymän rajapinta katsotaan ainoaksi tavaksi paljastaa sovelluksen sisältämää informaatiota sen ulkopuolelle. Integraatioiden näkökulmasta sovellusta kehitettäessä tulisi ottaa huomioon integraatioiden tarvitsemat rajapinnat ja sovelluksen kyky tuottaa aineistoa integraatioiden käsiteltäväksi.

”-- integraation näkökulmasta pitäisi olla erillinen aineistointegraatio, joka kykenee palvelemaan montaa eri tarkoitusta --.” (Liite 2.)

Tällaiset puutteet sovelluksissa ovat pakottaneet integraatiokehittäjät kehittämään integraatioihin erilaisia kiertoteitä ratkaisuiksi ja mahdollisesti jopa kehittämään ulkoisen komentosarjan eli skriptin, joka on käytännössä pieni ohjelma, jolla tämä puuttuva ominaisuus sovelluksessa korjataan.

”Tuotteiden rajoitukset, kaikki tuotteet eivät näytä tukevan kaikkia mahdollisia rajapintoja tai jos väittävätkin tukevan niin eivät toimi kuitenkaan. -- Silloin joutuu käyttämään kiertoteitä tai jopa tekemään uuden ulkoisen skriptin, joka sitten hoitaa kyseisen tehtävän.” (Liite 2.)

Integraatioiden valvonnassakin koettiin ongelmalliseksi eri ympäristöjen eroavaisuudet ja dokumentaation puutteellisuus. Integraatiovalvonnassa oli myös havaittu, että jossain asiakasympäristössä on saatettu kehittää ensimmäiset integraatiot hyvin vanhalla teknologialla, jonka päälle on myöhemmässä vaiheessa kehitetty lisää, mutta edellistä uudemmalla teknologialla. Näiden eri teknologioiden yhteensopivuus on saattanut aiheuttaa ongelmia integraatioissa ja niiden valvonnassa tapahtuneiden virheiden selvityksessä.

” -- joskus on sellaisia iänikuisen vanhoja juttuja, joiden päällä on sitten uudempia juttuja. Ei välttämättä natsaa suoraan niin hyvin yhteen ja sitten dokumentaatio toisinaan myös puutteellista sen takia, kun siellä on sitä vanhaa kamaa -- .” (Liite 2.)

Haastattelussa tuotiin myös esiin geneeristen integraatioiden valvonta ja mahdollisuudet sen helpottamiseen. Todettiin, että mikäli ympäristö olisi täysin geneerinen toteutuksiltaan, olisi sen toteutusten valvonta myös samanlainen kaikille sen eri osa-alueille ja voitaisiin tätä valvontaa myös automatisoida.

” -- sen valvonnan laadun täytyy olla samanlainen niille eri objekteille eli jos on geneerinen integraatioympäristö, niin toteutuksien tulisi olla toistensa kaltaisia, jotta niiden valvonta voidaan automatisoida – .” (Liite 2.)

Integraatioiden jatkokehityksessä ja päivittämisessä koettiin myös ongelmalliseksi kommunikaatio sovelluskehittäjien kanssa ja heidän mahdollinen ymmärtämättömyytensä muutosten vaikutuksesta integraatioihin. Esimerkkinä ongelmatilanteesta annettiin, että sovellukseen tehdään uusi kenttä, joka tulee mukaan integraatioiden käyttämään aineistoon, jolloin tältä kentältä puuttuu integraatioista mahdollisesti käsittelyt, joka aiheuttaa virheen integraatioissa.

”Jos ei ole rajapintoja, niin kehitystiimi saattaa tehdä mitä tahansa muutoksia tuotteeseen, ja jos siinä ei ole kunnon rajapintaa välissä eli kunnon API:a, kunnon välikerrosta, niin kaikki muutokset mitä he tekevät, vaikuttaa saman tien integraatioihin.” (Liite 2.)

Vaihtoehtoisesti sovellukseen lisätty kenttä on merkitty pakolliseksi, jolloin integraatioiden kyky tuoda dataa ulkoisesta järjestelmästä estyy, jos kenttää ei ole huomioitu integraatioissa tai mahdollisesti jopa ulkoisessa sovelluksessa. Hankaloittavana tekijänä katsottiin olevan myös vanhentunutta teknologiaa ja ohjelmaversioita käyttävät alustat ja ympäristöt joihin muutosta halutaan tehtävän, jolloin uuden integraation kehittämiseen tai uudelleenkonfigurointiin ei löydy enää ohjeistusta tai osaamista. Vanhemmassa teknologiassa on todettu myös olevan hankaluuksia yhteensopivuuksien kanssa uutta kehitettäessä tai vanhaa integraatiototeutusta korjatessa. Myös toisinaan vastaan tulevat vanhentuneet dokumentaatiot mainittiin ongelmien aiheuttajana.

”Valvonnassa suurimpana ongelmana on ohjeiden puute tai ohjeet ovat vanhentuneet” (Liite 2.)

6.2 Integraatioiden eri toteutustavat

Integraatioiden moninaisuuksien ymmärtämiseksi haastateltavilta kysyttiin heidän näkemystään siitä, kuinka monella tavalla voidaan yksi toteutus tehdä ja ovatko he havainneet keskenään yhtäläisiä integraatioita, joiden toteutukset ovat poikenneet toisistaan. Nähtiin, että integraatioita voidaan kehittää niin monella eri tavalla, kuin kehittäjällä mielikuvitusta riittää. Huomioitiin myös kehittäjän mahdolliset taidot eri työkalujen ja ohjelmointikielien käytössä. Työkaluja toteutuksiin katsottiin olevan tarjolla runsaasti ja ne taipuvat tarvittaessa lukemattomiin eri tapoihin.

”Sanotaan että niin monella kuin mielikuvitusta riittää. Työkalut yleensä taipuvat melkein kaikki melkein kaikkeen.” (Liite 2.)

Ainoat rajoittavat tekijät integraatioratkaisuja tehdessä ovat ympäristötekijät. Mahdollisesti asiakkaan järjestelmä ei tuota kuin XML-muotoista dataa, joka pitää muuntaa vastaanottavan järjestelmän käyttämään muotoon tai vaihtoehtoisesti järjestelmällä on täysin oma tietomalli, jolle täytyy integraatiokehittäjän kehittää täysin oma käsittely, ennen kuin sitä pystytään muuten käsittelemään integraatioissa.

”Esimerkiksi rajapinta pystyy palauttamaan yleensä sekä jsonia että XML:ää jolloin, jos integraatiokehittäja tahtoo prosessoida datan jsonina niin hän voi sen käsitellä proseduraalisesti tai Javalla helposti, mutta silloin tarvitsee osata itse koodata sinne väliin.” (Liite 2.)

Integraatiototeutusten katsottiin myös poikkeavan suuresti toisistaan, mikäli tiimin sisällä ei ole selkeitä kriteerejä integraatioiden kehitykselle (definition of done) tai ei ole tarkkaan tiimin sisällä sovittu, miten tämä kyseinen integraatiototeutus tehdään. Näiden selkeiden suuntaviivojen puuttuessa annetaan integraatiokehittäjälle vapaat kädet integraatioiden toteutuksissa.

”-- ja jos tiimillä ei ole tarkkaan määritelty tiettyjä kriteerejä, definiton of done, --- tai jos ei ole tarkkaan sovittu esimerkiksi tiimin sisällä, että miten tämä kyseinen integraatiototeutus tehdään. -- se antaa sen taiteellisen vapauden integraatiokehittäjälle.” (Liite 2.)

6.3 Nykyisten ja uusien integraatioratkaisutapojen yhtenäistäminen

Integraatiokehityksen ja uusien liittymien toteutusten yhtenäistäminen katsottiin mahdolliseksi, mutta koska integraatioiden toteutukset eivät aina ole integraatiokehittäjästä kiinni, on tämän toteuttaminen hankalaa. Yhtenä ehdotuksena oli, että integraatiotuote ohjaisi kehittäjää tekemään vain ja ainoastaan yhdellä tavalla, jolloin tehdään mahdottomaksi toimia väärin integraatioita kehitettäessä.

” Tietenkin paras tapa tähän on se, että integraatiotuote ohjaisi tekemään vain ja ainoastaan yhdellä tavalla sen asian, jolloin on mahdotonta tehdä väärin.” (Liite 2.)

Moneen taipuvan integraatioalustan huomioitiin olevan integraatiokehityksessä sekä uhka että mahdollisuus, sen palvellessa niin montaa eri käyttötarkoitusta, että sitä voidaan käyttää miten ja mihin tahansa. Tämä toisaalta myös mahdollistaa tahalliset ja tahattomat väärinkäytökset. Toisaalta integraatioiden kehityksessä on saatettu noudattaa jotain tiettyä ennalta sovittua spesifikaatiota, jonka mukaan toimitaan.

” Yleensä on noudatettu jotain speksiä, jonka mukaan toimitaan. Ja kyllä parempi olisi, että olisi tuollainen yhtenäinen käytäntö kaikessa, että sitten jos tulee jotain uutta, niin sitten tehdään siitä tällainen speksi, että se käy kaikille.” (Liite 2.)

Yhteisen linjauksen kehittämisen integraatiokehitykseen todettiin olevan hyvä idea, mutta katsottiin olevan ehkä mahdoton toteuttaa. Yhteinen linjaus helpottaisi uusien kehittäjien kehitystyön alussa ja auttaisi ongelmanselvityksessä. Mahdottomaksi tämän tehnee se, että eri aikaan kehitetyissä toisistaan erillisissä asiakasympäristöissä on jo käytetty erilaisia toimintatapoja ja suotavaa olisi käyttää sille ympäristölle ominaisia integraatiotapoja edelleen, jotta ympäristö olisi toteutuksiltaan koherentti eli yhtenäinen ja yhteistä linjaa noudattava.

”-- se olisikin suotavaa, mutta eipä taida olla mahdollista :) Asiakkaiden ympäristöt on toteutettu eri aikoina ja keskenään hyvinkin eri tavoin, joten mielestäni olisi syytä noudattaa asiakasympäristössä käytössä olevaa tapaa tehdä asioita.” (Liite 2.)

”Ainakin silloin kun aloittelee näitä konffauksia ja ensimmäisten rakentamisia niin se (geneerisyys) vois olla hyvä tapa päästä siihen juoneen käsiksi, että olisi joku sellainen pohja tai joku malli.” (Liite 2.)

6.4 Olemassa olevan integraation mallintaminen uutta kehitettäessä

Haastatteluilla haluttiin myös selvittää, että onko vastaajien keskuudessa käytetty jotain olemassa olevaa toista integraatiota mallina uutta kehitettäessä. Tähän saatiin jokaiselta vastaajalta myönteinen vastaus ja perusteluina oli kehittämisen helpottaminen ja nopeuttaminen sekä toteutusten pitäminen samassa ympäristössä yhtäläisenä ja helpommin ylläpidettävänä. On jopa tehty mallinnus itselle talteen, mikäli on löytynyt joku kehitystyön alla olevaa vastaava toteutus ja tätä mallia on käytetty mallintamaan esimerkiksi sanoman käsittelyä integraatioissa ja lähdetty tekemään uutta toteutusta noudattaen samaa mallia.

”Teen myös aika usein sitä että kun olen nähnyt jonkun vastaavan tyyppisen toteutuksen niin käyn kopiomassa siitä muistiinpanoihin templaten, että millä tavalla siellä kuljetetaan sitä sanomaa ja missä vaiheessa tehdään muunnoksia ja mitä muuta. Sitten mulla on sellainen pohja tuossa sivussa, että sitten kun alkaa itse tekemään ja testailemaan --.” (Liite 2.)

6.5 Ympäristöriippumattomat integraatiot ja niiden mahdollisuudet

Haastattelussa kysyttiin myös vastaajien näkemystä siitä, että voisiko integraatiot lähtökohtaisesti tehdä ympäristöriippumattomiksi. Vastauksista ilmeni, että tällainen kyky tuotteistaa integraatiototeutuksia on olemassa jo muun muassa DellBoomi integraatioteknologialla, joka on pilvipohjainen integraatioalusta (integration platform as a service) eli iPaas-palvelu. Teknologia tarjoaa valmiina konnektoreita yleisimpiin tuotteisiin, joiden avulla integraatioiden kehitys nopeutuu ja integraatiot ovat yhdenmukaisempia. Tämän lisäksi on olemassa myös monenlaisia integraatiototeutuksia, jotka ovat yleiskäyttöisiä. Esimerkkinä annettiin aineistopohjaiset integraatiot, joissa aineisto kertoo integraation tarvitsemat tiedot, kuten esimerkiksi viestityypin, jolla sitä tulee käsitellä ja minne se on menossa. Vaihtoehtoisesti integraatiossa on voitu käyttää mäppäystaulua, jossa määritellään aineiston perusteella sille tehtävät käsittelyt.

”Mutta on jo monenlaisia integraatiototeutuksia, jotka ovat yleiskäyttöisiä. --- On aineistopohjaisia integraatioita, jossa aineisto itse kertoo millä messagetyypillä ja kenelle se on matkalla. Tai sitten on mäppäystaulu joka määrittää sen. Mutta se on se sama integraatio, sama entrypoint joka käy hakemassa montaa eri aineistoa ja se osaa sen jälkeen reitittää ne eri paikkoihin --.” (Liite 2.)

Nämä ei kuitenkaan välttämättä täytä geneeristen integraatioiden määritelmiä, kun useat eri sanomat käyttävät samaa integraatiota ja sisääntuloväylää (entrypoint), mutta kykenee reitittämään sen oikealle vastaanottajalle, jolloin kyse ei välttämättä ole muusta kuin reitittämisestä. Näiden lisäksi nähtiin, että ainakin joitain osia integraatioista voitaisiin tehdä enemmän ympäristöriippumattomiksi erilaisten skriptien avulla.

”Kyllähän varmaan noilla skripteillä onnistuis tekemään paljonkin sellaista geneeristä kamaa mikä sopis moneen.” (Liite 2.)

Tällainen kaikille sopiva toteutus on ollut joillekin jo pitkäaikainen toive, mutta ei ole saavuttanut toteuttamiskelpoisuutta yrityksessä. Tähän syynä vaikuttaisi olevan asiakasvaatimukset ja ympäristöt.

Integraatioiden yleiskäyttöisyyden kuitenkin ymmärrettiin mahdollistavan ratkaisuja osaan nyt esiintyvistä ongelmista. Näistä mainittuna ylläpidettävyys, kustannustehokkuus, vikasietoisuus, testaamisen parantuminen sekä läpinäkyvyys. Kun samaa integraatiototeutusta käytettäisiin monessa paikassa ja on tiedossa, että se toimii kaikkialla samalla tavoin, voidaan virheen tullessa korjata sama ongelma kaikista ympäristöistä, vaikka kyseinen virhe ei olisi muualla vielä ilmennytkään.

”ylläpidettävyyden parantaminen, yhden yksittäisen integraatiototeutuksen kustannustehokkuuden parantaminen, vikasietoisuus, testaamisen parantaminen, läpinäkyvyys, eli sama integraatio toteutus on useassa eri paikassa ja tiedetään että se toimii kaikissa paikoissa samalla tavalla ja jos se menee yhdellä asiakkaalla virheeseen niin sama virhe voidaan korjata kaikilta asiakkailta, vaikka se ongelma ei ole tullut niillä vastaan vielä.” (Liite 2.)

”No ainakin sen, että meidän tekemiset vähenisi. Keventäisi sillä tavalla, ettei tarvittaisi työvoimaakaan niin paljon, jos olisi geneerinen tieto.” (Liite 2.)

”Ainakin nopeuttaa toteutusta yksinkertaisissa tapauksissa.” (Liite 2.)

”Ainakin se säästäisi aikaa konffailulta. Sitä olisi varmaan helpompi myydä asiakkaalle sitä tuotetta, jos se olisi mahdollisimman helppokäyttöinen ja siellä olisi valmista kalustoa enemmän. Ja sitten sitä selvityspuolta kanssa, että jos jotain tapahtuu, niin olisi jossain kuvauksessa, että mitä tuommoinen geneerinen skripti tekee siellä niin se ongelman selvittely olisi helpompaa.” (Liite 2.)

Kuitenkin haastatteluissa ilmeni myös näkemys, että uusien ohjelmistokielien, käytäntöjen ja sovellusten myötä integraatiokehitys on mennyt geneerisyydestä kauemmaksi. Haastateltavissa heräsi myös pohdintaa geneeristen integraatioiden kehittämisen esteistä tai ongelmista, joita voisi tulla vastaan, kun käsiteltävänä on erilaisia tietomalleja. Näiden käsittely vaatisi muuntimen, joka osaisi tulkita kaikkia aineistoja ja muotoilemaan niistä jonkin kanonisen tietomallin integraation käsittelyyn. Haastattelussa tuotiin myös esille se seikka, että on huomioitava ero integraatiossa, jota voidaan monistaa muihin ympäristöihin, joissa se kuitenkin toimii samalla lailla kussakin ja sellaisessa integraatiossa, joka pystyy käsittelemään useampaa kuin yhtä aineisto- tai sanomatyyppiä yhdessä ympäristössä.

7 Johtopäätökset

Tutkimuksen haastatteluja tehdessä selvisi, että geneerisiä integraatioita on kautta aikain yritetty kehittää, mutta lukuisten eri tietomallien käyttö eri järjestelmissä hankaloittaa täysin geneerisen integraation kehitystä. Integraatioihin pystytään kyllä kehittämään osittain geneerisiä käsittelyjä kuten, jos asiakkaalla on yksi järjestelmä, johon tarvitsee tuoda dataa useammasta muusta eri järjestelmästä tai sovelluksesta, voidaan integraatio kehittää käyttämään esimerkiksi ennalta suunniteltua kanonista tietomallia. Tämän tietomallin sisältämä informaatio voidaan muuntaa järjestelemällä se kohdejärjestelmän tietomallin mukaiseksi käyttäen vain yhtä muunninta.

Kuitenkin ongelman saattaa muodostaa usean lähettävän järjestelmän käyttämät mahdollisesti lukuiset eri tietomallit. Jokaiselle erilaiselle tietomallille pitää olla erillinen käsittely, jotta ne saadaan muunnettua tähän kanoniseen tietomalliin, jota voidaan integraatiossa käsitellä. Tässä suhteessa integraatioiden kehittäminen ei tule loppumaan, sillä erilaisia järjestelmiä kehitetään jatkuvasti lisää ja nämä järjestelmät saattavat jopa käyttää niille dedikoituja tietomalleja. Tässä tapauksessa integraatioin tarvitaan täysin uusia muuntimia. Kuitenkin haastatteluissa esille tulleiden iPaas-integraatioalustoihin kehitettyjä valmiita tuotteistettuja integraatiota voitaisiin hyödyntää integraatiokehityksessä, mikäli vastaavien palveluiden käyttö on asiakkuudessa mahdollista.

Vastaavaan kehitykseen yrityksen oman tuotteen kanssa voitaisiin yrittää pyrkiä, jotta integraatioiden kehittäminen saataisiin kustannustehokkaammaksi ja valvontaa helpotettaisiin. Näitä valmiita tuotteistettuja integraatioita voitaisiin esimerkiksi sopia ja kehittää tiimin sisäisesti, mikäli tiedetään, että tietyn tyyppisiä integraatioita tehdään paljon tai integraatioita tehdään usein johonkin tiettyyn järjestelmään. Tällöin uuden integraatiokehitystyön tullessa jäljellä olisi vain lähettävän järjestelmän liittäminen integraatioon, aineistolle muunnoksen teko ja mahdolliset asiakas- tai ympäristöriippuvaiset konfiguraatiot tähän valmiiksi kehitettyyn integraatioon.

Opinnäytetyötä tehdessä yrityksessä oli käytössä lukuisia eri integraatiotyökaluja ja useita tiimejä, joissa näitä integraatiotyökaluja saatettiin samassa projekteissa käyttää useampaa samanaikaisesti. Tämäkin tuo omat haasteensa integraatioiden geneerisyydelle. Joissain tapauksissa myös asiakkaalla on täysin eri integraatiotuote tilattuna toiselta toimittajalta ja asiakas tilaa tähän integraatiotuotteeseen tietovirran toisesta järjestelmästä, jolloin integraatioita kehitetään kahden integraatiotuotteen sekä integraatiotuotteen ja järjestelmän välille.

Myös jo olemassa olevien asiakasympäristöjen integraatiot on syytä ottaa huomioon. Kuten haastatteluissa ilmeni, lienee hyvän tavan mukaista kehittää valmiisiin ympäristöihin uudet integraatiot noudattaen samaa mallia, kuin siellä olevat, jolloin ympäristö pysyy koherenttina. Näiden ympäristöjen rakentaminen kokonaan uudestaan ei taas ole liiketoiminnan

kannalta järkevää, jos ympäristö on sellaisenaan nyt toimiva eikä asiakas tilaa muutosta tai muutosten teko ei ole perusteltua.

Näiden lisäksi isona tekijänä integraatiokehityksessä haastattelujen mukaan olivat asiakasvaatimukset. Asiakkaan tarjoamat ympäristöt ja tilaamat liittymät saattavat rajoittaa integraatiokehitystyötä ja pakottamaan noudattamaan tiettyä kaavaa. Asiakkaalla saattaa myös olla näkemys integraation toimintatavasta tai muista integraatioiden kehitykseen vaikuttavista asioista ja näihin kehittäjällä ei välttämättä ole mahdollisuuksia vaikuttaa.

Lähtökohtaisesti idea geneeristen integraatioiden kehityksestä vaikutti haastateltavien mielipiteissä silti pääasiallisesti hyvältä ja tähän kannattaisi jo kustannustehokkuuden vuoksi pyrkiä. Alustavasti voitaisiin yrittää uusia integraatioita kehitettäessä koittaa pohtia, pystyykö sen kyseisen integraation joiltain osin tekemään geneeriseksi. Mikäli näin on, näistä integraatioista tai niiden osasista voitaisiin kerätä yhtiön sisäiseen käyttöön kirjasto, josta tarvittaessa voidaan etsiä valmiita integraatioita tai aineiston käsittelyjä seuraavaan kehityskohteeseen. Tässä kirjastossa tulisi kuitenkin ottaa tarkkaan huomioon, ettei siellä ole tarjolla mitään asiakasspesifistä informaatiota vaan vain geneerisiä integraatiovälineitä.

8 Yhteenveto

Tutkimuksella oli tarkoitus selvittää geneeristen integraatioiden käyttö- ja kehittämismahdollisuuksia toimeksiantajayrityksessä. Työllä pyrittiin selvittämään näiden ympäristöriippumattomien integraatioiden hyödyt sekä mahdolliset ongelmat yrityksen integraatiokehitystyössä sekä myöhemmin tehtävässä valvonnassa ja virheenkorjaustilanteissa.

Tutkimuksen teoreettisena pohjana käytettiin saatavilla olevaa alan kirjallisuutta sekä luotettavaksi katsottuja internetsivustoja. Teoriaa tukemaan tehtiin lyhyt puolistrukturoitu haastattelu yrityksen neljälle eri osaamistason omaavalle integraatiokehittäjälle. Haastateltavia valitessa yhteisenä tekijänä oli tiimi sekä heidän päivittäisessä käytössänsä oleva yrityksen sisällä kehitetty integraatiotyökalu.

Haastattelujen tuloksena saatiin runsaasti kehittäjien omia näkökulmia tämän hetkisestä tilanteesta integraatiokehityksessä sekä heidän näkemyksiään geneerisistä integraatioista. Haastattelujen litterointeja ei niiden sisällön vuoksi voida järkevästi julkaista työn liitteenä. Näistä kuitenkin poimittiin tutkimuksen kannalta oleellisimmat kohdat, kuitenkaan työn eettisyyttä vaarantamatta.

Kehitysehdotuksena tutkimuksen pohjalta voitaisiin ajatella mallia, jossa jatkuvien palveluiden sisäiseen käyttöön kehitettäisiin esimerkiksi kirjasto näistä geneerisempään käyttöön soveltuvista erilaisista integraatio moduuleista ja muuntimista sekä niiden käyttöön liittyvät tarkat dokumentaatiot. Näiden lisäksi kehittäjät voisivat uusien asiakkuuksien kohdalla

suunnitella integraatioita yhdessä ja pohtia samalla mahdollisuuksia kehittää uusiin ympäristöihin heti geneerisempiä integraatioratkaisuja.

Tutkimuksen aihepiiri on hyvin monitahoinen ja hankala. Tutkimus on tehty hyvin korkealta sen aihepiiriä katsoen ja tutkimuksen luotettavuuteen vaikuttaa työyhteisö, jolle ominaisia työskentelytapoja ajatellen tutkimus on tehty. Tutkimus sisältää yleistä tietoa integraatiokehitysmaailmasta, mutta myös vain yksittäisen yrityksen sisäisen tiimin kokemuspohjalta näkemyksiä geneerisistä integraatioista ja niiden tuomista mahdollisuuksista. Tutkimus on toteutettu vain toimeksiantajayrityksen käyttöä varten ja heidän tarpeisiinsa.

9 Oman oppimisen arviointi

Opinnäytetyön tekeminen auttoi ymmärtämään integraatioarkkitehtuureja paremmin sekä havainnoimaan mahdollisia keinoja, joilla uusia integraatioita kehitettäessä voitaisiin näitä pyrkiä heti suunnittelemaan käyttämään jotain geneeristä moduulia tai muunninta. joka voitaisiin mahdollisesti myös implementoida toiseen täysin eroavaan ympäristöön. Havaitsin myös, että integraatioita voidaan konfiguroida lukuisilla eri tavoilla. Tavat riippuvat täysin kehittäjän osaamisesta sekä käytettävissä olevista työkaluista ja ympäristövaatimuksista.

Omassa ammatissani tutkimustyö tulee tukemaan tulevia integraatiokehitystöitä. Tutkimuksessa ilmenneet nykyiset ongelmat auttavat välttämään vääriä tai haitallisia toimintatapoja kehitystyössä ja pyrkimään kohti integraatioiden geneerisyyttä riippumatta asiakasympäristöstä. Sain myös haastatteluja tehdessäni vahvistuksen käsitykselle geneeristen integraatioiden hyödyille, mutta myös selkeitä ohjeita asioista mitä tulee välttää tämän kaltaisessa kehitystyössä.

Lähteet

Painetut

Bloomfield, B. Coombs, R. Knights, D. Littler, D. 2000. Information Technology and Organizations: Strategies, Networks, and Integration. New York: Oxford University Press.

Fielding, N. Qualitative interviewing. 1992. Teoksessa Hirsjärvi, S. Hurme, H. Tutkimushaastattelu: teemahaastattelun teoria ja käytäntö. 2000. Helsinki: Yliopistopaino.

Fong, J. 2015. Information systems reengineering, integration and normalization. 3. painos. Springer

Hirsjärvi, S. Hurme, H. Tutkimushaastattelu: teemahaastattelun teoria ja käytäntö. 2000. Helsinki: Yliopistopaino.

Hirsjärvi, S. Remes, P. Sajavaara, P. 1997. Tutki ja kirjoita. 11. painos. Jyväskylä: Gummerus Kirjapaino Oy.

Hohpe, G. Woolf, B. 2004. Enterprise Integration Patterns: Designing, Building and Deploying Message Solutions. United States: Addison Wesley.

Laine, M. Bamberg, J. Jokinen, P. 2007. Tapaustutkimuksen taito. Gaudeamus, Helsinki University Press. Helsinki.

Linthicum, D. 2001. B2B Application Integration, e-Business-Enable Your Enterprise. 2. painos. Addison-Wesley.

Linthicum, D. 2000. Enterprise application integration. 3. painos. Addison-Wesley.

Robertson, M. 2007. Seven steps to ICT integration. ACER Press.

Schmutz, G. Liebhart, D. Welkenbach, P. 2010. Service-Oriented Architecture: An Integration Blueprint. Birmingham: Packt Publishing.

Tähtinen, S. 2005. Järjestelmäintegraatio: Tarve, vaihtoehdot, toteutus. Helsinki: Talentum.

Sähköiset

Enterprise Integration Patterns. 2017. Viitattu 20.11.2018.
https://www.enterpriseintegrationpatterns.com/patterns/messaging/

Canonical Data Model. 2018. Viitattu 20.11.2018.
https://www.bmc.com/blogs/canonical-data-model/

AMIS Technology Blog - Benefits of a Canonical Data Model (CDM) in a SOA environment. 2016. Viitattu 15.4.2019
https://technology.amis.nl/2016/08/08/soa-benefits-of-a-canonical-data-model/

Avoimen rajapinnan määritelmä. 2014. Viitattu 18.10.2018.
http://avoinrajapinta.fi/

Hiltunen, L. 2009. Validiteetti ja reliabiliteetti. Jyväskylän yliopisto. Graduryhmä 18.2.2009. Viitattu 18.11.2019.
http://www.mit.jyu.fi/OPE/kurssit/Graduryhma/PDFt/validius_ja_reliabiliteetti.pdf

Standard ECMA-404 - The JSON Data Interchange Syntax. 2017. Viitattu 27.11.2018
http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

Data Model. Techopedia. 2018. Viitattu 27.11.2018.
https://www.techopedia.com/definition/18702/data-model

A generic integration architecture for cooperative information systems. 1996. Viitattu 3.4.2019.
https://www.academia.edu/19478448/A_generic_integration_architecture_for_cooperative_information_systems

Kuviot

Liitteet

Liite 1: Haastattelujen kysymykset

1. Mitä erilaisia ohjelmia/sovelluksia olet käyttänyt integraatiototeutuksissa?
2. Mitä suurimpia/yleisimpiä asiakkaasta riippumattomia ongelmia olet havainnut
    a) integraatiokehityksessä
    b) integraatioiden valvonnassa
    c) integraatioiden jatkokehityksessä/päivittämisessä
3. Oletko tavannut työssäsi integraatioita, joiden toimintaperiaatteet ovat keskenään yhtäläiset, mutta toteutukset poikkeavat selkeästi toisistaan?
4. Oletko koskaan uutta integraatiota tehdessä käyttänyt jotain aiempaa toteutusta mallina? Miksi?
5. Olisiko integraatiokehitystä ja uusien liittymien toteutusta mahdollista yhtenäistää? Esimerkiksi tiimin sisäisesti sovitaan tietynlaisille liittymille tietty toteutustapa?
6. Voisiko integraatiototeutukset tehdä yleiskäyttöisiksi?
7. Mitä ongelmia integraatiototeutusten yleiskäyttöisyydellä voitaisiin ratkaista?
8. Muita aiheeseen liittyviä kommentteja?

Liite 2: Haastattelujen litterat

Haastattelujen litterointeja ei voida sisältönsä vuoksi julkaista järkevästi opinnäytetyön liitteenä.