jamk

Integraatioliittymien siirto Digia iSuite 6:een

Matti Leppäkorpi

Opinnäytetyö, AMK
Huhtikuu 2023
Tieto- ja viestintätekniikka

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Leppäkorpi, Matti

Integraatioliittymien siirto Digia iSuite 6:een

Jyväskylä: Jyväskylän ammattikorkeakoulu. Huhtikuu 2023, 54 sivua.

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Opinnäytetyössä oli tehtävänä siirtää tilaajan, Digia Oyj:n, Integraatio ja API-liiketoiminnan asiakkaan integraatioliittymät integraatioalusta iSuite 5:ltä uudemmalle iSuite 6:lle, jotta vanhenevan iSuite 5:n käytöstä voitaisiin luopua. Liittymät tuli siirtää asiakkaan liiketoiminnan siitä kärsimättä ilman sanomaliikenteen keskeytyksiä testaamalla ne ensin testiympäristössä ja siirtämällä testien jälkeen myös tuotannon integraatioliittymät. Liittymien tietokanta oli jo valmiiksi kopioitu vanhemmalta iSuitelta uudemmalle, joten pääosin työ oli konfiguraatioiden tarkastamista ja korjaamista sekä liittymien testaamista ennen siirtoa. Lisäksi liittymät oli dokumentoitava huolella liittymistä vastaavan Ratkaisutuki-tiimin käyttöön.

Ennen varsinaisen siirron aloittamista siirrettävät liittymät kartoitettiin huolella. Liittymien siirto tehtiin liittymä kerrallaan, ja kaikki liittymien konfiguraatiot ja sanomien transformaatiot tarkastettiin ja testattiin ennen varsinaista siirtoa. Myös uusia liittymiä luotiin jakamalla yksi useamman sanomatyypin sisältävä liittymä useaan osaan. Työn eteneminen kirjattiin Jira-tiketin mikrotehtäviin sitä mukaa, kun liittymiä saatiin siirrettyä. Samalla pidettiin kirjaa siirrettyjen liittymien konfiguraatioista sekä transformaatioista ja siirron valmistuttua luotiin dokumentaatio siirretyistä liittymistä sovittuun paikkaan.

Kaikki liittymät saatiin onnistuneesti siirrettyä ilman suurempia ongelmia ja dokumentaatio ajan tasalle asiakkaan liiketoiminnan siitä kärsimättä. Ongelmia ilmaantui vain muutamien liittymien konfiguroinneissa sekä osassa sanomien transformaatioita. Dokumentaatio parantui aiempaan verrattuna siten, että liittymien konfiguraatiotiedot saatiin yhteneviksi ja helpommin saataville. Lisäksi saatiin hyödyllistä tietoa liittymien siirtämistä hankaloittavista tekijöistä ja kannan kopioinnin ongelmista.

Työn tuloksena Digian sanomavälityskeskus iCare voi luopua vanhemman iSuiten käytöstä kokonaan. Lisäksi mahdollinen seuraava vastaava liittymien siirto on saatujen tietojen avulla helpompi toteuttaa. Raportissa käydään läpi myös tietojärjestelmän vanhenemiseen liittyviä syitä ja riskejä sekä järjestelmäintegraatioiden perusteita.

Avainsanat (asiasanat)

järjestelmäintegraatio, tietojärjestelmän vanheneminen, migraatio, Digia iSuite

jamk

Description

Leppäkorpi, Matti

Migration of integrations to Digia iSuite 6

Jyväskylä: JAMK University of Applied Sciences, April 2023, 54 pages

Degree Programme in Information and Communications Technology. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

The assignment, given by Digia Oyj, was to migrate business-to-business integrations made with integration configuration platform Digia iSuite 5 to newer Digia iSuite 6. A customer of Digia's Integration and API-business has integrations running in Digia iCare service transmitting for example orders and invoices to partner businesses. The purpose was to move all existing integrations from iSuite 5 so it could be run down. All the integrations were to be tested and the migration to be done without any disruption to the customer. The database of the integrations was already copied to iSuite 6, so the work was mainly inspecting and fixing the configurations and testing the message flows and transformations. All the integrations were documented to be maintained by Integration Solution Support Team.

Before the migration work all the integrations were examined to make the work and the documentation easier. The migration was done one integration at time and all the configurations and message transformations were tested in the test environment before the migration to the production environment. Also, some new integrations were developed as one iSuite entrypoint hosted many connections and was split. The work progress was kept up to date in JIRA-ticket and it's microtasks. After all the integrations were migrated the documentation was finalized.

The migration of the integrations was done with no notable problems and the documentation updated without disruptions to the customer. Couple of message transformations didn't work right away but they were fixed. Also, some minor errors occurred in the configurations which were all re-configurated successfully. The documentation of the integrations is now easily available and a good model for documenting equivalent integrations in the future.

As a result, iSuite 5 can be run down in Digia iCare, after which only iSuite 6 is to be maintained. The work done and the documentation is a good model for a possible similar migration work of integrations in iSuite in the future. The few problems faced are now known and easier to deal with or avoid.

Keywords/tags (subjects)

enterprise application integration, software obsolescence, migration, Digia iSuite

Sisältö

1 Johdanto 3
2 Työn lähtökohdat 4
2.1 Työn tavoite 4
2.2 Kehityshaasteet 4
3 Vanhenevan tietojärjestelmän ongelmat 6
4 Järjestelmäintegraatiot 9
4.1 Järjestelmäintegraation perusteita 9
4.1.1 Järjestelmäintegraation kuvaus 9
4.1.2 Järjestelmäintegraation hyödyt 11
4.1.3 Järjestelmäintegraation käsitteitä 11
4.1.4 Integraatioprosessi 15
4.2 Järjestelmäintegraation toteutustapoja 17
4.2.1 Point-to-point integraatiot 17
4.2.2 Keskitetty integraatio 19
4.2.3 Palvelupohjainen arkkitehtuuri 21
4.2.4 Modernit integraatiot 24
4.2.5 Integraatioiden suunnittelumallit 25
4.3 Informaation eri muodot 26
4.4 Järjestelmäintegraatioiden testaaminen 28
4.5 Yleisimpiä integraatioalustoja 30
5 Integraatioalusta Digia iSuite 33
5.1 Digia iSuiten kuvaus 33
5.2 Digia iSuiten toimintaperiaate 35
5.3 Digia iSuite 5 ja Digia iSuite 6 37
6 Toteutus 40
6.1 Lähtötilanne 40
6.2 Liittymien kartoitus 41
6.3 Liittymien siirto ja toiminnallinen testaus 43
6.4 Liittymien dokumentaatio 46
6.5 Työn tulos 47
7 Pohdinta 47
Lähteet 51

Kuviot

Kuvio 1. Yritysten välinen viestien vaihto (Manouvrier & Menard, 2008, 112). 10
Kuvio 2. Integraation malli Tähtisen mukaan (Tähtinen 2005, 73). 12
Kuvio 3. Synkronisen ja asynkronisen tiedonsiirron ero (Hohpe & Woolf 2003a). 13
Kuvio 4. Sanomanvälitys (Hohpe & Woolf 2003a) 14
Kuvio 5. Integraation peruselementit (Hohpe & Woolf 2003b) 15
Kuvio 6. iSuite 6:n message log (Hirvonen 2021) 16
Kuvio 7. Point-to-point integraatio (Hogg 2008) 18
Kuvio 8. Liittymien määrän kasvu eri integraatiomalleilla (Tähtinen 2005, 67, muokattu). 19
Kuvio 9. Hub-and-spoke arkkitehtuuri (Manouvrier & Menard 2008, 96) 20
Kuvio 10. Snowflake-arkkitehtuuri Manouvrierin ja Menardin (2008, 98) mukaan. 21
Kuvio 11. ESB (Toivanen 2022) 23
Kuvio 12. Integraatioarkkitehtuurien vertailu (Hogg, 2008). 24
Kuvio 13. JSON-datan muuttaminen objektiksi MuleSoftissa. 27
Kuvio 14. MuleSoftin AnypointStudion flow-näkymä. 31
Kuvio 15. Azuren Standard LogicApps ja Consumption LogicApps. 32
Kuvio 16. iSuite 6:n käyttöliittymän aloitusnäkymä. 33
Kuvio 17. iSuite 6:n monitorointinäkymä (Hirvonen 2021). 34
Kuvio 18. Digia iSuiten referenssiarkkitehtuuri (TechIntro - Digia iSuite 2023). 35
Kuvio 19. Sanoman kulku iSuitella (Hirvonen 2021). 36
Kuvio 20. iSuiten toimintaperiaate (TechIntro - Digia iSuite 2023). 37
Kuvio 21. iSuite 5:n messages-näkymä (Hirvonen 2021). 38
Kuvio 22. Isuite 6:n entrypointin general-välilehti (Hirvonen 2021). 39
Kuvio 23. iSuite 5:n entrypointin general-välilehti (Hirvonen 2021). 39
Kuvio 24. Message Script iSuite 6:ssa (Hirvonen 2021). 43

Taulukot

Taulukko 1: Toiminnalliset vaatimukset 5

1 Johdanto

IT-alalla integraatiot ovat lähes aina olleet tärkeä osa projekteja. Nykyään pilvipalveluiden, sosiaalisen median ja älypuhelimien aikakaudella integraatiot ovat tärkeämpiä kuin koskaan (Heritage 2014). Integraatioiden avulla siirretään tietoa tietojärjestelmästä toiseen (Toivanen, 2022). Tutkimusyhtiö Gartner esittää, että jopa 50 % digitalisaatiosta tehdään nykyään integraatioiden avulla (50 % digitalisaatiosta hoidetaan yllättävällä tavalla, integraatioilla 2021). Vuodesta 2020 vuoteen 2021 integraatioliiketoiminnan markkinat kasvoivat 15,5 % ollen 25 miljardia dollaria maailmanlaajuisesti. iPaas-palvelut (integration platform as a service) olivat nopeimmin kasvava segmentti. (Biscotti, Menon, Mehta & Chibber 2022.) Annenkon (2022) mukaan integraatiot ja niiden puute tai tehottomuus ovat myös yksi IT-alan suurimmista haasteista.

Opinnäytetyön tehtävänä oli siirtää Digia Oyj:n Integraatio & API-liiketoiminnan asiakkaan (jatkossa Asiakas) integraatioliittymiä Digian sanomavälityskeskus iCaren palvelimella sijaitsevalta integraatioalusta Digia iSuite 5:ltä uudemmalle Digia iSuite 6: lle. Digia iSuite (jatkossa iSuite) on Digia Oyj:n oma integraatiotuote, joka on laajasti käytössä Digian integraatioasiakkuuksissa. Siirrettävät liittymät ovat kumppaniliittymiä Asiakkaan ja sen yhteistyökumppaniyritysten välillä, ja niillä kulkee esimerkiksi tilauksia, laskutusta ja erilaisia aineistosanomia. ICaren ja sillä olevien integraatioiden toiminnasta huolehtii Integraatio ja API-liiketoiminnan alainen ratkaisutuki-tiimi, joka valvoo integraatioita ja tekee niihin liittyen erilaisia pienkehitystöitä. Asiakkaan liittymien siirto pois iSuite 5:ltä oli ratkaisutuen tarpeista lähtöisin oleva projekti, jonka tavoitteena oli luopua vanhenevan iSuite 5:n käytöstä iCarella.

Toimeksiantaja Digia Oyj on juuriltaan suomalainen IT-alan pörssiyhtiö, jossa on työntekijöitä yli 1400. Liikevaihto vuonna 2022 oli 170,8 miljoonaa euroa. Toimipisteitä Digialla on Jyväskylän lisäksi Helsingissä, Tampereella, Joensuussa, Turussa, Oulussa, Raumalla, Vaasassa ja Lahdessa sekä tytäyhtiöt Ruotsissa ja Alankomaissa. (Digia yrityksenä n.d.) Digian palvelualueet ovat nimiltään Digital Solutions, Business Platforms, Financial Platforms sekä Managed Solutions, johon kuuluu myös Integraatio & API. Hyvässä kasvussa olevassa integraatioliiketoiminnassa Digia on pohjoismaiden suurimpia ja Euroopan mittakaavallakin merkittävä tekijä.

2 Työn lähtökohdat

2.1 Työn tavoite

Työn tavoittena oli, että iSuite 5:n käytöstä Digian iCaressa voitaisiin luopua, koska Asiakkaan liittymät olivat viimeiset aktiiviset liittymät iSuite 5:llä muutamien Digian omaan käyttöön tarkoitettujen tilastoliittymien lisäksi. Asiakkaan kumppaniliittymistä suurin osa oli tehty iSuite 5:llä aikana ennen iSuite 6:ta. Lisäksi Asiakkaalla oli muutamia uudempia liittymiä, jotka oli tehty iSuite 6:lla, joten niitä ei tarvinnut huomioida. Liittymien siirron eli migraation jälkeen kaikkien iCaren integraatioliittymien oli tarkoitus olla iSuite 6:lla.

iSuite 5:stä luopumisen syitä olivat etenkin iSuite 6:n parantuneet tietoturvaominaisuudet sekä iSuite 5:n riippuvuus vanhenevista teknologioista, joilta on tuki loppumassa tai jo loppunut. Lisäksi iSuite 5:n tuki loppuu tulevaisuudessa, ja myös joitain sen tarvitsemia lisenssejä oli vanhentumassa. Siirtämällä liittymät iSuite 6:een voitaisiin asiakkaalle tarjota ajantasaista tietoturvaa, parantunutta vakautta ja suorituskykyä, sekä helpottaa integraatioiden valvontaa.

Työn teki haastavaksi se, että kaikki liittymät olivat tuotantokäytössä, ja Asiakkaan sanomaliikenteelle ei saanut aiheutua häiriöitä. Digialla on testiympäristöt liittymien testaamista varten, mutta lopullisen varmuuden liittymän onnistuneesta siirrosta sai vasta tuotannon sanomien mennessä onnistuneesti läpi. Siirrettävänä oli yhteensä 28:n kumppanin liittymät, joilla kulki 37 erityyppistä sanomaa. Kaikki liittymät eivät olleet aktiivisia, mutta myös ne oli siirrettävä. Työn toteutuksesta sekä siirrettyjen liittymien testaamisesta oli tehtävä suunnitelma, työn kulun seuranta oli pidettävä ajan tasalla, ja työn vaiheet sekä siirretyt liittymät oli dokumentoitava huolella.

2.2 Kehityshaasteet

Vastaavanlaisia siirtoja eli migraatioita oli iSuitelle tehty ennekin, mutta yleensä pienemmässä mittakaavassa siirtäen alustalta toiselle yksittäisiä liittymiä. Tällä kertaa siirrettäviä liittymiä oli poikkeuksellisen paljon, ja vanha integraatioalusta oli saatava kokonaan pois käytöstä. Työn suorittamisen haastavimmat asiat on listattu alle kehityshaasteiksi:

- Haaste 1: Kuinka varmistaa, ettei tuotantoympäristöön aiheudu liittymien siirrosta haittoja?

- Haaste 2: Kuinka dokumentoida tehty työ niin, että seuraavan migraation tai muun konfiguraatiotyön yhteydessä työ käy helpommin?
- Haaste 3: Kuinka varmistaa, että kaikki liittymät on siirretty onnistuneesti?
- Haaste 4: Miten siirtää asiakas.dir-entrypoint, jolla on erittäin paljon liikennettä?

Taulukkoon 1 on listattu hyväksyttävästi tehdyn työn vaatimukset. Liittymien tekninen dokumentointi tulisi toteuttaa niin, että liittymien konfiguraatioiden kannalta oleellisimmat asiat ovat helposti saatavilla. Asiakas.dir-entrypointilla oli yhteensä 19:ta kumppanin liikenne ja yhtä monta eri sanomatyyppiä ja transformaatiota, joten kaikkien siirtäminen kerrallaan ei ollut järkevää vaan tarvittaisiin toisenlainen toteutustapa.

Taulukko 1. Toiminnalliset vaatimukset

Vaatimus id	**Kuvaus**	**Lisätiedot**	**Pakollinen**
Toiminnallinen vaatimus 1	Kaikki Asiakkaan liittymien käytössä olevat transformaatiot tekevät muunnoksen oikein	Käytetään testaamisessa tuotantoaineistoa, jota voidaan verrata tuotannossa onnistuneeseen transformaatioon	x
Toiminnallinen vaatimus 2	Kaikki Asiakkaan liittymien entrypointit hakevat oikean tiedoston noutokansiosta	Tuotannossa täytyy liikkua sanoma onnistuneesti	x
Toiminnallinen vaatimus 3	Kaikki Asiakkaan ja kumppanien liittymien käytössä olevat messagescriptit toimivat oikealla tavalla	Tuotannossa täytyy liikkua sanoma onnistuneesti	x
Toiminnallinen vaatimus 4	Kaikkien liittymien sanomat menevät oikealle noden endpointille	Tuotannossa täytyy liikkua sanoma onnistuneesti	x
Toiminnallinen vaatimus 5	asiakas.dir entrypointin liikenne on saatu kokonaisuudessaa siirrettyä		x
Toiminnallinen vaatimus 6	Siirrettävä liittymä on dokumentoitu kuten erikseen määritetty		
Toiminnallinen vaatimus 7	Kaikki AMQ:n liittymät on suljettu		
Toiminnallinen vaatimus 8	Asiakkaan sanomaliikenne ei saa häiriintyä siirroista		
Toiminnallinen vaatimus 9	Siirrettävä liittymä on testattu testiympäristössä		

3 Vanhenevan tietojärjestelmän ongelmat

Tietojärjestelmä on Schmidtin & Schmidtin (2013) mukaan monitahoinen ihmisen, ohjelmistojen ja fyysisten laitteiden, kuten tietokoneiden ja tiedonsiirtovälineiden, muodostama ihmisen kehittämä tuote jotain tiettyä tarkoitusta varten. Suurin osa erilaisista tietojärjestelmistä on kehitetty tehostamaan liiketoimintaa lisäämällä automaatiota ja vähentämällä mahdollisesti kalliiksikin käyviä virheitä, joita manuaalinen työ saattaa aiheuttaa. (Schmidt & Schmidt 2013, 2–4.)

Haikala ja Märijärvi (2004) luokittelevat ohjelmistoja erityyppisiin kategorioihin. Niitä ovat esimerkiksi erilaiset varus- ja työkaluohjelmistot kuten käyttöjärjestelmät tai tietokantajärjestelmät. Teknis-tieteelliset ohjelmistot ovat esimerkiksi erilaisia suunnitteluohjelmistoja. Asiantuntijajärjestelmiä ovat yleensä tietyn alan käyttöön erikoistuneet järjestelmät. Lisäksi on kaupallis-hallinnollisia ohjelmistoja, kuten palkanlaskenta tai tuotannonohjaus, sekä prosessinohjausjärjestelmiä ja erilaisiin teknisiin laitteisiin sulautettuja järjestelmiä. (Haikala & Märijärvi 2004, 17–18.)

Ohjelmistolla on elinkaari, joka alkaa sen kehittämisen aloittamisesta ja päättyy sen käytöstä poistamiseen. Ohjelmiston ylläpitovaihe voidaan jakaa korjaavaan, täydentävään ja adaptiiviseen ylläpitoon. Korjaavassa ylläpidossa korjataan ohjelmiston virheitä, täydentävässä ylläpidossa ohjelmistoon lisätään ominaisuuksia ja adaptiivisessa ylläpidossa ohjelmistoa muutetaan vaatimusten muuttuessa. Yleensä näitä lisäyksiä ja muutoksia tehdään ohjelmiston seuraavaan versioon, mutta tärkeimpiä päivityksiä tarjotaan myös verkon välityksellä jaettavina muutostiedostoina. (Haikala & Märijärvi 2004, 36, 41.)

Ylläpitovaiheessa ohjelmistoon saatetaan joutua tekemään muutoksia esimerkiksi laitteiston uusimisen vuoksi, mikäli vanha ohjelmisto ei ole uuden laitteiston kanssa yhteensopiva. Uusia ominaisuuksia tai päivityksiä lisättäessä ohjelmistoon tulee mahdollisesti myös uusia virheitä, joita joudutaan korjaamaan. Useamman ohjelmistoversion ylläpito vaatii paljon työvoimaa ollen kallista ja sitä hankalampaa mitä erilaisempia eri versiot ovat. Tämän vuoksi ohjelmistojen omistajat yleensä julkaisevatkin EOS (End of Support) ajankohdan, jonka jälkeen ohjelmistoa ei enää ylläpidetä ja kehitystiimi voi keskittyä tekemään ohjelmiston seuraavaa versiota. Tuen lopettaminen hyödyttää ohjelmistojen valmistajia myös siten, että asiakkaiden on tuen loppuessa yleensä syytä päivittää järjestelmänsä uudempaan versioon. (Gordon-Byrne 2014.) Sandborn (2007) tähdentää, että eten-

kin sovellukset, jotka ovat yhteydessä julkiseen verkkoon, muuttuvat tuen loppumisen jälkeen tietoturvariskeiksi, koska vaadittavia turvallisuuden päivitystiedostoja ei enää saa. Ohjelmistojen vanheneminen aiheuttaa ongelmia myös ohjelmistojen välisten rajapintojen hallinnassa. (Sandborn 2007).

Yksi yleisimmistä ja vaikeimmin selvitettävistä syistä ohjelmistojen toiminnassa muodostuviin ongelmiin ovat erilaiset versionkorotukset ja korjaukset muihin ohjelmistoihin, etenkin käyttöjärjestelmiin, joiden toiminnasta ohjelmisto on riippuvainen. Myös erilaisia konflikteja ohjelmistojen välillä esiintyy, jolloin toinen ohjelmisto voi jostain syystä vaikeuttaa toisen toimintaa. Syitä voi olla erittäin vaikea tai jopa mahdoton löytää. (Gordon-Byrne 2014.)

Ylläpitovaihetta seuraa ohjelmiston vanheneminen (software obsolescence) toiminnallisista, teknisistä tai logistisista syistä, joka yleensä johtaa ohjelman käytöstä poistoon ennemmin tai myöhemmin (Sandborn 2007). Vaikka tuen loppuminen ei tarkoita ohjelmiston välitöntä vanhentumista, on sen käytön jatkaminen hankalampaa ja riskialttiimpaa, ja monesti uusimpia ominaisuuksia ei enää vanhempiin ohjelmistoversioihin ole saatavilla. Jotkin yhä toiminnassa olevat ohjelmistot voivat olla niin vanhoja ja huonosti dokumentoituja, että niihin päivityksien tekeminen on erittäin vaikeaa. (Gordon-Byrne 2014.)

Vanhoja ohjelmistoja, joita sanotaan legacy-ohjelmiksi, käytetään monesti siksi, että ne yhä toimivat ja ovat tärkeitä yrityksen toiminnalle. Niiden korvaaminen uusilla voi olla kallista ja aikaa vievää, ja työvoimaa muutokseen voi olla hankala löytää. Ohjelmiston muuttuminen legacyksi johtuu yleensä tuen loppumisesta tai siitä, ettei ohjelmisto skaalaudu enää riittävästi. Syynä voi olla myös vanhentunut ohjelmisto, jota on pidetty toimintakunnossa erilaisilla päivityksillä erittäin pitkään, ja sen käyttämä tekniikka voi olla niin vanhaa, ettei osaajia ylläpitoon löydy. (Barney, Rouse & Mell 2022.) Birchall (2016) lisää, että monesti vanhat ohjelmistot ovat suuria, ja niitä on vaikeaa ja riskialtista yrittää korvata, joten niiden käyttö jatkuu, vaikka ne vanhenevat. Yleensä niiden alkuperäiset kehittäjät ovat jo vaihtuneet uusiin tekijöihin ja ylläpitäjiin, ja samalla ymmärrys monesti monimutkaisen ja puutteellisesti dokumentoitujen ohjelmiston toiminnasta hämärtyy. (Birchall 2016.)

Vanhan ohjelmiston käyttöön liittyy riskejä. Turvallisuusriskien lisäksi niiden ylläpito on kallista, niiden suorituskyky heikkenee, resurssienkäyttö kasvaa ja virheitä tapahtuu useammin. Niiden integrointi toisiin järjestelmiin ei välttämättä onnistu, jolloin niihin varastoitu tieto ei ole muiden järjestelmien käytössä. Joissain tapauksissa vanhat järjestelmät eivät täytä erilaisia vaadittuja standardeja. (Barney ym. 2022.) Ongelmaksi vanhojen ohjelmistojen kanssa muodostuu myös se, ettei niihin useinkaan saada tehtyä uusia ominaisuuksia, jolloin ohjelmiston käytön tarjoamat hyödyt voivat jäädä jälkeen esimerkiksi kilpailijoiden vastaavista, uudemmista tuotteista (Birchall 2016).

Ohjelmistoissa on lähes aina useita riippuvuuksia erilaisista kolmansien osapuolien tekemistä ohjelmistoista ja komponenteista. Open Source Dependency Management: Trends and Recommendations (n.d.) artikkelin mukaan Java-ohjelmistolla on keskimäärin 148 riippuvaisuutta, ja näitä ulkopuolisia kirjastoja päivitetään keskimäärin 10 kertaa vuodessa. Project quality metrics (n.d.) artikkelin mukaan yleisimmin käytetyillä kirjastoilla on lisäksi keskimäärin 5,7 periytyvää riippuvuutta muista komponenteista, ja kuusi seitsemästä haavoittuvuudesta löytyy näistä periytyvistä riippuvuuksista. Ohjelmiston vanheneminen on yksi merkittävimmistä ohjelmistoihin liittyvistä tietoturvaongelmista, koska näissä komponenteissa voi olla mitä tahansa vaarallisimmiksi luokitelluista haavoittuvuuksista (A security practioners's guide to software obsolescence 2020). Näin ollen on erittäin tärkeää pitää huolta siitä, että ohjelmiston käytössä ovat päivitetyt, turvallisimmat versiot komponenteista. Tuen loputtua näin ei kuitenkaan ole. Ulkopuolisten kirjastojen käytössä on riskinä myös se, että jotkin niistä saattavat olla nimenomaan lataavan tahon vahingoittamiseen suunniteltuja (Open Source Supply, Demand and Strategy n.d.).

Vaikka tietojärjestelmät ovat nykyään arkipäivää lähes jokaisen elämässä sekä työ- että vapaa-ajalla, eikä niiden käytöltä oikein voi välttyä, ovat integraatiot sen sijaan tuntemattomampi osa-alue tietojärjestelmien käyttäjille. Järjestelmien väliset Integraatiot pysyvät yleensä käyttäjiltä piilossa, eikä niiden olemassaoloa välttämättä edes tiedosta, ennen kuin jokin menee vikaan. Silti integraatioiden merkitys hyvin toimivissa järjestelmäkokonaisuuksissa on huomattava.

4 Järjestelmäintegraatiot

4.1 Järjestelmäintegraation perusteita

4.1.1 Järjestelmäintegraation kuvaus

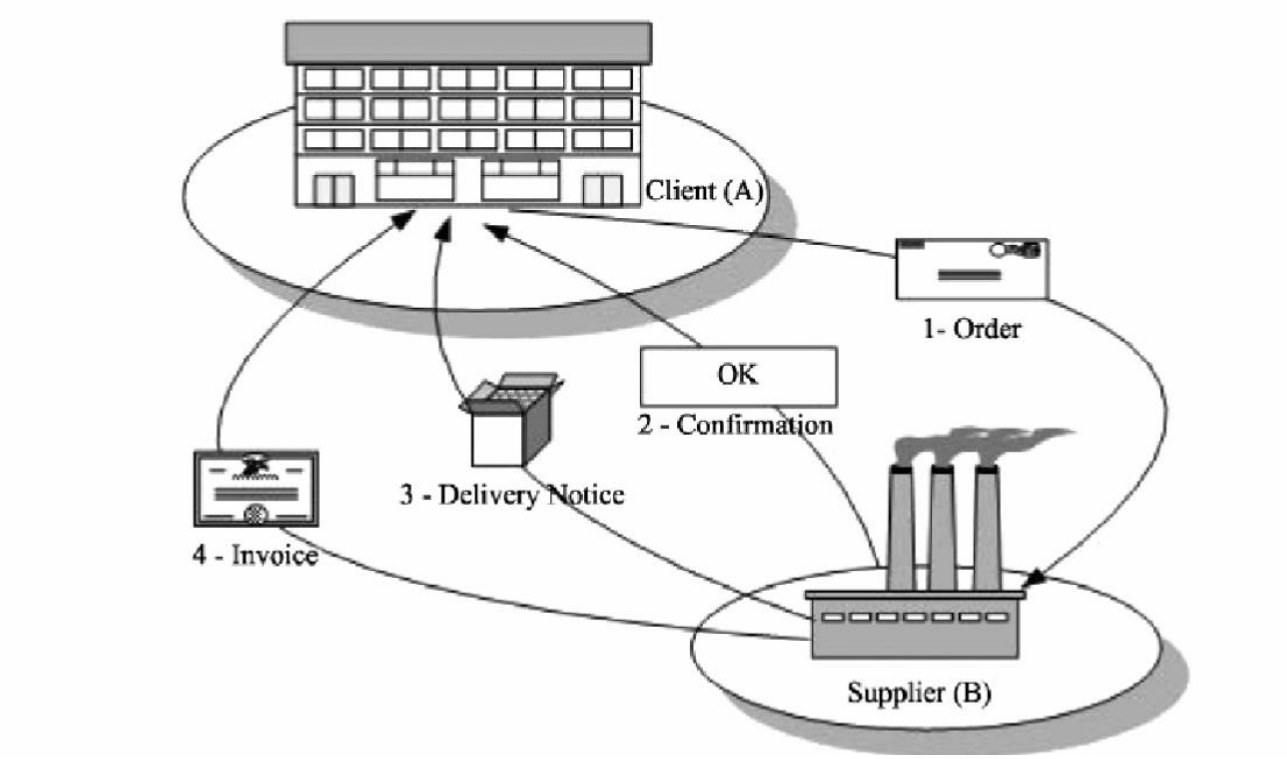
On hyvin yleistä, että ajan saatossa tietotekniikan yleistyttyä monilla yrityksillä on otettu käyttöön tietojärjestelmiä liiketoiminnan eri osa-alueille ajattelematta sen suuremmin kokonaisuutta. Myöhemmin näiden toisistaan erillisten ja monesti eri tekniikoilla toteutettujen järjestelmien välille on täytynyt saada kommunikaatiota liiketoiminnan tehostamiseksi ja virheiden vähentämiseksi. Näin syntyi tarve järjestelmäintegraatiolle. (Sherif 2010.) Tuo tarve on ajan myötä kasvanut, ja Annenko (2022) toteaakin yrityksien käytössä olevin järjestelmien ja sitä myötä tarvittavien integraatioiden määrän jatkavan kasvamistaan.

Tähtisen (2005) mukaan ”Järjestelmäintegraatio on kokoelma toimintatapoja, joiden avulla yrityksen tietotekniset järjestelmät saadaan valjastettua mahdollisimman hyvin yrityksen liiketoiminnan tarpeisiin.” Hän määrittelee järjestelmäintegraation suppeasti eri teknologioksi ja toimintatavoiksi, joiden avulla muuten yhteensopimattomat tietojärjestelmät saadaan kommunikoimaan keskenään automaattisesti. Järjestelmäintegraatio ei kuitenkaan ole pelkkä teknologia tai tuote, jonka voi ottaa käyttöön, vaan ennen kaikkea sitä pitäisi ajatella liiketoiminnan tehostamisen työkaluna. Järjestelmäintegraatiota voi verrata hajautettuun sovelluskehitykseen, jossa useiden eri komponenttien avulla muodostetaan toimiva sovellus. (Tähtinen 2005, 13–14.)

Manouvrier ja Menard (2008) tarkentavat määrittelyä vielä toteamalla, että järjestelmäintegraatiota (EAI, Enterprise Application Integration) ei pidä sekoittaa sovelluksen sisäisiin eri komponenttien väliseen informaation vaihtoon, joka on sovellusarkkitehtuurin keinoin ratkottava ongelma. He myös painottavat sitä, että järjestelmäintegraatio käsittää nimenomaan heterogeenisten sovellusten välisen kommunikoinnin, koska homogeeniset sovellukset ovat jo integroituja. (Manouvrier & Menard 2008, 23–24.)

Järjestelmäintegraation yhteydessä puhutaan yrityksen sisäisestä integraatiosta (EAI) sen omien järjestelmien välillä, kuten esimerkiksi laskutustietojen välittämisestä, sekä yritysten välisestä integraatiosta, B2Bi (business-to-business integration) tai B2Bai (business-to-business application integration), jossa välitetään esim. tilaus- ja laskutustietoja (Tähtinen 2005, 24, 34). Nämä eroavat

siten, että yritysten välisissä integraatioissa (ks. kuvio 1) yrityksen mahdollisuudet vaikuttaa tiedonsiirron tapaan sekä tiedonvaihdon formaattiin ja prosessiin ovat pienemmät (Manouvrier & Menard 2008, 25). Integraatioon haasteita tuo se, että eri yrityksillä on harvoin yhteinen tietoverkko, sama tiedon formaatti ja tiedonvaihdon tapa (Manouvrier & Menard 2008, 112). Tähtinen (2005, 96) tähdentää, että toimiva yritysten välinen integraatio edellyttää myös toimivaa sisäistä integraatiota. Termiä EAI käytetään toisinaan molemman tyyppisistä järjestelmäintegraatioista.



Kuvio 1. Yritysten välinen viestien vaihto (Manouvrier & Menard, 2008, 112).

ERP-järjestelmä (Enterprise resource planning eli yritysresurssisuunnittelu) on yritystoimintaan suunniteltu ohjelmisto, jonka avulla hallitaan useita eri liiketoiminnan osa-alueita, kuten kirjanpitoa, henkilöstö- sekä asiakastietoja (Samara 2015, 10). ERP-järjestelmien yleistyttyä myös järjestelmäintegraatioiden tarve kasvoi, koska ERP-järjestelmät kattoivat harvoin aivan kaikkia liiketoiminnan osa-alueita, eikä niitä yleensä ollut suunniteltu kommunikoimaan ulkopuolisten järjestelmien kanssa (Manouvrier & Menard 2008, 99). Samara (2015) toteaa, että Schonefeldin ja Veringin (2000) mukaan yritykset ovat vain harvoin kokonaan luopuneet vanhoista järjestelmistään ERP:n käyttöönoton yhteydessä. Näin ollen vanhoja järjestelmiä on järjestelmäintegraation keinoin jouduttu yhdistämään ERP:n kanssa. (Samara 2015, 22.)

Haasteita integraatioille aiheuttaa lukuisten erilaisten integroitavien, tiettyyn tehtävään erikoistuneiden järjestelmien lisäksi se, että niihin ei yleensä voi tehdä muutoksia. Ongelmat on ratkaistava integraation avulla. Toisaalta toimimaton integraatio voi aiheuttaa ongelmia ja ylimääräisiä kustannuksia liiketoiminnalle. Integraatioiden ylläpito ja etenkin virheiden löytäminen ja korjaaminen on haastavaa ja vaatii monesti erityisiä taitoja. (Hohpe & Woolf 2003b.) Toivasen (2022) mukaan integraatioiden kuluista merkittävä osa tuleekin niiden ylläpidosta.

4.1.2 Järjestelmäintegraation hyödyt

Järjestelmäintegraation keinoin tapahtuvalla tietoteknisten ratkaisujen tehostamisella yritykset pyrkivät tehostamaan liiketoimintaprosessejaan. Informaation välittyminen automaattisesti eri prosessien tietojärjestelmien välillä vähentää virheitä ja nopeuttaa prosesseja, mikä puolestaan lisää kilpailukykyä ja säästöjä. Kun järjestelmäintegraatio on hyvin suunniteltu, saadaan tietojärjestelmäarkkitehtuurista yksinkertaisempi, mikä helpottaa sen ylläpitoa. Integraation avulla voidaan myös laajentaa yksittäisen ohjelmiston käyttökelpoisuutta ja pidentää sen elinkaarta. Hyvin suunnitellun integraation keinoin voidaan lisäksi parantaa yrityksen joustavuutta mahdollistamalla nopeampi reagointi erilaisiin organisaation tai prosessien muutoksiin. (Tähtinen 2005, 23, 25–27.)

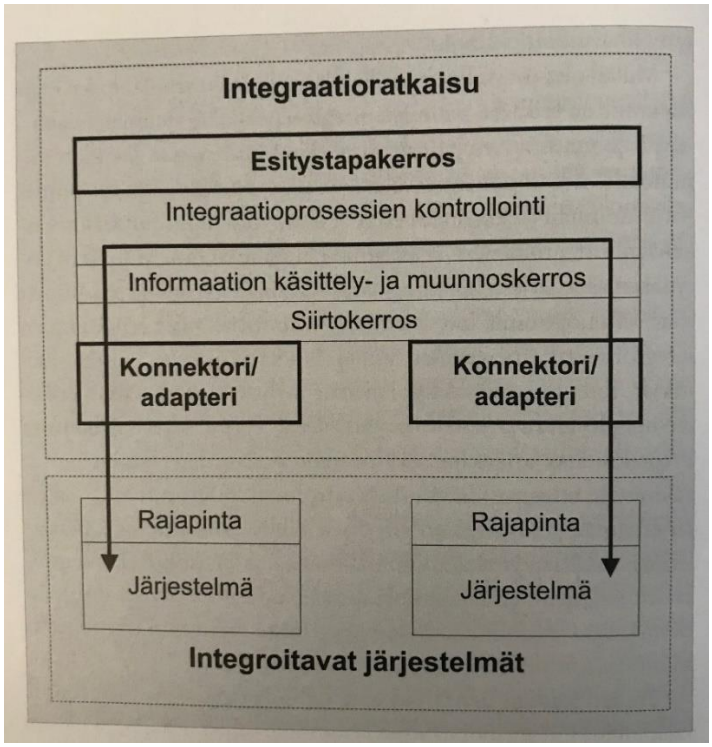
Heritage (2014) esittää, että integraation tulee varmistaa tiedon kohteeseensa meneminen ja toisaalta duplikaatteja ei saa muodostua. Sen täytyy mahdollistaa heterogeeniset ympäristöt ja sen on oltava helposti muokattavissa ja skaalattavissa vaatimusten muuttuessa. Ratkaisun tulee täyttää liiketoiminnan vaatimukset ja sen on oltava suorituskykyinen sekä varmistettava tiedon suojaus. (Heritage 2014.) Koska järjestelmäintegraatioratkaisulla on koko ajan pääsy yrityksen liiketoiminnan tietoihin, voidaan sitä käyttää myös liiketoimintaan liittyvään raportointiin ja monitorointiin (Tähtinen 2005, 32).

4.1.3 Järjestelmäintegraation käsitteitä

Järjestelmäintegraatio voidaan jakaa muutamiin osa-alueisiin, jotka integraatioista on yleensä tunnistettavissa. Nämä osa-alueet näkyvät kuviossa 2. Tähtinen (2005, 48) määrittelee järjestelmäintegraation tarkemmin:

> *toimintamalleiksi ja tekniikoiksi, joiden avulla voidaan saattaa vähintään kaksi eri toiminnallisuutta tarjoavaa tietojärjestelmää jakamaan informaatiota siten, että informaation siirto ja muunnokset ovat kontrolloitavissa ja monitoroitavissa yhdestä tai useammasta keskitetystä pisteestä.*

Integraatioon kuuluu Tähtisen (2005) mukaan siis tiedon siirto, mahdolliset muunnokset sekä näihin liittyvä kontrollointi, valvonta ja raportointi. Eri järjestelmien välisen integraation mahdollistamiseksi tarvitaan järjestelmien välille joku fyysinen väylä tiedon siirtoon sekä jonkinlaiset rajapinnat, joista tieto järjestelmiin ja niistä pois liikkuu. (Tähtinen 2005, 48–49.)

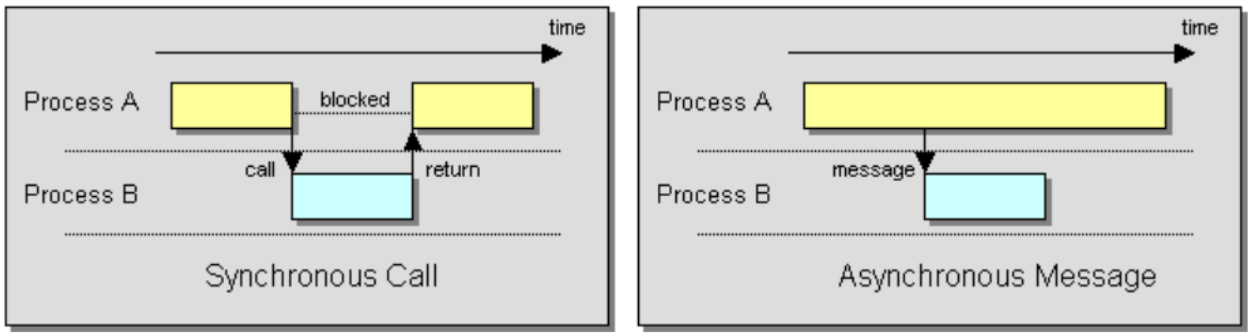


Kuvio 2. Integraation malli Tähtisen mukaan (Tähtinen 2005, 73).

Ennen kuin tiedonsiirtojärjestelmät olivat tarpeeksi kehittyneitä, saattoi integroitavien järjestelmien välinen tiedonvaihto tapahtua esimerkiksi toimittamalla fyysinen tallennusväline paikasta toiseen (Sherif 2005). Nykyään järjestelmien välinen tiedonsiirto tapahtuu lähes poikkeuksetta tietoverkkojen, etenkin internetin, välityksellä (Manouvrier & Menard 2008, 114). Verkon yli tehtäviä integraatioita tehdessä on huomioitava, että verkkoyhteydet saattavat olla epäluotettavia ja hitaita, ja myös tietoturva on otettava erityisesti huomioon (Hohpe & Woolf 2003a).

Integroitavien järjestelmien rajapinnat (API, application programming interface) ovat erityisen tärkeitä integraatioiden kannalta. Rajapintojen kautta järjestelmän tietoja voidaan hakea ja sinne voidaan syöttää tietoa. Ne ovat järjestelmästä riippuen asetuksiltaan, toteutustavoiltaan ja parametreiltaan erilaisia ja ne yhdistyvät integraatioratkaisun rajapintakomponentteihin. Näistä rajapinnoista käytetään monesti nimityksiä konnektori, adapteri tai agentti. (Tähtinen 2005, 49,71.)

Tähtinen (2005), kuten myös Sherif (2010) jakaa tiedonsiirron neljään luokkaan: tiedonsiirto voi tapahtua sanomanvälityksenä, tiedostonsiirtona, jaettuna tietokantana tai etäohjelmistokutsuna. Tiedonsiirto voi olla synkronista tai asynkronista kuten kuviossa 3 näkyy. Asynkroninen tiedonsiirto tarkoittaa sanomien välittämistä yleensä erilaisissa jonotusjärjestelmissä, joissa lähettäjä voi unohtaa sanoman sen mentyä jonoon ja vastaanottaja hakee tiedon jonosta silloin kun sopii. Synkronisessa tiedonvälityksessä, kuten etäohjelmistokutsut, kutsuva ohjelmisto odottaa vastausta ennen jatkamistaan. Jaetussa tietokannassa varsinaista tiedon siirtoa ohjelmien välillä ei tapahdu. (Tähtinen 2005, 52, 123–125; Sherif 2010.) Tiedostonsiirto ohjelmien välillä voitaneen myös tulkita jossain määrin asynkroniseksi välitykseksi. Toisaalta Heritage (2014) mainitsee myös synkronisen sanomanvälitysvaihtoehdon. Integraatioiden yhteydessä sanomanvälitys on yleensä asynkronista.

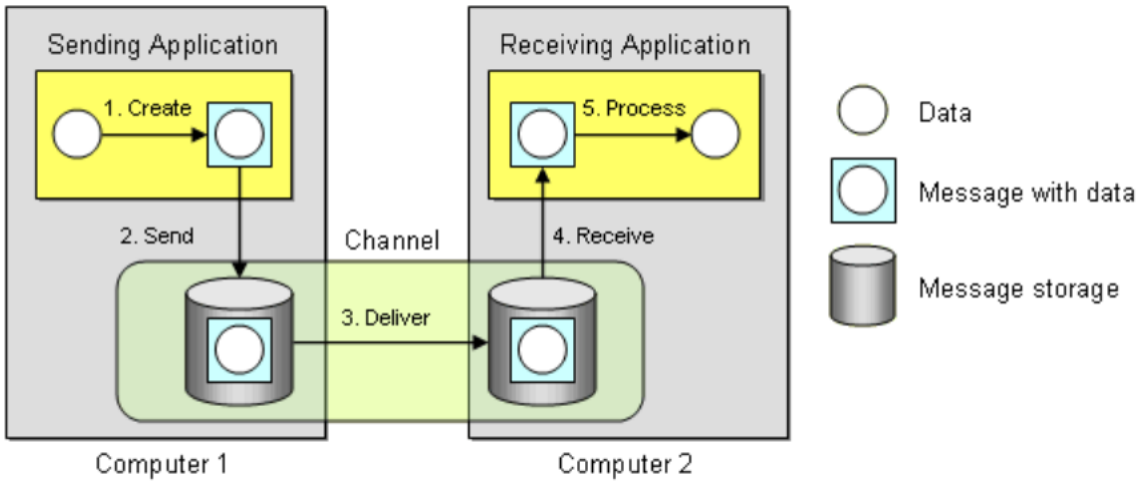


Kuvio 3. Synkronisen ja asynkronisen tiedonsiirron ero (Hohpe & Woolf 2003a).

Integraatioratkaisuissa sanomanvälitys on ohjelmistojen moninaisuuden vuoksi tärkeä ominaisuus (Tähtinen 2005, 125). Se mahdollistaa nopean, asynkronisen ja luotettavan tiedonsiirtotavan erilaisten ohjelmistojen välille (Hohpe & Woolf 2003a). Sanoma sisältää käyttötarkoituksesta riippuen erilaista informaatiota. Se on itsenäinen kokonaisuus, jolla on yleensä otsikko-osio (headers), joka

sisältää tietoa sanomasta kuten reititykseen, käsittelyyn ja esitysmuotoon liittyvää tietoa, sekä varsinainen sanoma hyötykuorma (payload) -osiossa. (Tähtinen 2005, 124.)

Sherifin (2010) mukaan sanomanvälitykseen pohjautuvassa integraatiossa järjestelmät kommunikoivat väliohjelmiston, middlewaren, kautta. Väliohjelmisto varmistaa sanomien kulun oikeaan paikkaan. Sanomapohjaisesta väliohjelmistosta käytetään termiä MOM (Message-Oriented Middleware). (Sherif 2010.) Jonojärjestelmän avulla sanoma tallennetaan jonoon odottamaan siirtymistä lopulliseen päämääräänsä. Lähettäjä voi unohtaa sanoman ja luottaa, että se löytää tiensä vastaanottavalle järjestelmälle. (Toivanen 2022.) Jonotusjärjestelmä selvittää esimerkiksi huonon verkkoyhteyden vuoksi aiheutuvat ongelmat toistamalla sanoman lähettämistä, kunnes se onnistuu. Vastaanottaja voi myös säätää vastaanottamiensa pyyntöjen määrää niin ettei ruuhka kasva liian suureksi. (Hohpe & Woolf 2003a.) Kuviossa 4 esitellään sanomanvälityksen vaiheet ja osapuolet.

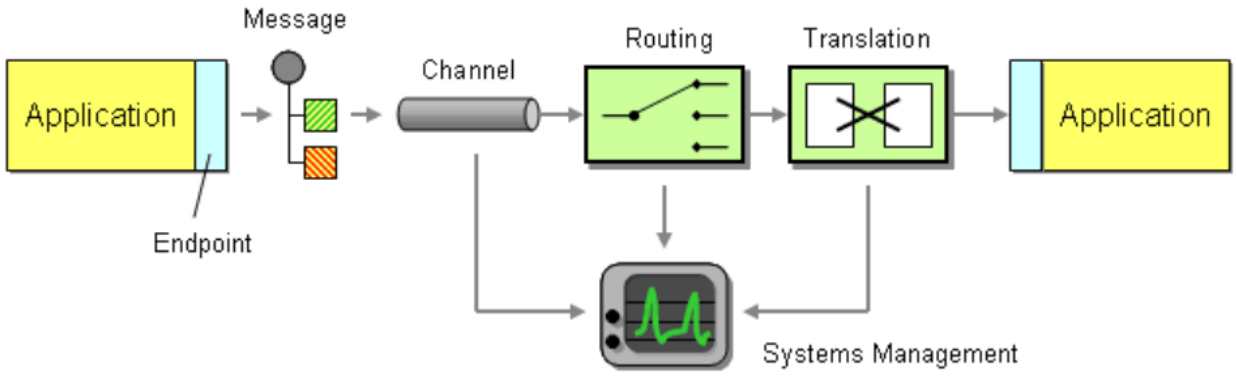


Kuvio 4. Sanomanvälitys (Hohpe & Woolf 2003a)

Publish/subrcribe-teknologian avulla lähettäjä voi julkaista tietoa, josta etukäteen ilmoittautuneet halukkaat tilaajat saavat ilmoituksen ja voivat ottaa haluamansa tiedon vastaan. Ilmoituksesta voidaan käyttää myös termiä tapahtuma. Publish/subscribe teknologia pitää ilmoituksen jälkeen huolta siitä, että tilaajat saavat tilaamansa tapahtuman tiedon, ja viestinnän osapuolet eivät välttämättä ole toisistaan edes tietoisia. Publish/subscribe onkin hyvä tapa välittää tietoa hajautettujen komponenttien välillä. Publish/subscribe-teknologia soveltuu moniin eri käyttötarkoituksiin ja

sen pohjalta on tehty paljon erilaisia järjestelmiä, joita yhdistää erilaisten järjestelmien välillä tapahtuva asynkroninen yhdestä moneen -kommunikaatio. (Tarkoma 2012, 2–4)

Koska eri sovellusten käyttämä informaatio on usein eri muodossa, tarvitsee informaation vaihdon yhteydessä monesti tehdä siihen erilaisia muunnoksia, jotta tiedon vastaanottava järjestelmä ymmärtää lähettävältä järjestelmältä saamansa informaation. Tiedon siirron ja mahdollisten muunnosten määrittelyä varten tarvitaan keino kontrolloida integraatiota. (Tähtinen 2005, 56, 59.) Lisäksi muunnoksia voi kohdistua myös käytettävään protokollaan (Toivanen 2022). Kuviossa 5 näkyvät integraatioiden peruselementit. Kuvion endpoint on yhdenlainen adapteri.



Kuvio 5. Integraation peruselementit (Hohpe & Woolf 2003b)

4.1.4 Integraatioprosessi

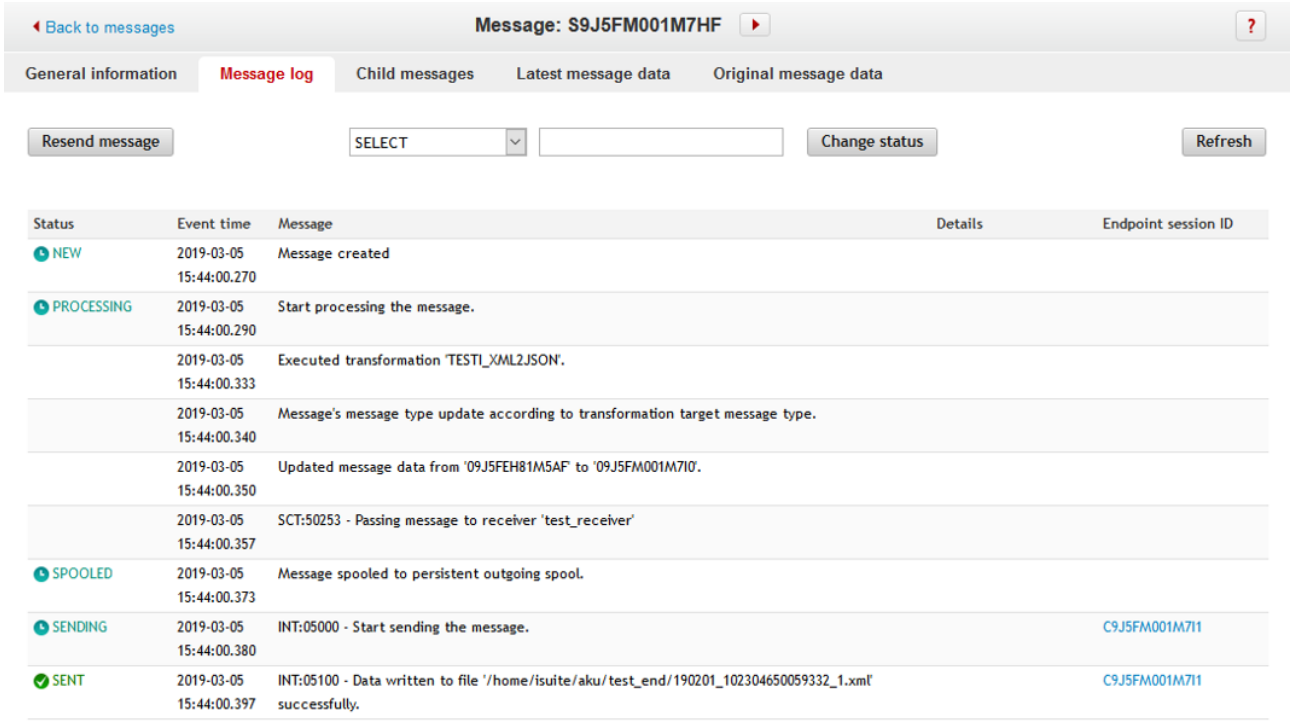
Integraatiossa tapahtuu yleensä joukko peräkkäisiä toimintoja. Aluksi adapteri saa informaation, jota seuraa informaation siirto ja siihen liittyvät toiminnot, minkä jälkeen tapahtuu mahdollisesti informaation muokkausta ennen kuin informaatio reititetään oikeaan paikkaan ja tarvittaessa varastoidaan. (Manouvrier & Menard 2008, 70.) Yleensä integraatioprosessin toiminnot tapahtuvat automaattisesti ja käynnistyvät joko spontaanisti, esimerkiksi ajastetusti, tai esimerkiksi käyttäjän tai toisen ohjelmiston pyynnöstä eli ulkoisesta herätteestä (Tähtinen 2005, 62–63).

Tähtisen mukaan prosessin ohjelmiston komponentteja ovat mm. käynnistymien, suoritus, joita voi olla useita erilaisia, kuten informaation hakeminen tai muuntaminen, sekä päätöksenteko eri-

laisissa tilanteissa prosessin aikana. Lisäksi komponentteina voi olla mahdollinen prosessin eri haarojen välinen synkronointi sekä prosessin lopettaminen ja siihen liittyen esimerkiksi kuittaussanoman lähettäminen. (Tähtinen 2005, 134–145.)

Integraatioratkaisu voi ajastusten ja ohjelmoidun logiikan avulla välittää tietoja järjestelmien välillä. Se voi toimia myös palvelukerroksena, jolle voidaan lähettää toisesta järjestelmästä pyyntö saada jokin informaatio, jonka integraatioratkaisu noutaa ja toimittaa kysyvälle järjestelmälle. (Tähtinen 2005, 94–95.) Toivanen (2022) lisää, että integraatioprosessi voi käynnistyä myös esimerkiksi tietokannan muutoksesta tai tiedoston saapumisesta.

Toivasen (2022) mukaan nykypäivän IT-ympäristössä vian aiheuttamasta korjaustyöstä suurin osa menee vian etsintään. Sen vuoksi Integraatioprosessia tulisi pystyä seuraamaan alusta loppuun, ja integraatiojärjestelmän tulisi lähettää informaatiota prosessin eri vaiheista sekä mahdollisia hälytyksiä virhetilanteista. Myös mahdollisuus manuaaliseen prosessiin puuttumiseen virhetilanteiden varalta on tarpeellinen. (Manouvrier & Menard 2008, 70–71.) Toivanen (2022) korostaa, että integraatioiden operoitavuus ja monitoroitavuus ovat ylläpidon kannalta oleellisia. Kuviossa 6 näkyy yksittäisen sanoman lokitiedot iSuite 6 integraatioalustalla.



Kuvio 6. iSuite 6:n message log (Hirvonen 2021)

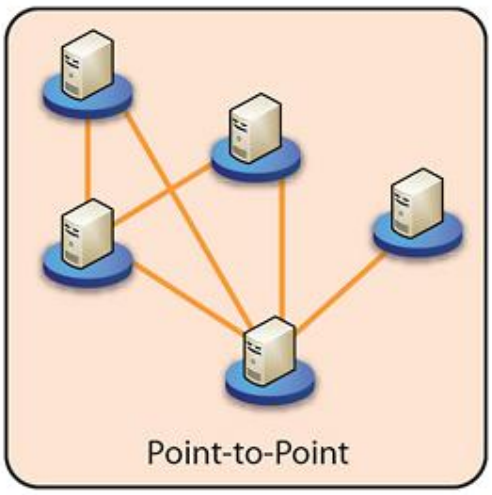
Usein uusia integraatioita tehdään liittämällä uusi järjestelmä johonkin olemassa olevaan kokonaisuuteen. Vaatimuksia integraatiojärjestelmälle voidaan kartoittaa selvittämällä järjestelmät, joiden kanssa uusi järjestelmä joutuu kommunikoimaan, millaista tietoa ja millaisia määriä aiotaan vaihtaa, kuinka osalliset järjestelmät pystyvät kommunikoimaan sekä mikä on tiedon vaihdon rytmi. (Manouvrier & Menard 2008, 100–101.)

4.2 Järjestelmäintegraation toteutustapoja

Järjestelmäintegraation toteuttamiseen on monenlaisia eri vaihtoehtoja. Tähtinen (2005) korostaa, että yrityksen tietoteknisen arkkitehtuurin tulisi olla järjestelmäintegraation suunnittelun pohjana, ja arkkitehtuurikuvauksen puolestaan pitäisi pohjautua yrityksen liiketoimintaprosesseihin teknisten yksityiskohtien ollessa sivuseikkoja. Sekä liiketoiminnassa että tekniikassa tapahtuvat kehittyminen ja muutokset tulisi huomioida. Yrityksen tietojärjestelmien kokonaisuuden hahmottaminen ja pyrkimys yksinkertaisuuteen ovat arkkitehtuurin kannalta olennaisia seikkoja. (Tähtinen 2005, 77–79, 90.)

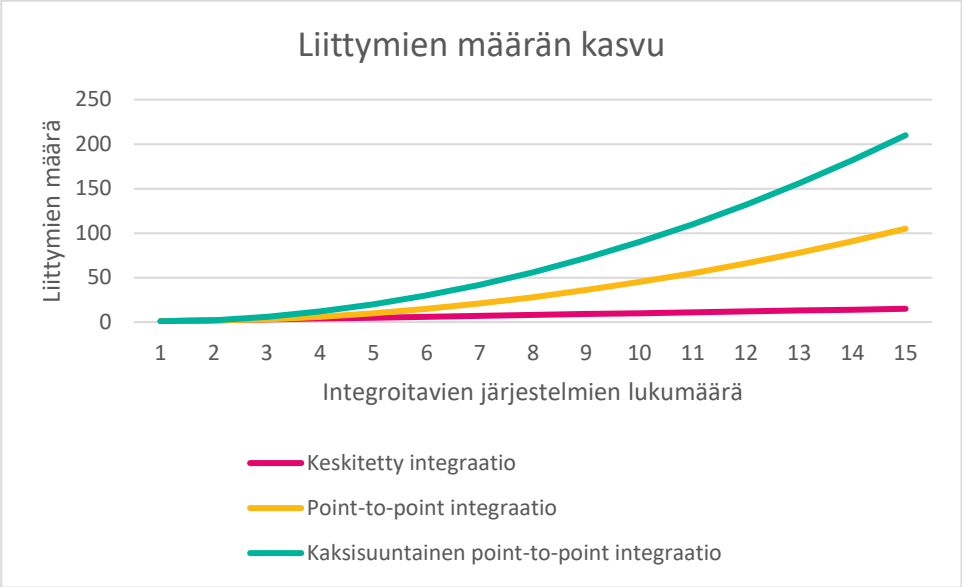
4.2.1 Point-to-point integraatiot

Järjestelmäintegraatiot voivat olla ns. point-to-point integraatioita (kuvio 7) kahden järjestelmän välillä. Point-to-point integraatioita on tehty järjestelmäintegraatioiden kehityksen alkuaikoina paljon, jolloin yksittäisiä järjestelmiä yhdistettiin keskenään. Tällaisessa integraatiossa sovelluksiin ohjelmoidaan toiminnallisuus, jolla pystytään joko siirtämään tietoa sovellusten välillä tai lähettämään kutsuja toisen sovelluksen tarjoamille palveluille. (Tähtinen 2005, 122.) Point-to-point integraatiot ovat yksinkertaisin ja melko edullinen tapa yhdistää kaksi järjestelmää, ja tällä tavoin voidaan toimia, mikäli yhdistettävien järjestelmien määrä on pieni (Sherif 2010).



Kuvio 7. Point-to-point integraatio (Hogg 2008)

Järjestelmien määrän kasvaessa point-to-point integraatiot ovat kuitenkin melko hankalia ylläpitää. Mikäli useampi järjestelmä n yhdistetään toisiinsa, tuloksena on kuviossa 8 näkyvä eksponentiaalisesti kaavalla $\frac{n(n-1)}{2}$ kasvava määrä liittymiä joiden hallinta on luonteeltaan hajautettua, minkä vuoksi niiden kontrollointi ja tarkkailu on vaikeaa. (Tähtinen 2005, 59, 65–66.) Toisaalta, kuten Sherif (2010) esittää, mikäli ajatellaan rajapintojen määrää, joka enimmillään on tehtävä, määrä lasketaan kaavalla n^2. Molempiin suuntiin kaikkien järjestelmien välille tehtäessä liittymien määrän laskukaava saa muodon $n(n-1)$. Sherif (2010) lisää, että myös muutoksiin reagoiminen point-to-point integraatioiden kyseessä ollessa on hankalaa, koska integroitujen järjestelmien suuren määrän lisäksi myös niiden rajapinnat ovat yksilöllisiä ja vaativat muokkaamista. Tähtinen (2005, 66) huomauttaa lisäksi, että mikäli point-to-point integraatioratkaisuun päädytään, on niiden huolellinen dokumentointi erittäin tärkeää.



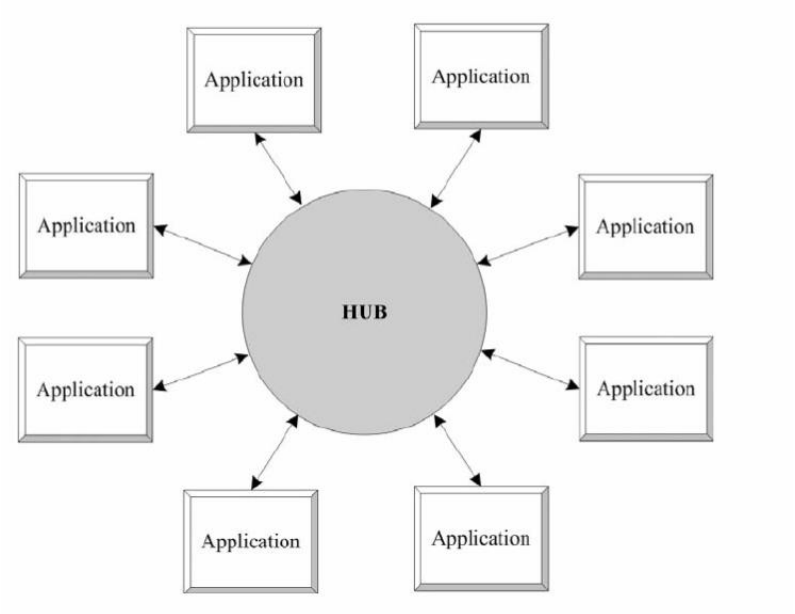
Kuvio 8. Liittymien määrän kasvu eri integraatiomalleilla (Tähtinen 2005, 67, muokattu).

4.2.2 Keskitetty integraatio

Koska point-to-point integraatiot eivät enää tule kysymykseen vähänkään suuremmassa integraatioprojektissa, on niiden tilalle kehitetty tehokkaampia, yleisemmin käytössä olevia integraatioarkkitehtuureja. Tähtinen (2005) esittelee termin keskitetty kontrolli kuvaamaan arkkitehtuuria, jossa järjestelmien välistä kommunikaatiota ja tietomuunnoksia voidaan kontrolloida yhdestä pisteestä. Tällaisessa keskitetyssä integraatioratkaisussa liittymiä lisättäessä niiden määrä kasvaa lineaarisesti. Keskitetty ratkaisu on yksinkertaisempi, joustavampi sekä helpompi ylläpitää, monitoroida ja kontrolloida. Kontrolloinnin ja valvonnan keskittäminen myös helpottaa tietoturvan hallinnointia vähentämällä järjestelmien välisten linkkien määrää. (Tähtinen 2005, 67–69, 181.) Toivanen (2022) toteaa, että tutkimusyhtiö Gartnerin mukaan keskitetyllä integraatioratkaisulla voidaan säästää jopa 50 % IT-kuluista verrattuna lukuisiin point-to-point-integraatioihin.

Myös hub-and-spoke-nimellä tunnetussa arkkitehtuurissa jokainen integroitava järjestelmä (spoke) on liitetty kuvion 9 mukaisesti väliohjelmistoon, hubiin, joka on keskitetty integraatioratkaisu (Tähtinen 2005, 143). Hub-and-spoke ratkaisussa lähettävän järjestelmän ei tarvitse määritellä vastaanottajaa vaan väliohjelmisto ohjaa sanoman oikeaan paikkaan sanoman sisällön ja muodon perusteella (Sherif 2010). Hub-and-spoke on toimiva ratkaisu liittymien määrän kasvaessa

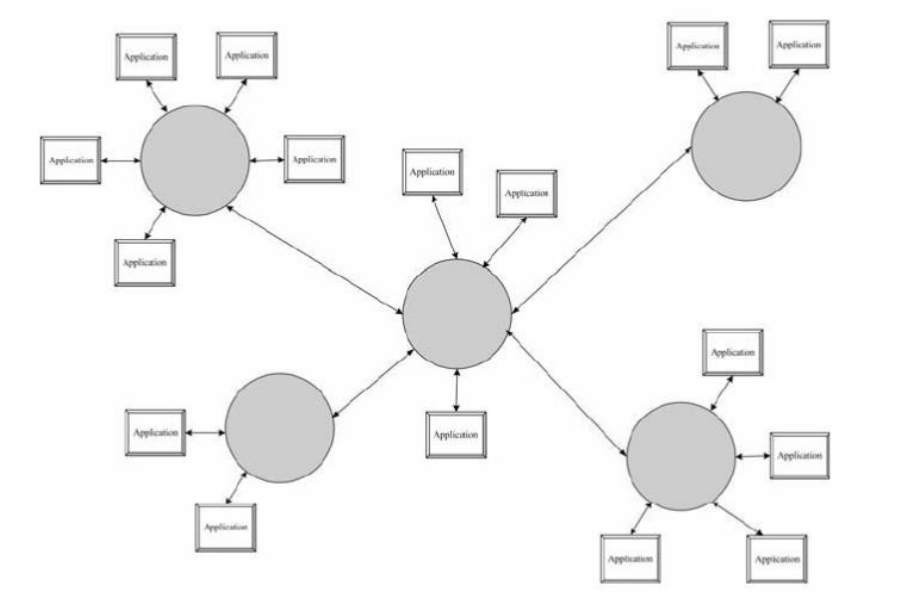
mutta ainoastaan tiettyyn rajaan asti. Jossain vaiheessa ratkaisun suorituskyky alkaa rajoittaa liittymien määrää. (Tähtinen 2005, 143–144.)



Kuvio 9. Hub-and-spoke arkkitehtuuri (Manouvrier & Menard 2008, 96)

Keskitetyssä ratkaisussakin on riskinsä. Koska kaikki tärkeä info kulkee yhden pisteen kautta, sen täytyy olla erittäin vikasietoinen (Tähtinen 2005, 67–68). Keskitetyn arkkitehtuurin ratkaisu muodostaa ns. single point of failuren (SPOF), jossa kontrollipisteen pettäminen esimerkiksi ylikuormituksen tai muun vian takia vaarantaa koko järjestelmän (Manouvrier & Menard 2008, 96).

Manouvrier ja Menard (2008) esittelevät lisäksi snowflakeksi (kuvio 10) nimeämänsä mallin, jossa on useita keskitetyn kontrollin pisteitä, joilla on omat vastuualueensa ja ne kommunikoivat tarvittaessa keskenään. Etenkin suuren yrityksen infrastruktuurin kyseessä ollessa voidaan tämän hajautetun arkkitehtuurin mallin avulla vähentää keskitetyn ratkaisun riskejä. (Manouvrier & Menard 2008, 96–98.) Myös Tähtinen (2005, 69) mainitsee, että keskitetyn kontrollin haasteita voidaan huolellisen suunnittelun ja toteutuksen sekä erilaisten varmennusten lisäksi hallita muodostamalla loogisia kokonaisuuksia joilla on omat integraatiopisteensä.



Kuvio 10. Snowflake-arkkitehtuuri Manouvrierin ja Menardin (2008, 98) mukaan.

4.2.3 Palvelupohjainen arkkitehtuuri

Hub-and-spoke mallia uudempi näkemys integraatioista on palvelupohjainen arkkitehtuuri (SOA, Service Oriented Architecture). Sen tarkoitus on tehdä ohjelmistokomponenteista uudelleenkäytettäviä. Eri liiketoiminnan osa-alueet jaetaan ohjelmistoihin eli palveluihin, jotka ovat toisistaan riippumattomia ja pyydettäessä esimerkiksi tarjoavat osa-alueensa tietoja tai tekevät jonkin muun yksittäisen liiketoimintasuoritteen. Ohjelmistokomponentit tarjoavat palvelurajapinnan, joilla on yhteinen kommunikointitapa verkon välityksellä, ja niiden palveluita voidaan kutsua verkon kautta. Palvelupohjainen arkkitehtuuri integroi nämä erilliset komponentit toimimaan yhdessä. (What is service oriented architecture (SOA) 2020; Sherif 2010.)

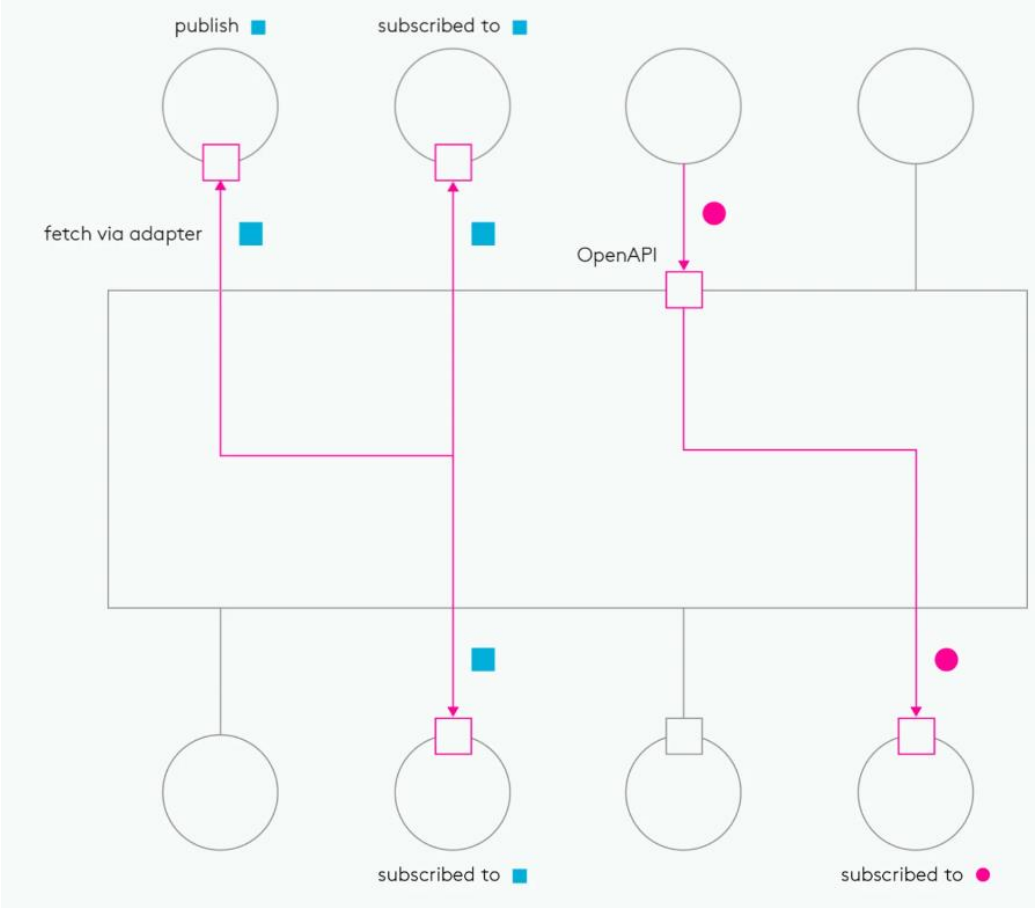
Palvelupohjaisessa arkkitehtuurissa osapuolina ovat palvelun tarjoaja, joka tarjoaa palveluaan palvelurekisterille tai palvelun välittäjälle, sekä palvelun kuluttaja, joka löytää tietoja tarjoavalta rekisteriltä tai välittäjältä haluamansa palvelun, johon ottaa yhteyttä (What is service oriented architecture (SOA), 2020). Tarkoma (2012, 292) tarkentaa, että SOA perustuu sanomanvälitykseen ja publish/subscribe -malliin, ja sen komponenttien on tarkoitus olla löyhästi kytkettyjä (loose coupling) mutta kuitenkin yhteensopivia verkon kautta saatavilla olevia palveluita, joita yhdistämällä voidaan luoda liiketoimintajärjestelmiä. Loose Coupling tarkoittaa sitä, että eri osapuolet tekevät mahdollisimman vähän oletuksia toistensa ominaisuuksista, jolloin muutoksiin reagoiminen on

helpompaa (Hohpe & Woolf 2003b). Tähtinen (2005) korostaa, että palvelupohjaisen arkkitehtuurin ideana on myös olla riippumaton käytettävistä tekniikoista. Palvelupohjaista arkkitehtuuria soveltaessa kehittäjät joutuvat erityisesti kiinnittämään huomiota avoimien rajapintojen tarjoamiseen ja palveluiden tehokkuuteen, mikä helpottaa järjestelmäintegraatiota (Tähtinen 2005, 97, 145).

Palvelupohjainen arkkitehtuuri tuo integraatioiden tekemiseen paljon lisää hyviä puolia. Se helpottaa ohjelmistojen uudelleenkäyttöä, tehostaa kehitystyötä ja lisää joustavuutta sekä helpottaa ylläpitoa. Se myös parantaa järjestelmän luotettavuutta ja skaalautuvuutta. (What is service oriented architecture (SOA), 2020.) Palvelupohjainen arkkitehtuuri auttaa myös miettimään uudelleen liitetoimintamalleja ja -prosesseja (Seth & Seth 2020). Tarkoman (2012, 292) mukaan SOA-toteutukseen on olemassa paljon valmiita suunnittelumalleja, jotka tehostavat integraatioiden tekemistä.

Haasteita palvelupohjaiselle arkkitehtuurin käyttöotolle aiheuttaa yrityksen yleensä hajallaan olevat ja erilaiset järjestelmät. Koska palvelupohjainen arkkitehtuuri hajauttaa infrastruktuurin, vaaditaan komponenteilta korkeaa saavutettavuutta ja skaalautuvuutta. (Seth & Seth 2020.) Toivanen (2022) lisää, että etenkin SOA:n alkuaikoina palveluiden ongelmana oli niiden raskaus ja hidas kehitys.

ESB, Enterprise Service Bus-ratkaisu (kuvio 11) on teknologia, joka toimii yleensä osana SOA arkkitehtuuria. Se välittää palvelupohjaisen arkkitehtuurin viestejä palvelulta toiselle tarjoten ikään kuin horisontaalisen väylän ohjelmistojen välille. (Tähtinen 2005, 145.) Toivasen (2022) mukaan ESB, suomeksi palveluväylä, kykenee välittämään viestejä monin eri tavoin ja se sisältää yleensä myös jonojärjestelmän publish/subscribe mallin lisäksi. Palveluväylää käytetään yleensä suurissa integraatioratkaisuissa. (Toivanen 2022.) Tarkoma (2012, 52) lisää, että ESB:lle ominaista on mm. tuki sekä synkronisen että asynkronisen viestinnän hyödyntämiselle, monipuoliset reititysominaisuudet, Message-oriented middleware sanoman käsittelyineen ja muunnoksineen, sekä Complex Event Processing mm. tapahtumien tunnistamiseen.



Kuvio 11. ESB (Toivanen 2022)

Palvelupohjaisen arkkitehtuurin liittyvistä tekniikoista Web Services-tekniikassa on hyödynnetty jo olemassa olevia suosittuja ja yksinkertaisia tekniikoita. Se pohjautuu sanomavälitykseen, jossa SOAP-protokollaa (Simple Object Access Protocol) käyttäen välitetään XML-pohjaisia sanomia yleensä HTTP(S)-siirtoprotokollalla. Tämä helpottaa etenkin yritysten välisessä viestinnässä palomuurien läpi pääsemistä kyseisten protokollien porttien ollessa yleensä avoimia. Palveluiden kuvaamiseen on kehitetty WSDL-kieli (Web Services Description Languge). Web services -pohjainen sanomanvälitys voidaan toteuttaa lukuisilla eri tavoilla, joten se soveltuu lähes kaikkiin ympäristöihin. Eri palvelut ovat löydettävissä UDDI-teknologian (Universal Description, Discovery and Integration) avulla. (Tähtinen 2005, 119.) Web Services hyödyntää yleensä ESB-toteutusta (Tarkoma 2012, 52). Kuviossa 12 on vertailtu tähän mennessä esiteltyjä integraatiotapoja.

	Notes	Managed	Centralized	Lines Of Connectivity	Scalability	Coupling	Management
Point-to-Point	Suitable for small environments	No	No	n^2	None	Tight	None
Hub and Spoke	Single point of failure	Yes	Yes	n	Limited	Supports loose	Centralized
	Message broker						
Message Bus	Common command and communication infrastructures	Yes	No	n	Very	Loose	Complex
	Proprietary communication protocols						

Kuvio 12. Integraatioarkkitehtuurien vertailu (Hogg, 2008).

4.2.4 Modernit integraatiot

Vaikka point-to-point linkeille on vielä nykyäänkin paikkansa, ja hub-and-spoke malli ja SOA ovat integraatioiden tekemisessä hyviä vaihtoehtoja, on integraatioiden tekeminen tänä päivänä monipuolistunut entisestään. Hautalan (2018) mukaan nykyisissä integraatioalustoissa on valmiina adaptereita yleisimpiin järjestelmiin ja valmiita keinoja sanomien muunnoksiin. REST/JSON rajapinnat yleistyvät, vaikka XML on edelleen paljon käytössä, koska sanomien muunnokset onnistuvat sillä helposti. (Hautala 2018.) REST tulee sanoista Representional State Transfer, jonka avulla hajautetut sovellukset kommunikoivat http(s) protokollalla (Kivisaari 2016). Hautalan (2018) mukaan mikropalveluarkkitehtuuri myös integraatioalustoissa on yleistymässä SOA:n tilalle, mikä helpottaa pilvipalveluiden hyödyntämistä. Mikropalvelut ovat muista järjestelmistä täysin riippumattomia erittäin pieniä skaalautuvia liiketoimintafunktioita (Toivanen 2022).

API-hallinta ja API-talous liittyvät nykyään vahvasti integraatioihin. EAI ja B2B-integraatioiden lisäksi nykyään rajapintoja täytyy tarjota entistä enemmän ulospäin, esimerkiksi mobiilisovelluksille, mikä on lisännyt integraatioiden määrää ja haasteita. Näistä ulospäin tarjottavista palveluista puhutaan API-rajapintoina, joiden hallinta on tärkeää. API-hallinnan avulla API:t mm. saadaan dokumentoitua ja tarjottua turvallisesti sekä pystytään seuraamaan niiden käyttöä ja hoitamaan esimerkiksi autentikointia. (Hautala 2018.) Nykyään API-termillä tarkoitetaan usein nimenomaan REST/API rajapintaa. Mikropalveluiden suosio on myös lisännyt API-integraatioita. (Kivisaari 2016.)

Nykyään integraatioalustoja käytetään myös julkaisemaan niihin kytkettyjen järjestelmien tietoja OpenAPI -standardin määrittelemillä tavoilla. API:t voidaan toteuttaa esimerkiksi REST/JSON-rajapintana. Modernien integraatioalustojen avulla voidaan toteuttaa jopa kokonaisia applikaatioita. Asiakasjärjestelmä on yhteydessä integraatioalustaan, jonne luodaan API:a, jotka ovat yhteydessä taustajärjestelmiin. Integraatiojärjestelmä voi toimia myös API välityspalveluna ja portaalina.

Julkiset APIT mahdollistavat esimerkiksi uusia myyntikanavia. (Toivanen 2022.) Rajapintojen avulla voidaan myös luoda uutta liiketoimintaa (Kivisaari 2016.)

Tapahtumapohjainen tiedonvälitys on kasvattamassa suosioitaan. Jonon sijaan sanoma välitetään suoraan integraatioalustan REST/JSON rajapintoihin, josta ne välittyvät vastaanottavalle järjestelmälle. Kasvavaan kuormitukseen vastataan skaalautuvuudella esimerkiksi konttiteknologiaa hyödyntäen. (Toivanen 2022.) Tapahtumapohjaiset integraatiomallit perustuvat muutoksista tehtäviin tapahtumasanomiin, joita sovellukset voivat tilata. Tapahtumapohjaiset alustat jaetaan lokipohjaisiin, jonopohjaisiin ja tilauspohjaisiin alustoihin, joista lokipohjaisilla alustoilla, kuten Apache Kafka, voidaan tehdä laajimpia ratkaisuja. Tapahtumapohjaisilla integraatioilla tieto välittyy reaaliaikaisesti ja ne myös nopeuttavat sovelluksien kehitystä. (Källström 2022.) Niemisen (2020) mukaan niputtamalla yhteen joukko tapahtumia saadaan tapahtumavirta, event stream, jota voidaan analysoida reaaliaikaisesti, mikä mahdollistaa nopean reagoinnin esimerkiksi liiketoiminnan muutoksissa. Kafka tallentaa tapahtumat sanomalokiin, mikä mahdollistaa suurten tietomassojen välittämisen. (Nieminen 2020.)

IPaas-integraatioalustat, kuten Dell Boomi, ovat alkaneet hyödyntämään tekoälyä avustamaan integraatioiden teossa. Tämä perustuu pilvipohjaisen alustan sisältämään suureen integraatioihin liittyvään dataan, jota voidaan hyödyntää koneoppimisessa. Järjestelmä voi esimerkiksi ehdottaa uudelle integraatiolla sopivaa mallipohjaa ja olla hyödyksi datan muunnoksissa ja virhetilanteiden korjauksissa ja ennakoinnissa. Tulevaisuudessa tekoälyn avulla voidaan myös tunnistaa datasta erilaisia tietoja, kuten GDPR:n liittyviä henkilötietoja, joiden käsittelyyn liittyy erilaisia vaatimuksia. (Nieminen & Rannikko 2020.)

4.2.5 Integraatioiden suunnittelumallit

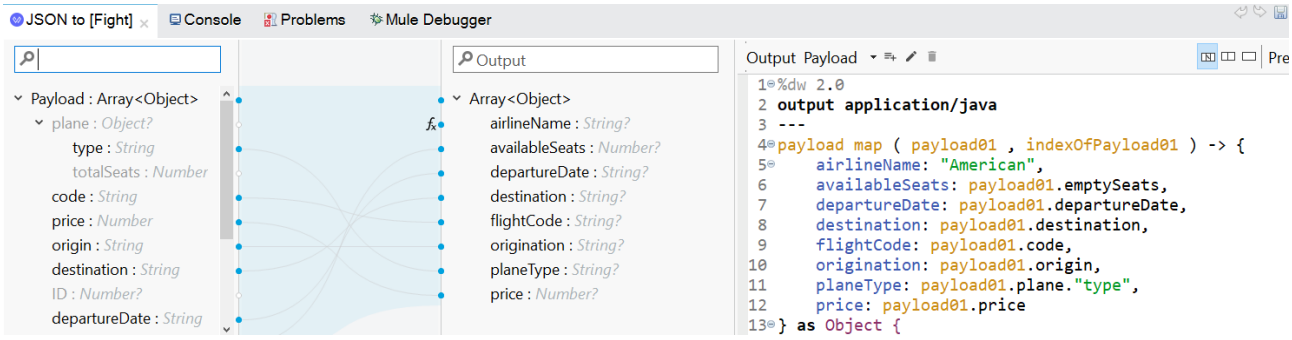
Integraatioita, kuten muitakin tietojärjestelmiä tehdessä tulee yleensä toistensa kaltaisia tapauksia vastaan, jolloin jo aiemmin tehtyjä, hyväksi havaittuja dokumentoituja malleja voidaan käyttää uudelleen (Tähtinen 2005, 89). Näitä integraatiomalleja on olemassa kymmenittäin, ja mm. Hohpe ja Woolf ovat kuvanneet erilaisia sanomapohjaisia integraatioiden suunnittelumalleja kirjassaan Enterprise Integration Patterns (2003) sekä verkkosivuillaan enterpriseintegrationpatterns.com. Kirjassa integraatiomallit on jaoteltu eri kategorioihin. Integration Styles esittelee malleja eri integraatiotavoille. Muut kategoriat keskittyvät sanomanvälityksen osa-alueiden malleihin. Channel

Patterns esittelee sanomien välityksen malleja, Message Construction Patterns sanoman rakenteen malleja, Routing Patterns sanoman reitytyksen malleja ja Transformation Patterns sanomien muunnoksien malleja. Endpoint Patterns esittelee kuinka osapuolet tuottavat tai tilaavat sanomia, ja System Management Patterns mallit kuvaavat kuinka sanomapohjaisen järjestelmän ylläpitoa voidaan hoitaa. (Tarkoma 2012, 102–103; Hohpe & Woolf 2003a.) Källström (n.d.) toteaa, että vaikka Hohpen ja Woolfin mallit ovat edelleen osittain valideja, uudempiin API- ja streaming-pohjaisiin integraatioihin niiden sopivuutta voi kyseenalaistaa.

Digialla yleisimmät integraatioiden suunnittelumallit ovat Managed File Transfer, joka kuvaa tiedostonsiirtoa integraatioalustan avulla, Event Sourcing on malli tapahtumien seuraamiseen pohjautuvaan integraatioon, Managed API puolestaan on malli API:en julkaisemiseen API-hallintatyökalun avulla. Intelligent Messaging on malli sanomien välitykseen MOM:n välityksellä, ETL Interactions on malli datan välittämiselle tietokantojen välillä esimerkiksi data warehousing -tapahtuman yhteydessä. Change Data Capture mallia käytetään usein yhdessa Event Sourcing mallin kanssa huomaamaan muutoksia datassa. API-Based Data Integration malli mahdollistaa datan integraation API:en avulla. (Källström n.d.)

4.3 Informaation eri muodot

Integroitavien sovellusten käyttämä ja jakama informaatio on harvoin valmiiksi oikeassa muodossa toisen sovelluksen käyttöön. Samakin tieto voidaan esittää lukuisilla eri tavoilla, ja monesti tietoa joudutaan muuttamaan eri formaattiin (Ks. kuvio 13) ja mahdollisesti täydentämään sitä tai yhdistämään eri lähteistä saatua tietoa. Tiedon sisältö saattaa myös vaikuttaa integraatiojärjestelmän toimintaan, esimerkiksi saada sen lähettämään jonkinlaisen hälytyksen tietynlaisen informaation perusteella. (Tähtinen 2005, 56–58.)



Kuvio 13. JSON-datan muuttaminen objektiksi MuleSoftissa.

Mikäli kaikkien integroitavien järjestelmien käyttämä informaatio on erilaista, joudutaan tieto-muunnoksia tekemään jokaisen liittymän välillä, jolloin keskitetyn integraatioratkaisun hyödyt menetetään. Huonoimmassa tapauksessa muunnosten määrä lasketaan samalla kaavalla $\frac{n(n-1)}{2}$ kuin point-to-point integraatioiden määrän muuttujan n ollessa liittymien määrä. Integraatiossa saatetaankin käyttää tiedon kuvaukseen erilaisia välimalleja, johon eri järjestelmien käyttämä informaatio muunnetaan. Tällöin muunnosten määrä vähenee huomattavasti ollen suurimmillaan $2n$. (Tähtinen 2005, 84–87.) Tätä välimallia kutsutaan kanoniseksi tietomalliksi, joka on järjestelmistä riippumaton (Hohpe & Woolf 2003b).

Vanhin mutta hyvin toimiva tapa välittää dataa on paikkasidonnaiset datan muodot, joissa tietyn tiedon kertova osio datasta on aina saman mittainen, ja mikäli varsinainen data on lyhyempi tai se puuttuu, täytetään tyhjä osa esimerkiksi tyhjillä lyönneillä tai nollilla. Toinen yleinen tapa on pitää datan eri osat aina samassa järjestyksessä ja erottaa ne sovitulla erottimella, esimerkiksi pilkulla. Data voi olla jäsennelty myös avainsanojen avulla, jolloin eri osien järjestys on vapaampi. Puumaiset tietorakenteet yhdistelevät kaikkia edellä mainittuja formaatteja. (Manouvrier & Menard 2008, 53–54.) Yleinen sovitulla erottimella eroteltu datan muoto on CSV (Comma Separated Values) jossa pilkku toimii erottimena.

EDI (Electronic Data Interchange) tarkoittaa UN/EDIFACT-standardin mukaista tiedonvaihtoa. YK:n talouskomission kehittämä standardi on vuodelta 1988, jolloin sanomien siirto oli kallista, ja sen vuoksi EDI esittää tietoja hyvin tiiviisti. Lyhenne tulee sanoista Electronic Data Interchange for Administration, Commerce and Transport. (Hirvonen & Laaksonen n.d.) EDIFACT-sanoma koostuu erilaisista segmenteistä, joissa tietoa välitetään. Suomessa on lisäksi käytössä ED2-sanomakehys,

jonka sisällä voidaan välittää muitakin kuin EDI-tyypin sanomia (Hirvonen & Laaksonen). Suomessa EDI:n ohella käytetään termiä OVT, joka tulee sanoista organisaatioiden välinen tiedonsiirto. EDI on melko huonosti muunneltavissa ja pienille organisaatioille melko kallis, mikä voi jarruttaa käyttöönottoa (Manouvrier & Menard 2008, 114).

XML (eXtensible Markup Language) on melko yleinen dataformaatti integraatioiden yhteydessä. XML on rakenteeltaan yksinkertainen ja laajennettava, ja sen data sisältää aina metadataa, koska data rakentuu erilaisien kuvauskenttien varaan, kuten esimerkiksi <email>testi@testaaja.fi</email>. Näitä kenttiä voi olla sisäkkäin useita ja niillä voidaan esittää listoja tai puumaisia tietorakenteita, joita on helppo lukea. Yksittäisiin kenttiin voidaan lisätä tageja eli tunnisteita, jotka lisäävät tietoa eri kentistä. XML Schemat ovat eräänlaisia dokumenttimalleja, joissa elementtien ominaisuuksia määritellään tarkemmin. XSLT (Extensible Stylesheet Language Transformations) on kieli, jolla XML-dokumentteja pystytään muuntamaan. XML on melko raskas dataformaatti kaikkine kuvauskenttineen ja dokumentin koko kasvaa helposti suureksi. (Tähtinen 2005, 83, 127–133.)

XML:ää kevyempi datan välitysmuoto on JSON (JavaScript Object Notation). Se on suosittu tekstipohjainen, ohjelmointikielestä riippumaton datan välitysmuoto. Sitä on helppo ymmärtää ja toisaalta sitä on helppo käsitellä ja luoda koneellisesti. Data säilötään avain – arvo -pareina, jotka erotellaan pilkulla ja eri objektit aaltosuluilla, esimerkiksi {”nimi”:”Matti”, ”Sukunimi”:”Meikäläinen”}. Hakasulkeiden sisään voidaan listata useita avain – arvo -pareja tai objekteja pilkulla eroteltuna. (JSON, n.d.)

4.4 Järjestelmäintegraatioiden testaaminen

Kuten ohjelmistoja, myös integraatioita on testattava kehitettäessä uusia tai korjattaessa vanhaa. Tähtisen (2005, 158) mukaan integraation toimivuus on syytä varmistaa ennen tuotantoon viemistä testiympäristössä, joka on mahdollisimman samankaltainen kuin tuotantoympäristö. Dolbyn (2023) mukaan integraatiot ovat tekemisissä useiden eri ulkoisten palveluiden kanssa, ja testatessa integraatioita on näiden palveluiden oltava saatavilla. Integraatioiden testaamien on haastavaa juuri näiden ulkoisten osapuolten takia, jotka voivat olla eri tekniikalla tehtyjä ja osittain jo vanhentuneita, mikä tekee testitapausten ja tulosten määrittämisestä vaikeaa. Näihin ulkoisiin järjestelmiin voi olla myös hankala päästä käsiksi testaamista varten, jos ne ovat ulkopuolisen tahon

hallussa. Myös testiympäristön luominen identtiseksi tuotantoympäristön kanssa on erittäin haastavaa tai jopa mahdotonta. (Hohpe & Istvanick 2002 6–7.)

Sanomanvälitykseen pohjautuvan integraation asynkroninen luonne ja monien prosessien riippuvuus aikamäärittelyistä, kuten erilaisista ajastuksista, aiheuttaa testaamiselle lisää haastetta. Integraatioissa pienin testattava yksikkö on yleensä laajempi kuin perinteisessä sovelluskehityksessä ja monesti hyvin riippuvainen erilaisesta datasta. Siitä huolimatta myös integraatioita olisi hyvä testata kerroksittain, aloittaen yksittäisistä komponenteista, kuten adaptereista, ja testaamalla sen jälkeen laajempia kokonaisuuksia. (Hohpe & Istvanick 2002, 7.)

Integraatioita testatessa tarvitaan yleensä testidataa, etenkin kun testataan transformaatioita. Jonkinlainen automaattinen testidatageneraattori voi olla hyödyllinen. Muuten transformaatioiden testaaminen on ehkä integraatiotestaamisen helpoin osa-alue. Muunnoksen jälkeistä dataa verrataan toivottuun tulokseen ja tehdään tarvittavat johtopäätökset. Sen sijaan liiketoimintaprosessien testaamien on haastavaa ja vaatii yleensä useita eri testejä ja ulkopuolisia palveluja jäljittteleviä niin sanottuja tynkiä (stub). (Hohpe & Istvanick 2002, 9–10.)

Dolbyn (2023) mukaan yleensä yksikkötestejä, eli pieniä osia integraatiosta, voidaan testata ilman ulkoisia osapuolia tai tekemällä ulkoista osapuolta jäljittelevä komponentti avuksi. Yksikkötestaamisen tulisi olla mahdollista kehittäjän omalla koneella. Koko tietovirran (flow) testaamien alusta loppuun sen sijaan vaatii yleensä myös ulkopuoliset osapuolet saataville. Mikäli ulkopuolisissa osapuolissa tapahtuu muutoksia, on myös jäljittelevä komponentti päivitettävä. Hän esittelee tämän ongelman ratkaisuun komponenttitestin, joka muistuttaa yksikkötestiä mutta voi ottaa yhteyden ulkopuolisiin järjestelmiin. (Dolby 2023.)

Integraatioiden ei-toiminnalliset ominaisuudet, kuten suorituskyky ja skaalautuvuus, ovat monesti hyvin pitkälti riippuvaisia käytetystä integraatioalustasta. Niitä voidaan testata luomalla automaattisesti testimateriaalia ja kuormittamalla järjestelmää. Virheistä toipumista voidaan testata luomalla erilaisia virhetilanteita tai muokkaamalla dataa virheelliseksi. Integraation testattavuus ja ylläpidettävyys puolestaan riippuvat paljon arkkitehtuurista. Huolellinen suunnittelu voikin helpottaa muuten melko haastavaa integraatioiden toiminnallista testaamista. (Hohpe & Istvanick 2002, 13–15.)

Jotkut integraatioalustat, kuten IBM App Connect Enterprise (ACE), tarjoavat mahdollisuuden yksikkötestien tekoon avustetusti tai automaattisesti. ACE:n avulla voi luoda testin esimerkiksi yksittäiselle tapahtumalle, kuten muunnokselle, tai kokonaiselle flow:lle. Lisäksi integraation flown voi nauhoittaa, jonka pohjalta ACE luo automaattisesti testitapauksen, ja nauhoitetun flown pohjalta testin voi ajaa myös, vaikka joku ulkoinen osapuoli ei olisi käytettävissä. (Developing integration tests, 2023.)

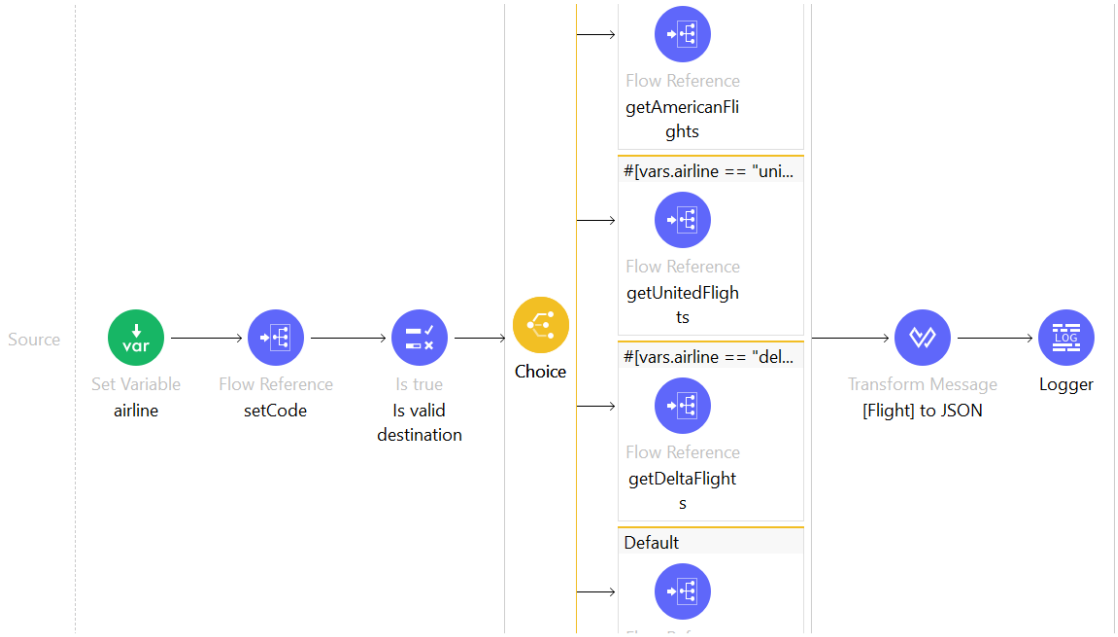
Alustojen omien testimahdollisuuksien lisäksi esimerkiksi automaatiotestaukseen kehitettyä Robot Frameworkia voidaan käyttää myös integraatioiden testaamiseen. Dolby (2023) kertoo, että automaattiset testit ajetaan myös integraatioratkaisujen yhteydessä aina osana julkaisuputkea (pipeline), kun muutoksia tai uusia osia on tehty. Näihin kuuluu sekä yksikkö- että komponenttitestit ja laajemmat, koko integraation testitapaukset. Näiden jälkeen voidaan vielä tehdä manuaalisia testejä. Ylipäätään integraatioiden testaaminen on monesti hitaampaa kuin sovellustestaus, ja testissä paljastuneiden virheiden syitä voi olla vaikeampi löytää. (Dolby 2023.)

4.5 Yleisimpiä integraatioalustoja

Integraatioalustat ovat erilaisia ja niiden toiminnallisuudet ja kyvyt ovat erilaisia. Annenko (2022) toteaakin, että tutkimus- ja konsulttiyritys Gartner on useita vuosia korostanut, ettei mikään yksittäinen markkinoilla oleva integraatiotuote täytä kaikkia integraatioiden tarpeita. Alusta kuitenkin toimii integraatioiden hallintapisteenä, jolloin alustoja käytettäessä puhutaan keskitetystä integraatiosta. Osa alustoista vaatii koodaamista, osassa integraatioita voidaan tehdä visuaalisesti mallintamalla. Koodaamiseen perustuvia alustoja kutsutaan full-code-alustoiksi ja ei ollenkaan koodaamista vaativia alustoja no-code-alustoiksi. Väliin mahtuu vielä low-code-alustat. No-code-alustat toimivat sellaisenaan melko harvoin, ja full-code alustat vaativat paljon osaamista. Low-code-alustat ovatkin integraatioissa hyvä vaihtoehto. (Toivanen 2022.)

Erilaisia integraatioalustoja on lukuisia. Gartner on listannut suosituimpia integraatioalustoja vuodelta 2022. Listan kärjessä ovat mm. InterSystemsin Ensemble, Zapier, Software AG:n webMethods Integration Platform, WSO2 API Manager, SEEBURGER BIS, MuleSoft Anypoint Platform sekä Workato (Enterprise Application Integration Platforms Reviews and Ratings 2023). Digia Oyj:n eniten käyttämiä iSuiten ohella ovat mm. (Dell) Boomi, Azure Integration Services, Camel Open Source Stack, Red Hat Integration/Fuse, Mulesoft sekä IBM App Connect Enterprise (Integration

technology categorization, n.d.). Näistä monet perustuvat graafiseen käyttöliittymään ollen low code -alustoja kuten kuviossa 13 näkyvä Mulesoft.

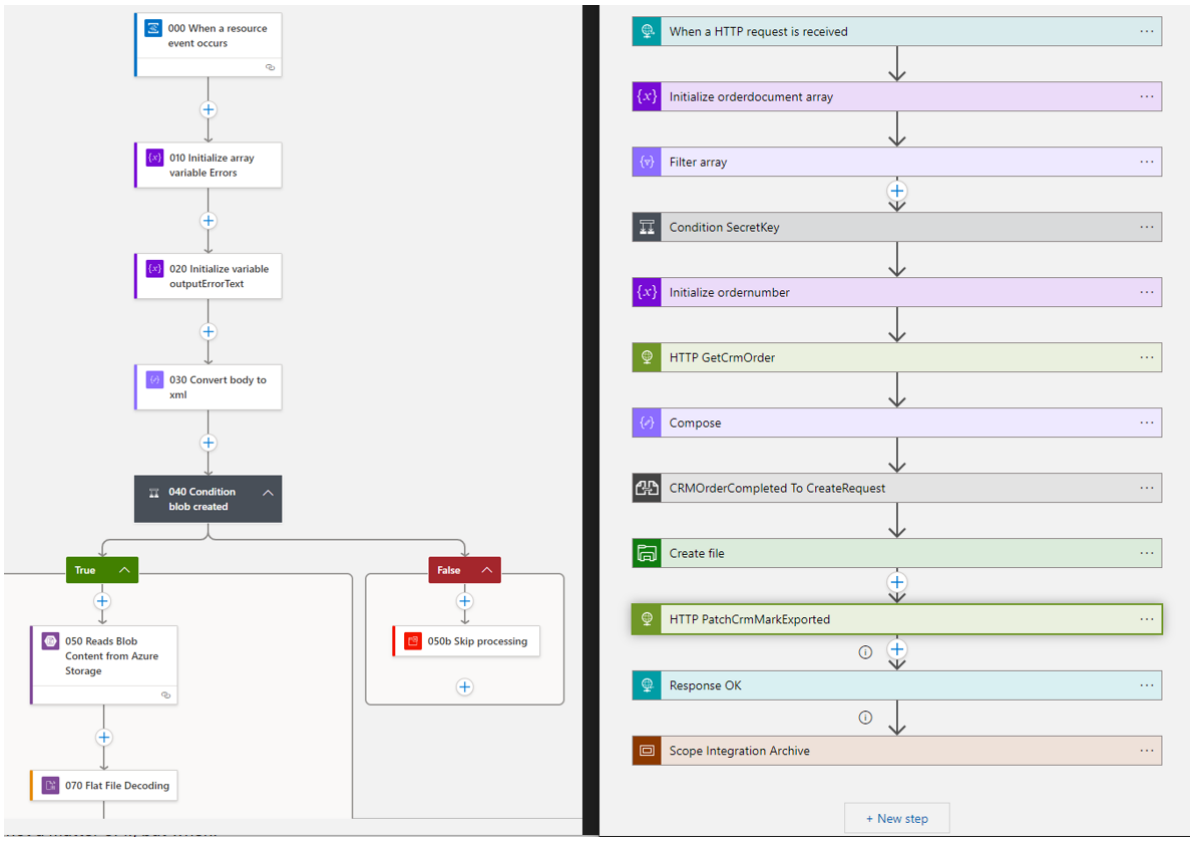


Kuvio 14. MuleSoftin AnypointStudion flow-näkymä.

Avoimen lähdekoodin integraatioratkaisut ovat myös suosittuja. Esimerkiksi aiemmin mainittu Apache Kafka on avointa lähdekoodia. Myös Hohpen ja Woolfin teokseen Enterprise Integration Patterns perustuva Apache Camel on suosittu avoimen lähdekoodin integraatioalusta, joka tosin vaatii koodamistaitoja, kuten monet muutkin avoimen lähdekoodin ratkaisut. Camelilla onnistuu Hohpen ja Woolfin mallien lisäksi uudemmat mikropalveluihin pohjautuvat integraatiot. (Källström 2022; Apache Camel n.d.)

Integraatioita voidaan nykyään toteuttaa myös iPaas-alustoilla, joita ajetaan pilvessä. Ne ovatkin kasvattaneet suosiotaan pilvipalveluiden yleistyessä ja integraatioiden siirtyessä pilveen. Pilvipalveluiden tarjoajat, kuten Azure (kuvio 15) ja AWS, tarjoavat usein myös integraatioalustan. Myös tapahtumapohjaiset integraatiot, jotka usein pohjautuvat Kafkaan, ovat suosittuja. (Källström, 2022.) Osalla iPaas alustoista on mahdollista orkestroida prosesseja, joka automatisoi niiden kulun. (Toivanen 2022). EiPaas tarkoittaa enterprise iPaas-alustaa, jollainen on esimerkiksi kotimainen Frends- integraatioalusta. Nimensä eiPaas mukaisesti se soveltuu hyvin enterprise -luokan integraatioihin ollen erittäin monipuolinen ratkaisu. (What is an iPaas n.d.)

Toivanen (2022) vertailee pilvialustojen integraatiotoiminnallisuuksia ja varsinaisia integraatioalustoja ja toteaa mm., että varsinaisella integraatioalustalla monitorointikyvyt ovat paremmat, eikä koodaamistaitoja tarvita niin paljoa, ja pilvialustoilla puolestaan on monesti käytössä datan varastointiin ja koneoppimiseen liittyviä ominaisuuksia. Lisäksi hän korostaa, että ylläpidon aikaisten kustannusten matalana pitämiseksi on integraatioalustan monitorointi- ja operointimahdollisuuksien oltava hyvät. (Toivanen 2022.)



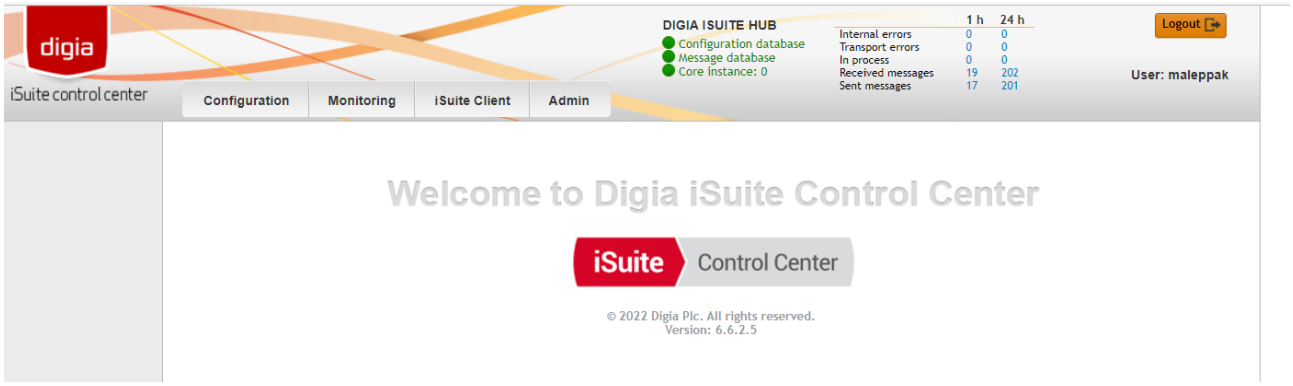
Kuvio 15. Azuren Standard LogicApps ja Consumption LogicApps.

Toivasen (2022) mielestä nykyaikaisen integraatioalustan tulee mm. kyetä API-hallintaan ja suorittamiseen sekä API:en luontiin visuaalisella notaatiolla, orkestrointiin joko nauhoituksiin tai koneoppimiseen perustuen ja tukea useita erilaisia sanomamuotoja. Lisäksi toiminen hybridinä sekä pilvessä että omassa konesalissa, skaalautuvuus ja hyvä valikoima valmiita adaptereita ovat tärkeitä. (Toivanen 2022.)

5 Integraatioalusta Digia iSuite

5.1 Digia iSuiten kuvaus

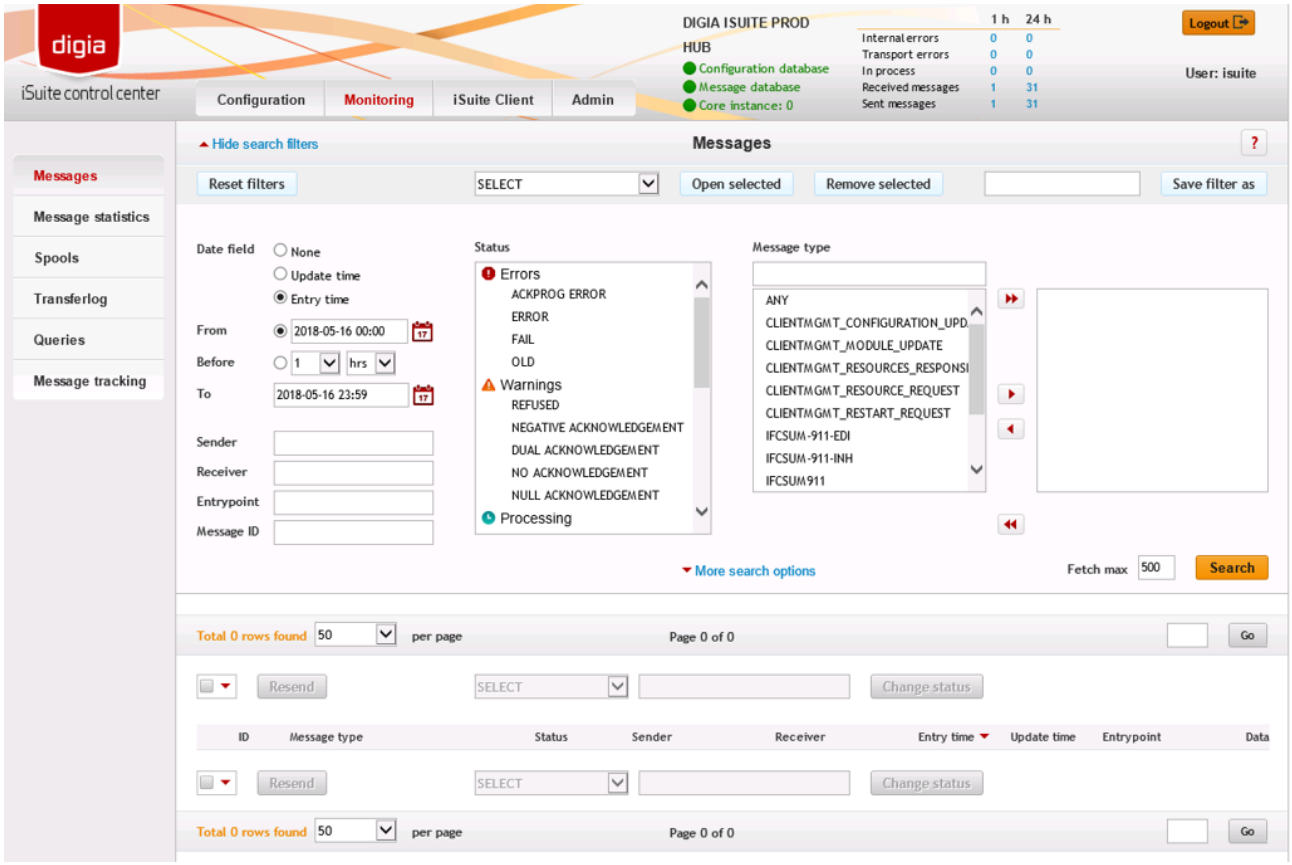
Digia iSuite on avoimiin standardeihin perustuva alustariippumaton Java-pohjainen integraatioalusta, jonka ideana on, että integraatiot tulee voida tehdä konfiguroimalla eikä koodaamista tarvita. Helppokäyttöinen iSuite soveltuu yrityksen sisäisiin sekä liiketoimintakumppanien kanssa toteutettaviin integraatioihin ja on räätälöitävissä asiakaskohtaisesti. iSuite HUB sisältää käyttöliittymän (kuvio 16) liittymien konfigurointiin ja sanomien seurataan sekä sanomien arkistoinnin. Valmiina tarjolla on myös valvonta- ja raportointimahdollisuuksia sekä kuittauskäsittelyt, hälytykset ja uudelleenlähetykset. (Hirvonen 2021; Digia iSuite tuotekuvaus n.d.) ISuite eroaa monesta muusta integraatiotuotteesta siten, että se on enemmänkin konfiguraatiotyökalu, joka toimii erittäin hyvin integraatioiden tekemisessä, kuin varsinainen integraatioalusta (Rantanen 2023). Sillä ei ole graafista käyttöliittymää kuten monilla muilla integraatioalustoilla vaan kaikki integraatioiden asetukset tehdään määrittämällä konfiguraatioita selainkäyttöliittymän kautta.



Kuvio 16. iSuite 6:n käyttöliittymän aloitusnäkymä.

iSuitella onnistuu reaaliaikainen ja tapahtumapohjainen sanomien välitys. Sitä voidaan käyttää myös mobiili- että IoT-palveluiden tiedon välittämiseen. iSuitella on tuki yleisimmille viestinvälitystavoille ja se on päätelaiteriippumaton sekä hyvin integroitavissa taustajärjestelmiin. iSuite on myös riippumaton tiedon rakenteesta, verkkoarkkitehtuurista sekä käyttöjärjestelmästä. (Digia iSuite tuotekuvaus n.d.) iSuitella on sisäänrakennettuja muunnosmoduuleja esimerkiksi XML- ja EDI muunnoksiin sekä valmius käsitellä paikkasidonnaisia tai tietyllä erottimella jäsenneltyjä datan

muotoja ja myös mahdollisuus lisätä räätälöityjä muuntimia. Sanomat voidaan salata ennen niiden näkymistä monitoroinnissa (kuvio 17). (TechIntro – Digia iSuite 2023.)

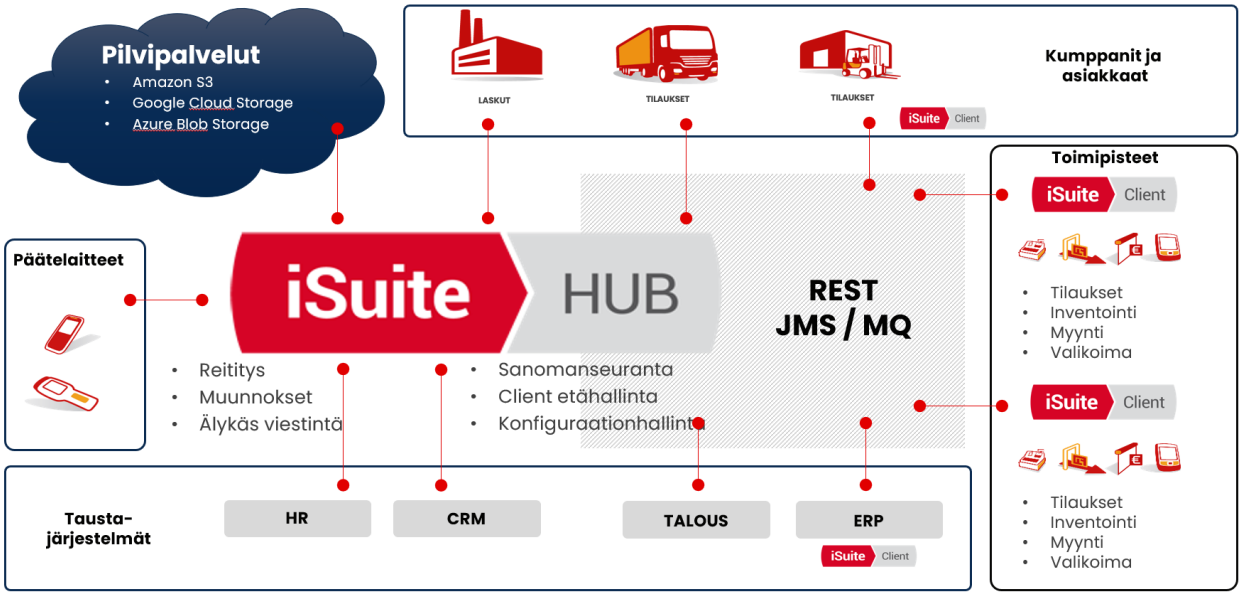


Kuvio 17. iSuite 6:n monitorointinäkymä (Hirvonen 2021).

iSuitella voidaan hallinnoida suurta määrää tietoa erilaisissa olosuhteissa ja sovellusympäristöissä. Sen keskeisiä sovellusalueita ovat esimerkiksi yrityksen järjestelmien integraatiot, B2B-integraatiot, langattomat integraatiot sekä mikropalveluiden väliset pilvi-integraatiot. iSuiten vahvuuksia ovat skaalautuvuus, toimintavarmuus, modulaarisuus ja ylläpidettävyys. iSuitella voidaan toteuttaa keskitetty integraatio tai hajautettu, useamman kontitetun iSuite-instanssin muodostama integraatio joko yrityksen omilla palvelimilla tai ulkopuolisen palveluntarjoajan pilvialustalla. (Digia iSuite tuotekuvaus n.d.) iSuite soveltuu toimivaksi yksinään tai täydentämään integraatioratkaisua esimerkiksi sanomanseurannan ja monitoroinnin avulla (TechIntro – Digia iSuite 2023).

iSuite toimii yleensä hub-and-spoke-mallisesti mutta täyttää myös ESB-määritelmän. iSuiten referenssiarkkitehtuuri on esitelty kuviossa 18. Digia iSuite Client on tiedonsiirtomoduli, agentti, jota

käytetään hajautettujen järjestelmien väliseen yhteyteen. Sen avulla tiedon prosessointi voidaan hajauttaa ja näin jakaa kuormitusta asentamalla Client etäpalvelimelle. Se toimii alijärjestelmän ja HUB:in välisenä kommunikaation ja sanomien välittäjänä. (Digia iSuite tuotekuvaus n.d.) Uusin versio iSuitesta on versio 6.6.2.5.1. Versio lienee viimeinen ennen iSuite 7:n julkaisua mahdollisesti 2023 (Rantanen 2023).

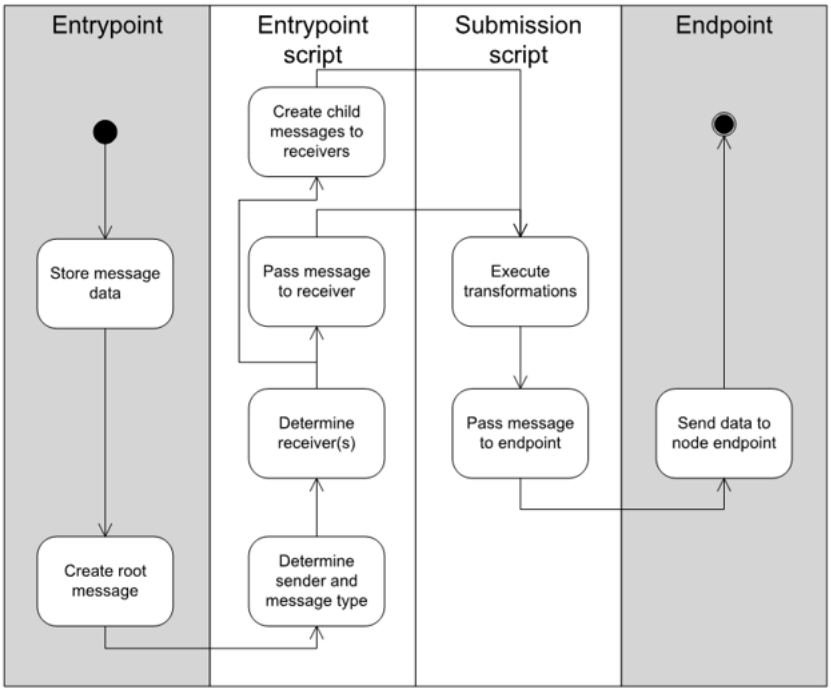


Kuvio 18. Digia iSuiten referenssiarkkitehtuuri (TechIntro - Digia iSuite 2023).

5.2 Digia iSuiten toimintaperiaate

Kaikkien komponenttien konfigurointi tapahtuu käyttöliittymän (Digia iSuite Control Center, ICC) kautta, josta tapahtuu myös sanomien seuranta. Konfiguraatioista säädetään esimerkiksi sanomien kulku, sanomien käsittely ja muunnokset. Monitorointi-osiosta nähdään sanoman status, lokit, mahdolliset virheet ja statistiikkaa. ICC kommunikoi HUBin kanssa käyttäen RESTiä. HUB puolestaan on yhteydessä iSuiten JDBC-tietokantaan, jonne konfiguraatiot ja sanomat tallennetaan JDBC-konnektorin avulla. iSuitella on mahdollista käyttää monenlaisia eri kommunikaatiometodeja, kuten HTTP(S), SFTP, FTP(S), pilvikonnektoreja sekä erilaisia sanomajonoja ja tietokantayhteyksiä. (TechIntro – Digia iSuite 2023.)

Luotaessa uutta liittymää iSuitelle tulee konfiguroida erilaisia komponentteja, kuten esimerkiksi sekä lähettävä että vastaanottava node. Näiden avulla sanoma saadaan reititettyä, ja monitoroinnissa nähdään heti mistä ja minne sanoma on menossa. Vastaanottavalla nodella voi lisäksi olla sanomatyypistä riippuvia prosessointeja sanomalle ja yleensä useampia node endpointeja, jotka määrittävät sanomatyypistä riippuen tarkasti, minne ja miten sanoma toimitetaan. Node endpointeja voi olla erilaisia, kuten esimerkiksi kansio (dir) tai SFTP endpointeja. Node endpoint siis hoitaa sanoman uloskirjoituksen. Kuviossa 19 näkyy sanoman kulku iSuitella. (Hirvonen 2021; System Overview 2023.)

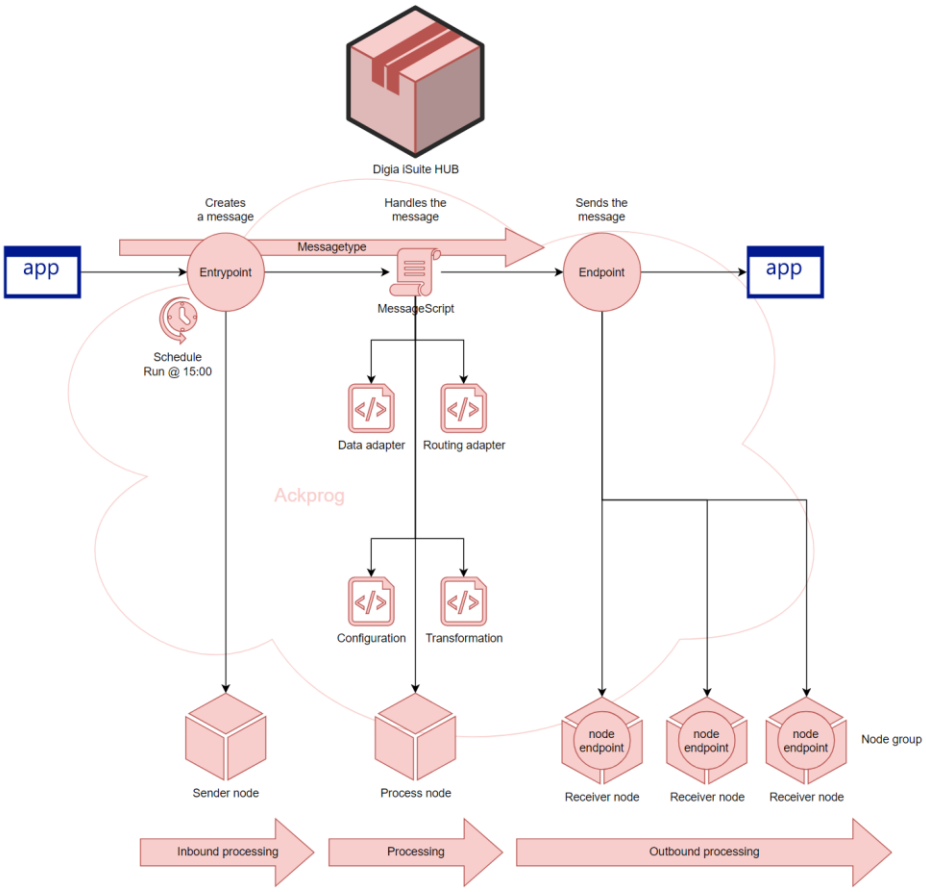


Kuvio 19. Sanoman kulku iSuitella (Hirvonen 2021).

Entrypoint hoitaa sanoman sisään luvun. Kaikkien liittymien ensisijaiset sanomat saavat alkunsa entrypointilla. Entrypointeja on erityyppisiä, kuten esimerkiksi kansio-entrypoint (dir) tai SFTP entrypoint. Data-adapteri nimeltä Entrypointscript poimii sanoman datasta tiettyjä attribuutteja lisäten ne sanoman attribuuteiksi. Lisäksi jokaiselle sanomalle tarvitaan messagetype määrittämään sanoma. (Hirvonen 2021; System Overview 2023.)

Reititysadapteri (submission script) hoitaa sanoman reitityksen konfiguraatioiden ja sanoman datan perusteella. Se myös tunnistaa sanomatyypin perusteella mitä transformaatioita sanomalle on

tehtävä. Messagescriptit hoitavat sanomien käsittelyn. Messagescript tunnistaa sanoman osapuolet, sanoman tyypin ja luo tarvittaessa lapsiviestin. Schedule-komponentilla voidaan luoda ajastuksia entrypointeille ja endpointeille. Ackprogien avulla voidaan sanoman statuksen muutoksien perusteella käynnistää erilaisia prosesseja, kuten hälytyksiä tai muita toimenpiteitä. Propertiesit ovat globaaleja attribuutteja, joita voi käyttää entrypointilla, endpointilla sekä message scripteillä. (Hirvonen 2021; System Overview 2023.) Kuviossa 19 näkyy iSuiten toimintaperiaate.

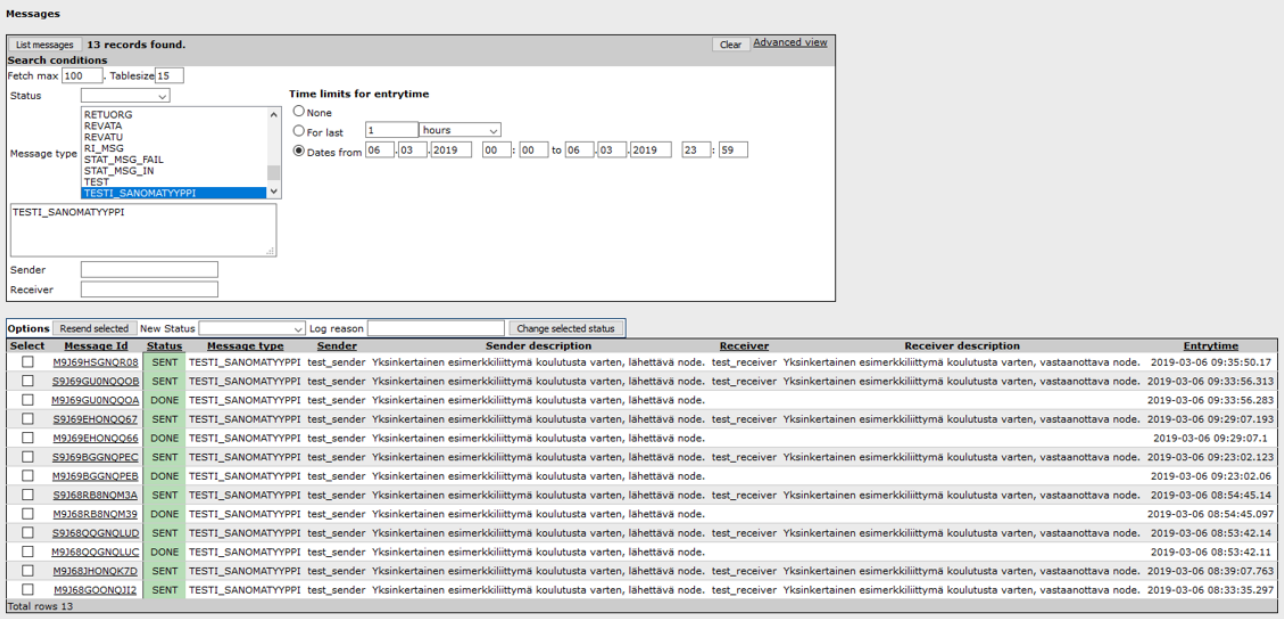


Kuvio 20. iSuiten toimintaperiaate (TechIntro - Digia iSuite 2023).

5.3 Digia iSuite 5 ja Digia iSuite 6

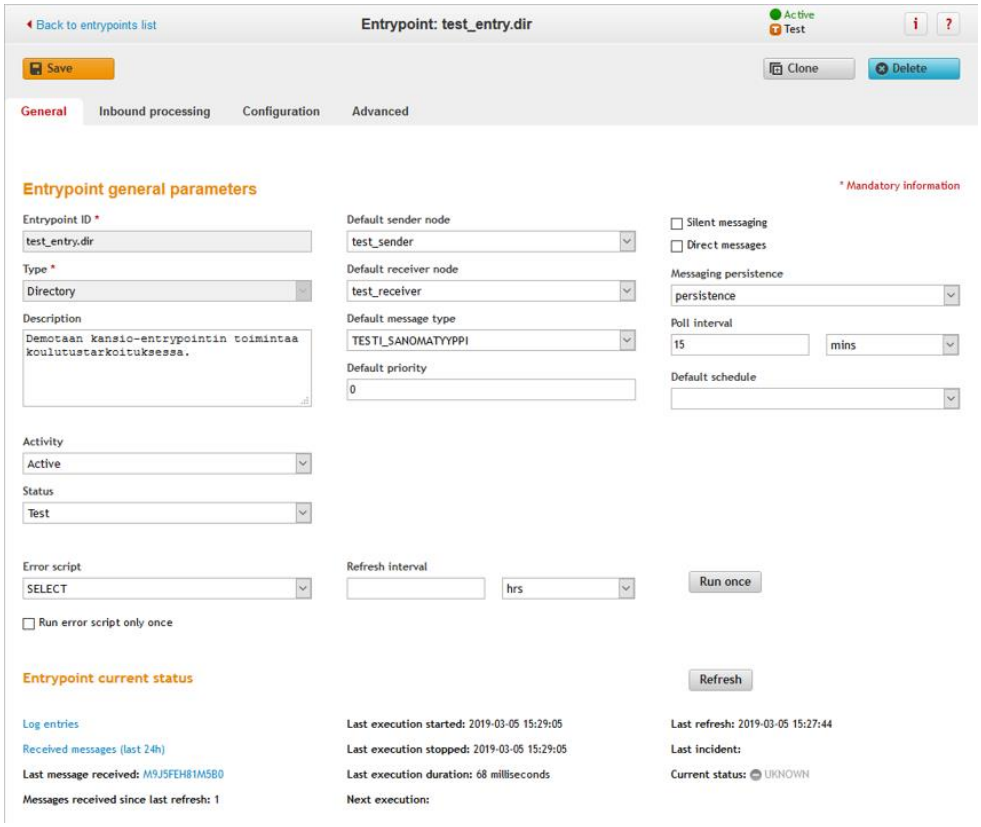
iSuite 5:n ensimmäinen versio julkaistiin vuonna 2007 ja iSuite 6 vuonna 2013 (Rantanen 2023). Etenkin käyttöliittymä on uusiutunut iSuite 6:lle paljon. Käyttöliittymän kautta iSuite 6:lla pystyy määrittämään enemmän konfiguraatioasetuksia. Oletuksena iSuite 6:lla sanomaseurannassa on kattavampi näkymä sanoman tietoihin ja sanomien seurannasta voi siirtyä linkin kautta suoraan

komponentin asetuksiin. (Hirvonen 2021.) Kuviossa 21 näkyy iSuite 5:n sanomaseurannan messages-näkymä, jota voi verrata kuvion 17 iSuite 6:n vastaavaan.

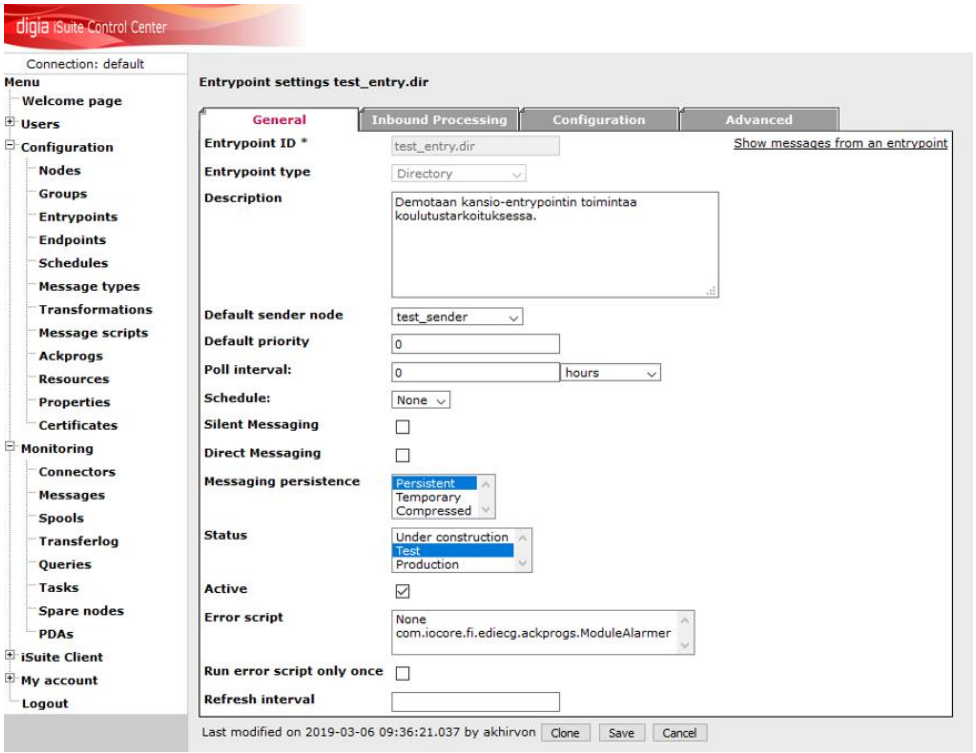


Kuvio 21. iSuite 5:n messages-näkymä (Hirvonen 2021).

iSuite HUB 6.6 toimii Java 11 alustalla. Yhteydet ovat tietoturvan kannalta kehittyneet, ja liitynnät eri pilvipalveluihin on lisätty valikoimaan. iSuite 6:lla progress-yhteysasetuksien määrittäminen onnistuu käyttöliittymän kautta. Myös muita uusia konfigurointi- ja monitorointirajapintoja on lisätty. (Hirvonen 2021.) ISuite6:ssa on myös high code-ominaisuuksia, jolloin käyttäjät voivat lisätä itse tekemiään komponentteja siihen. Modulaarisuus toteutuu myös iSuiten tuotekehityksessä ja eri komponenttien käytössä integraatioita tehtäessä. Lisäksi iSuite 6:ta löytyy resurssimanageri ja työjonomanageri. (Rantanen 2023.) Kuviossa 22 näkyy iSuite 6:n entrypointin general-välilehti, jossa määritellään mm. oletus vastaanottaja, sanomatyyppi ja entrypointin tyyppi. Kuviossa 23 on puolestaan iSuite 5:n entrypointin general-välilehti.



Kuvio 22. Isuite 6:n entrypointin general-välilehti (Hirvonen 2021).



Kuvio 23. iSuite 5:n entrypointin general-välilehti (Hirvonen 2021).

iSuite 5 on käymässä monella tapaa vanhaksi. Asiakkaita, joilla sen on vielä käytössä, rohkaistaan päivittämään ja siirtämään integraatiot uudempaan versioon, koska kahden version ylläpito ei ole järkevää. Lisäksi iSuite 5:n tuotekehittäminen ja toiminta on ei-modulaarista ja se käyttää Java 5:ttä, joka on erittäin vanha verrattuna iSuite 6:n Java 11:ta. Tämä aiheuttaa useita ongelmia tietoturvan kannalta, mm. vanhentuneita http-clienteja ja turvattomia chipehreitä eli tiedonsalausalgoritmeja, sekä muita hakkereiden hyödynnettäviä aukkoja tietoturvassa. Esimerkiksi tietoturvallisen HTTPS-yhteyden muodostuksessa pitäisi olla vähintään TLS 1.2 tason kättely, joka löytyy vasta Java 8:ssa. iSuite 5 on kehitetty Oraclen maksullisella Javalla, kun taas iSuite 6 on kehitetty OpenJDK:lla, joka on avoimen lähdekoodin implementaatio Javasta. Vaikka Javan ajoympäristö on taaksepäin yhteensopiva, loppuu yhteensopivuus jossain vaiheessa, mikä tarkoittaisi suuria muutoksia vanhaan järjestelmään. (Rantanen 2023.)

6 Toteutus

6.1 Lähtötilanne

Digian iCare-sanomavälityskeskus käsittää sekä iSuite 5:n, josta käytetään nimeä AMQ, sekä iSuite 6:n (WMQ). Nimet juontavat aikaan, jolloin iSuite 5:llä olivat ActiveMQ-jonoja käyttävät liittymät ja iSuite 6:lla Websphere MQ -yheydet (Laaksonen 2022). iSuitet toimivat Red Hat Enterprise Linux Serverillä ja MS SQL tietokanta Microsoft Windows Server 2008:lla. Samalla palvelimella tuotannon versioiden kanssa toimivat myös molempien iSuite:n testiversiot. iCarella liittymät ovat sanomapohjaisia, ja integraatiot on toteutettu hub-and-spoke-mallisesti liittymien määrän ollessa hallittavissa yhdellä hubilla.

iCaren AMQ:n tietokanta oli siirtotyötä aloittaessa jo kopioitu ja siirretty WMQ:lle mutta jätetty kaikki kopioidut sanoman välityksiä käynnistävät entrypointit inaktiivisiksi, jotta varsinainen sanomaliikenne ei siirtyisi ennen kuin liittymät olisi testattu. Näin ollen suurin osa tehtävästä työstä oli näiden kannasta kopioitujen liittymien konfiguraatioiden tarkastamista, muuttamista sekä testaamista. Kaikki konfigurointityöt tehtiin selaimen käyttöliittymän kautta, josta pystyi myös tekemään kyselyjä tietokantaan.

Suurin osa Asiakkaan kumppaneista sai joko oman tai asiakas.dir -entrypointin kautta tilauksia, jotka iSuite osaa erottaa toisistaan tiedoston nimen ja sanoman sisällön perusteella. Osalla kumppaneita oli lisäksi oma entrypoint laskujen lähettämiseksi Asiakkaalle. Entrypointille on yleensä määritelty oletuksena lähettäjä, ja entrpointscript tunnistaa sanoman sisällöstä lähettäjän sekä vastaanottajan. Etenkin tapauksissa, joissa sanoman sisältöä muokataan, iSuite luo yleensä pääviestille lapsiviestin.

Valtaosa datasta, jota siirrettävillä liittymillä liikkui, oli eri muodossa olevia tilauksia ja laskuja. Tilaukset olivat pääosin Asiakkaan ERP-järjestelmän tuottamia positiosidonnaisia sanomia, jotka iCarella muunnettiin EDIFACT-muotoon tai laskujen tapauksessa toisin päin. Muunnoksien teossa on käytetty hyväksi EdigateManager-nimistä muunnosohjelmistoa, jonka avulla iSuiten muuntimet on luotu. Näistä ERP-järjestelmien tuottamista sanomista käytetään nimitystä inhouse, INH. Aluksi INH-sanomalle tehdään ulkoisella Perl-skriptillä esikäsittely, jonka jälkeen iSuiten muunnin muuntaa sen EDIFACT-muotoon. Laskuja EDIFACT-muodosta INH-muotoon muuttaessa on useampi askel, ja sanoma muutetaan XML-muodon kautta INH-muotoon sekä iSuiten omia, että ulkopuolisia XSLT muuntimia käyttäen. Monet muuntimista on nimetty tavalla, jossa lähtödatan ja loppudatan väliin tulee numero 2, eli esimerkiksi CSV2XML.

Työ suunniteltiin tehtäväksi liittymä kerrallaan aluksi konfiguraatioita vertaamalla ja sen jälkeen testaamalla liittymä testipuolella. Kun liittymä olisi testattu ja toimivaksi todettu, voitaisiin AMQ-liittymä sulkea muuttamalla entrypoint inaktiiviseksi ja aktivoimalla samalla WMQ:n entrypoint, jolloin sanomaliikenne siirtyisi saman tien sinne. Valmiiksi siirron pystyi toteamaan vasta kun tuotantoympäristössä oli mennyt sanomia onnistuneesti. Tuotannon WMQ:lla oli olemassa lisätestausta varten test -entrypoint, jota ei kuitenkaan tarvinnut käyttää.

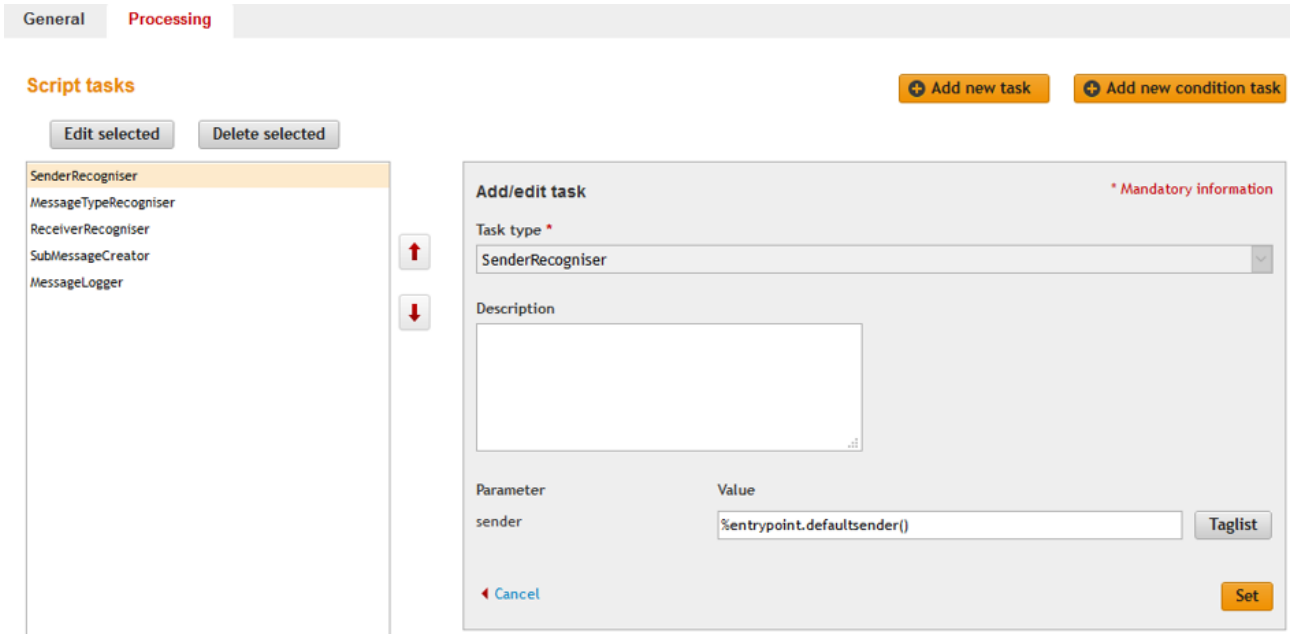
6.2 Liittymien kartoitus

Aluksi AMQ:lta kartoitettiin kaikki Asiakkaan jäljellä olevat liittymät. Liittymiä tutkittiin ICC:n kautta niiden komponenttien konfiguraatioista sekä tuotannon sanomaliikennettä tutkimalla ja kirjattiin Excel-taulukkoon liittymien tiedot. Taulukkoon kirjattiin liittymän osapuolet eli nodet. Asiakkaan liittymissä kaikkien osapuolten nodet oli nimetty EDI-ID:n mukaan, ja Asiakas tai Asiakas suorat laskut-nodet olivat osapuolena jokaisessa liittymässä. Lisäksi kirjattiin ylös nodeilla mahdollisesti tehtävät messagesriptit.

Taulukkoon kirjattiin myös liittymän entrypointin nimi sekä sen käyttämä entrypointscript tai vaihtoehtoinen messagescript. Lisäksi kansio-entrypointien tapauksessa kirjattiin ylös tiedostojen noutokansio, sftp-entrypointeilla lisäksi palvelimen osoite. Myös entrypointilla liittymän tiedoston nimeen liittyvä maski, jonka perusteella oikeanniminen tiedosto osataan noutaa, tai tiedoston nimen alku, kirjattiin ylös. Entrypointille voi iSuitella asettaa noutomaskin ja lisäksi exclude maskin, jonka avulla tietyn nimisiä tiedostoja voi jättää huomiotta. Siirrettävissä liittymissä entrypointit tekivät noutoyrityksen määritettyyn kansioon yleensä ajastetusti yhden minuutin välein.

Liittymän yhteyteen kirjattiin ylös myös sillä liikkuvien sanomien tyyppi eli messagetype sekä eri liittymillä sanomille tehtävät transformaatiot ja niiden sisältämät eri muunnosaskelmat. Lisäksi vastaanottajan nodelta kirjattiin ylös liittymään kuuluva noden endpoint ja sen tyyppi sekä kohdekansio ja palvelin. Myös liittymien aktiivisuus huomioitiin, koska joillain liittymillä ei ollut kulkenut sanomia yli kolmeen kuukauteen, mikä tulisi tekemään niiden liittymien testaamisesta hankalaa. ICaren sanomien säilytysajaksi on määritetty 90 vuorokautta.

Erityisesti oli huomioitava transformaatioiden käyttämien tiedostojen sijainnit ja tarkastuksen yhteydessä tarvittaessa päivittää tiedostopolut. Lisäksi lähes jokaisella entrypointilla ja vastaanottavan noden endpointeilla oli muutamia konfiguraatioita, jotka oli päivitettävä, koska ne tiedettiin jo etukäteen toimimattomiksi. Etukäteen oli tiedossa myös, että osa messgascripeistä ei tulisi toimiaan sellaisenaan, vaan päivityksiä vaadittaisiin. Tämä johtui pääosin siitä, että entrypointeille oli siirrettävissä liittymissä määritetty oletuslähettäjänode (default sender) sekä oletussanomatyyppi (default messagetype). Näiden määrityksien tapa on muuttunut hieman iSuite 5:n ja 6:n välillä, joten niiden päivittäminen oli huomoitava konfiguraatioita tarkastaessa ja tarkistettava että mahdollinen entrypointin messagescript hakee näitä tietoja oikealla tavalla. Kuviossa 24 näkyy messagescript, joka tunnistaa lähettäjän entrypointin defaultsender-osiosta. Jotkut kopioinnin mukana tulleet node endpointien tyypit eivät toimineet uudemmalla iSuitella, vaan ne oli muutettava. Tämä kävi melko helposti vaihtamalla kopioidun noden endpointin tyyppi noden konfiguraatioista.



Kuvio 24. Message Script iSuite 6:ssa (Hirvonen 2021).

6.3 Liittymien siirto ja toiminnallinen testaus

Siirrettävästä liittymästä kopioitiin ensin tuotannon sanomaliikenteestä yksi sanoma testidataksi. Näin saatiin mahdollisuus verrata testatun uuden liittymän mahdollisen transformaation toimivuutta testaamalla sitä tuotannon läpi menneeseen sanomaan. Tämän jälkeen tarkastettiin testipuolen liittymän konfiguraatiot ja tehtiin tarvittavat muunnokset, jonka jälkeen lähetettiin testisanoma testikansioon. Jos sanoma meni virheeseen, tutkittiin iSuiten antama virheilmoitus ja tarvittaessa iSuiten lokitietoja. Näiden tietojen avulla tehtiin korjaustoimenpiteet ja lähetettiin sanoma uudelleen. Onnistuneen lähetyksen jälkeen verrattiin vielä dataa tuotannon läpi menneeseen sanomaan. Kaiken ollessa kunnossa tehtiin vastaavat konfiguraatiot tuotannon puolelle ja siirrettiin sanomaliikenne WMQ:lle. Kaikki testaus oli manuaalista, eikä automaattisia testejä edes harkittu siirrettävien liittymien vähäisen määrän vuoksi.

Ensimmäisen liittymän siirto sujui hyvin. Entrypointin konfiguraatioita jouduttiin muuttamaan hieman, koska käytetty messagescript ei enää toiminut. Ongelma johtui siitä, että iSuite 5:llä default receiver ja default messagetype oli määritelty custom attribuutteina, ja iSuite 6:lla nuo määritykset tehtiin general parameters-määrityksissä, joten nämä määritykset tuli päivittää ajan tasalle sekä entrypointeille että messagescriptiin. Tämä aiheutti odottamattoman ongelman, koska eräs vanhoista jo aiemmin siirretyistä liittymistä käytti samaa messagescriptiä, ja muutoksen jälkeen

tuon liittymän sanomat menivät virheeseen. Ongelma saatiin korjattua muuttamalla myös tämän aiemmin siirretyn liittymän konfiguraatioita. Näin molemmat liittymät saatiin konfiguroitua niin kuin niiden kuuluisikin olla. Tilanne kuvaa hyvin integraatioiden testaamisen hankaluutta, kun korjaamalla jotain voi rikkoa jotain toisaalla. Seuraavien muutosten yhteydessä vastaavilta virheiltä vältyttiin kantakyselyn avulla, jolla selvitettiin millä kaikilla komponenteilla kyseinen messagescript on käytössä. Kysely SELECT * FROM entrypoints WHERE messagescipt= 'messagescriptin_nimi' AND active='0' listasi aktiiviset entrypointit, jotka käyttivät kyseistä messagescriptiä. Kyselyn sai tehtyä iSuiten ICC:ltä.

Seuraavaksi siirretyt liittymät olivat kaikki liikenteeltään melko hiljaisia, joten kaikkia ei saatu testidatan puutteen vuoksi testattua. Siirrot kuitenkin tehtiin konfiguraatioiden tarkastuksen jälkeen sillä ajatuksella, että iSuite ilmoittaa mahdollisesti virheeseen menneestä sanomasta, jonka jälkeen saadaan virheestä tarkempaa tietoa korjaustoimenpiteitä varten. Virheeseen mennyt sanoma on myös helppo uusia iSuitella korjausten jälkeen. Uusimisia jouduttiin tekemään muutamia. Eräs XSLT muunnos jäi päivittämättä tuotannon puolella, jonka vuoksi asiakkaalle meni virheellisen transformaation läpi mennyt sanoma. Asiakkaalle ilmoitettiin tapahtuneesta ja sanoma uusittiin.

Testi-iSuitella merkittävin eroava seikka tuotantoon verrattuna oli se, että testissä sanomat piti ohjata palvelimen sisäiseen kansioon dir -tyyppistä node endpointia käyttäen tuotannon mahdollisen sftp tai ftp-endpointin sijaan. Testatessa oli oltava tarkkana, että oikea node endpoint oli aktiivisena, jotta sanoma meni minne pitääkin. Lisäksi oli huomioitava, että palvelimella oli kaksi testi-iSuitea, AMQ ja WMQ, joten oli varmistettava, ettei väärä iSuite vahingossa poiminut testidataa noutokansiosta. Tämä varmistettiin inaktivoimalla testi-AMQ:n entrypointeja työn edistyessä. Testi-AMQ:ta ei voitu sulkea, koska sen toiminnan loppuminen olisi aiheuttanut tarpeettoman hälytyksen, sillä kaikkien neljän iSuiten toimintaa valvotaan Zabbix-ohjelmiston avulla.

Käytännössä testaus eteni niin, että tuotantosanoman alkuperäisdatasta ladattiin kopio, siirrettiin se WINSCP-ohjelman avulla oikeaan noutokansioon ja verrattiin testi-iSuiten lähettämän sanoman loppudataa tuotannossa läpi menneen sanoman loppudataan NotePad++ -ohjelman Compare-työkalulla. Joidenkin liittymien kohdalla luotiin vielä tuotannon puolellekin vastaanottavalle nodelle testi-endpoint, jotta saatiin varmuus transformaation toimivuudesta tuotantoympäristössä.

Yhdellä liittymällä oli määritelty CON-kuittaussanomat tilauksille. Ne lähtivät kuitenkin eri entrypointilta kuin siirrettävä tilausliittymä, joten näiden entrypointien siirto oli tehtävä samaan aikaan. Eräällä liittymällä transformaatio meni virheeseen testipuolella XSLT-muuntimen konfiguraatiossa olevan virheen takia. Konfiguraatiovirhe oli myös muilla XSLT muuntimilla sekä testin että tuotannon puolella päivittämättä, joten testissä virheeseen menneen sanoman avulla saatiin yksi yleinen konfiguraatio-ongelma korjattua. Yksi iSuite 5:n huonoista puolista huomattiin, kun testi-AMQ:lle tehtiin jotain kokeellisia konfiguraatiomuutoksia, ja se piti aina käynnistää uudelleen muutoksien saamiseksi voimaan. Tuotantoympäristössä tämä aiheuttaisi paljon harmia.

Osalla liittymiä ei tapahtunut minkäänlaista datan transformaatioita, joten ne olivat melko helpot siirtää. Riitti, että varmisti konfiguraatioiden olevan kunnossa. Transformaatioissa sen sijaan riitti testaamista. Monivaiheisia transformaatioita piti virheen sattuessa ajaa läpi askel kerrallaan, jotta selvisi missä muunnoksen vaiheessa virhe tapahtui. Yksittäisiä muunnoksia pystyi tarkemmin tutkimaan XMLSpy-nimisellä ohjelmistolla. Esimerkiksi erään ascii2xml- muunnoksen virheen syynä ilmeisesti olivat yhteensopivuusongelmat iSuite 5:n ja iSuite 6:n tietokantaversioissa. Yhden liittymän virheen aiheutti kannan kopioinnin yhteydessä tuntemattomasta syystä tapahtunut node endpointin tyypin muuttuminen default.jms-tyyppiseksi, vaikka konfiguraatioissa määritys oli default.ftp. Määritys saatiin korjattua muuttamalla se ftp.default-tyyppiseksi.

Viimeisenä vaiheena oli asiakas.dir-entrypointin liittymien siirto. Liittymille oli päätetty luoda omat entrypointinsa sekä testiin että tuotantoon WMQ:lle, jotta siirto kävisi hallitusti. Entrypointien luominen kävi melko helposti, koska iSuitella voi kloonata eri komponentteja. Kloonauksen jälkeen piti vain muuttaa noutomaski oikeaksi ja liittymää siirtäessä laittaa kyseinen maski iSuite 5:n asiakas.dir entrypointin määrityksiin exclude-maskiin, jotta AMQ ei noutaisi enää näitä sanomia. Koska monilla tuotannon liittymillä sanomia kulki iltaisin tai öisin, päästiin siirrettyjen liittymien sanomaliikennettä tutkimaan monesti vasta siirtoa seuraavana aamuna. Tämän vuoksi liittymien siirtoja ei viikonloppua vasten tehty, jotta korjaukset mahdollisiin virheisiin saataisiin tehtyä mahdollisimman pian.

Joissakin tapauksissa sanoma saattoi näyttää menneen onnistuneesti läpi, mutta vasta sanoman dataa tutkimalla selvisi, että muunnos ei ole toiminut toivotulla tavalla. Viimeisenä siirretyssä liit-

tymässä oli viisivaiheinen multistep-transformaatio, joka ei toiminut oikein. Yksi atox.pos -tyyppinen muunnin oli kannan kopioinnin seurauksena viallinen, joten se piti tehdä uusiksi. Transformaatio ei siitä huolimatta toiminut oikein, ja vasta lukuisten testien ja tutkimusten jälkeen eräästä xslt-muunnintiedostosta löytyi enkoodausasetus, joka ei jostain syystä uudemmalla iSuitella enää toiminut. Vika saatiin paikallistamisen jälkeen melko pienellä vaivalla korjattua. Vian löytäminen oli melko hankalaa, koska sanoma ei mennyt virheeseen vaan ainoastaan muunnoksen jälkeinen vääränlainen data paljasti, ettei muunnos toiminut. Korjatun muunnoksen toimivuus varmistettiin vielä tuotannossa luomalla vastaanottavalle nodelle testi-endpoint.

6.4 Liittymien dokumentaatio

Liittymien aiempaan dokumentaatioon oli listattu jokainen entrypoint, liittymän kuvaus ja sen osapuolet. Teknisiä yksityiskohtia ei ollut sivuille kirjattu, ja tarkempaa tietoa liittymästä saadakseen oli kirjauduttava iSuitelle konfiguraatioita tutkimaan. Migraatiotyön ohessa oli tarkoitus dokumentoida liittymät yksityiskohtaisemmin, jolloin dokumentaatiosta näkisi myös liittymien teknisiä yksityiskohtia, kuten käytössä olevat muuntimet ja niiden käyttämät muunnintiedostot. Lisäksi Ratkaisutuen virheenkäsittelyohjeisiin oli tarkoitus linkittää eri virheisiin liittyvän liittymän kuvaus.

Liittymät listattiin taulukkoon entrypointin perusteella, ja seuraavissa sarakkeissa kuvattiin liittymän osapuolet ja mitä aineistoja liittymässä liikkui. Lisäksi taulukkoon kirjattiin jokaisen liittymän noutokansio ja noutomaski sekä mahdollinen entrypointin palvelin, jos kysessä oli ftp- tai sftp- entrypoint. Omaan sarakkeeseensa merkattiin myös entrypointin käyttämä entrypointscript, liittymän sanomatyyppi sekä mahdollinen transformaatio ja muunnoksen jälkeinen sanomatyyppi. Lisäksi taulukkoon kirjattiin vastaanottavan noden endpoint, sen tyyppi ja mahdollinen palvelin. Mahdolliset muut tiedot merkattiin vielä muuta-sarakkeeseen.

Transformaatioista tehtiin vielä oma, erillinen taulukkonsa, jonne entrypointtaulukosta transformaatiosarakkeesta tuli linkki, joka vei kyseisen muuntimen kuvaukseen. Muunnostaulukossa oli myös linkki transformaatiosta takaisin entrypointin kohdalle entrypointtaulukkoon sekä muunnoksen from-sanomatyyppi ja to-sanomatyyppi. Lisäksi monivaiheisista multistep-muuntimista, joita oli suurin osa Asiakkaan liittymien muuntimista, kirjattiin jokaisen eri askelman muuntimen nimi sekä sen mahdollisesti käyttämä tiedosto. Samalla olemassa olevasta dokumentaatiosta päivitettiin jo pois AMQ-viittauksia sen lähestyvän käytöstä poistamisen vuoksi.

6.5 Työn tulos

Liittymien siirrot saatiin tehtyä onnistuneesti. Testi-AMQ:n entrypointit saatiin inaktivoitua jo hyvissä ajoin työn edetessä. Viimeisen tuotannon liittymän siirron jälkeen kuukauden vaihtuessa vanhalta iSuitelta tuli vielä tilastoliittymän tiedot kuukauden sanomaliikenteestä. Tämän jälkeen myös kaikki tuotanto-AMQ:n entrypointit inaktivoitiin. Myöhemmin ratkaisutukeen päivittäin entrypointien aktiivisuudesta tietoja lähettävää hiljaisuudenvalvontaskriptiä vielä muokattiin ajan tasalle ja AMQ-valvonta poistettiin Zabbix-ohjelmasta ja valvontanäkymä ratkaisutuen käytössä olevasta Grafana-järjestelmästä.

Asiakkaan kannalta liittymien siirto sujui lähes huomaamatta, vaikka he projektista olivatkin tietoisia. Ainoastaan yhdestä liittymien siirron vuoksi virheeseen menneestä sanomasta piti ilmoittaa asiakkaalle. Sanoma saatiin uusittua, joten mitään vahinkoa virheestä ei tapahtunut. Lisäksi yksi virhe aiheutui, koska konfiguraatioon tehdyt muutokset unohtuivat tallentaa.

Siirron vuoksi entrypointien määrä WMQ:lla kasvoi melko paljon. Jatkon kannalta on kuitenkin helpompaa, että jokaisella liittymällä on oma entrypointinsa ja asiakas.dir entrypoint saatiin pilkottua osiin. Näin mahdollisia virheiden etsintöjä tai muita ylläpidollisia toimia on helpompi tehdä. Lisäksi myös testiympäristö saatiin paremmin ajan tasalle.

7 Pohdinta

Siirrettävät liittymät olivat melko yksinkertaisia. Sanomat menivät lähettäjältä vastaanottajalle mahdollisen transformaation kautta, eikä muita osapuolia tai monimutkaisempia toimenpiteitä ollut, mikä teki siirrosta suhteellisen helppoa. Silti liittymien huolellinen testaaminen ja konfiguraatioiden tarkastaminen erittäin huolellisesti oli tärkeää, jotta siirrot eivät olisi aiheuttaneet tuotantoympäristöön häiriötä kuten ensimmäisessä kehityshaasteessa ja toiminnallisessa vaatimuksessa seitsemän todettiin. Käytetty menettelytapa oli hyvä, ja luottamus liittymien onnistuneeseen siirtoon toimimalla kyseisellä tavalla kasvoi liittymä liittymältä. Testiympäristössä tarkastettujen transformaatioiden ja sanomien reitityksen onnistumisen ja huolellisen konfiguraatioiden tarkastamisen sekä tarkan dokumentoinnin avulla saatiin minimoitua riski häiriöistä tuotantoympäristöön. Vastaava työskentelytapa liittymien siirtoon on hyvä malli jatkossakin, mikäli vastaavia migraatiotöitä tulee tehtäväksi.

Neljäs kehityshaaste ja toiminnallinen vaatimus viisi oli suuriliikenteisen entrypointin siirto. Asiakas.dir -entrypointin siirto pilkkomalla se osiin oli oikea ratkaisu, koska siirto saatiin tehtyä sujuvasti, ja jatkoa ajatellen liittymiä on helpompi ylläpitää. Entrypointin siirto sellaisenaan olisi todennäköisesti aiheuttanut huomattavasti enemmän ongelmia, koska kaikkien virheellisten konfiguraatioiden ja transformaatioiden huomaaminen etukäteen olisi ollut erittäin vaikeaa. Näin ollen työ olisi todennäköisesti häirinnyt Asiakkaan sanomaliikennettä ja luultavasti myös havaittujen virheiden korjaaminen olisi ollut hankalampaa.

Toisessa kehityshaasteessa ja toiminnallisessa vaatimuksessa kuusi esiin nostettu dokumentaatio tapahtui selainpohjaisesti sekä Excel-taulukkolaskentaohjelman ohella. Exceliin kirjattiin käytännössä kaikki liittymien tärkeimmät konfiguraatiot, joista dokumentaatioon tehtiin taulukko, johon kirjattiin liittymien yleisimmin tarvittavat tiedot. Liittymien dokumentaatiossa on hyvä malli myös muiden asiakkaiden liittymien dokumentointiin, ja sitä on helppo tarvittaessa muokata, mikäli jotain liittymäkohtaista lisätietoa olisi syytä saada helposti saataville.

Kolmannen haasteen ja toiminnallisten vaatimusten 1–4 ja 7 sekä 9 toteaminen hyväksytysti suoritetuksi on vaikeampaa, koska muutamalla liittymällä ei ole vielä mennyt sanomia, joten niiden osalta ei voida vielä aukottomasti todeta siirron onnistuneen. Pääosa liittymistä kuitenkin toimii kuten pitää, ja kaikki, joista oli testimateriaalia tarjolla, on myös testattu ja transformaatiot toimivat oikein. Noutokansioihin ei ole kertynyt tiedostoja, mikä tarkoittaa, että sanomien nouto onnistuu, ja messagescriptit toimivat kuten pitää. Sanomat menevät oikeiden node endpointien kautta eteenpäin kumppaneille ja kaikki AMQ:n liittymät saatiin suljettua.

Muutamien, lähes kaikissa siirrettävissä liittymissä olleiden konfiguraatio-ongelmien suhteen olisi hyvä miettiä voiko kyseisiä konfiguraatioita kenties muuttaa jo kannan kopioinnin vaiheessa, ainakin mikäli joskus tarvitsee siirtää vielä isompi massa liittymiä. Siirretyissä liittymissä oli myös muutamia eri lailla konfiguroituja ominaisuuksia, vaikka liittymät olivat lähes identtisiä. Tämän pohjalta voi miettiä, olisiko kehittäjille tarjottava vielä selkeämpi ohjeistus kyseisten kaltaisten liittymien luontiin, jotta konfiguraatioista saataisiin mahdollisimman yhteneviä. Tämä varmasti helpottaisi liittymien ylläpitoa ja tarvittaessa muokkausta tai siirtoa. Seuraavan iSuiten version kehittämisessä

olisi ehkä hyvä huomioida vastaavanlaiset liittymien siirrot versiolta toiselle ja miettiä kuinka kannan kopiointi kävisi niin, että liittymät voisivat olla mahdollisimman toimintakuntoisia suoraan siirron jälkeen.

Liittymien yhteydessä siirrettiin lukuisia eri transformaatioita. Monet näistä käyttivät samoja transformaatiotiedostoja. Jatkoa ajatellen voisi olla hyvä, mikäli jokaisella transformaatiolla olisi omat tiedostonsa, jolloin niihin olisi myös helpompi tehdä tarvittaessa muutoksia ilman, että muut transformaatiot muunnoksesta kärsisivät. Toki muunnosta tarvitsevalle liittymälle voi vasta muutoksen tullessa ajankohtaiseksi luoda oman tiedoston.

Liittymien testaaminen iSuitella on melko aikaa vievää lähinnä lukuisien eri konfiguraatioiden tarkastamisen vuoksi. Mikäli sanoma meni virheeseen, oli yleensä useampikin mahdollinen konfiguraatio, joita muuttamalla vika saattaisi korjaantua. Näin ollen sanomia piti monesti lähetellä lukuisia kertoja ennen kuin oikea konfiguraatio löytyi, ja tähän kului aikaa melko paljon. Testaamista vaikeutti myös se, että testi-ja tuotantoympäristö eivät olleet täysi identtisiä. Virheiden korjaamista saattaisi helpottaa, mikäli iSuiten tiedot virheen syystä olisivat selkeämpiä. Myös transformaatioita testatessa sanomia joutui lähettämään ja muuntuneita sanomia vertailemaan joskus jopa kymmeniä kertoja ennen kuin vian syy selvisi. Jonkinlainen sanoman kulun debuggausmahdollisuus voisi helpottaa testisanomien virheiden syiden löytymistä huomattavasti. Testiympäristön olisi voinut saada lähes identtiseksi tuotannon kanssa luomalla kumppanien palvelimille testikansiot, mutta sitä ei koettu tarpeelliseksi.

Liittymien siirron käytännön toteutus oli hyvin käyttäjäystävällistä, koska käytännössä koko työ tehtiin verkkoselaimen käyttöliittymän kautta. Siirrettävien liittymien kaltaisille melko vähäliikenteisille integraatioliittymille iSuite vaikuttaa erittäin hyvältä, helppokäyttöiseltä ja toimintavarmalta integraatioalustalta. Hub-and-spoke-arkkitehtuuri sopii iCaren sanomaliikenteelle hyvin, koska liittymien ja sanomien määrä on hyvin hallittavissa, ja iSuitelta on helppo tutkia ja kontrolloida sanomaliikennettä.

iSuitea on melko helppo oppia käyttämään, ja kun sen komponenttien roolit ymmärtää, konfiguroiminen on melko helppoa, eikä koodaustaitoja erityisemmin tarvita. Virheilmoituksia tutkiessa

koodin ymmärtämisestä on kuitenkin hyötyä, jotta iSuiten lähettämät Javan virheilmoitukset auttavat vikojen löytämisessä. iSuiten lokitiedostoja oli välillä hankala tutkia lokitettavan tiedon suuren määrän ja lokien lyhyehkön säilytysajan vuoksi. Tähän voi toki vaikuttaa lokituksen asetuksien avulla.

Liittymien siirto oli erittäin hyvä tapa perehtyä tarkemmin iSuiten toimintaan ja integraatioihin ylipäätään. Vaikka liittymät olivatkin melko paljon toistensa kaltaisia, oli niissä kuitenkin monenlaisia eri komponentteja ja konfiguraatioita, joihin piti perehtyä onnistuneen lopputuloksen aikaan saamiseksi. Samankaltaisia eri asiakkaiden kumppaniliittymiä on iCarella paljon, joten jatkossa virheiden ja konfiguraatio-ongelmien parissa työskentely helpottuu huomattavasti. Myös erilaisten pienkehitystöiden toteuttaminen on jatkossa huomattavasti helpompaa. Työn seurauksena ratkaisutuki voi ajaa AMQ:n alas, mikä helpottaa integraatioiden valvontaa, koska kaikki iCaren liittymät ovat yhdellä iSuitella. iSuite 6:n toimintavarmuus ja parempi tietoturva tuovat myös Asiakkaalle hyötyä siirrosta.

Lähteet

50 % digitalisaatiosta hoidetaan yllättävällä tavalla, integraatioilla. 2021. SSAB: uudenlainen näkökulma on suorastaan elinehto yrityksille. Blogiteksti digia.com-verkkosivulla. Viitattu 12.2.2023. blog.digia.com/integraatiot-ovat-elinehto-yrityksille.

Annenko, O. 2022. 12 New Application Integration Statistics and Trends for 2022. Artikkeli elastic.io sivustolla. Viitattu 28.2.2023. 12 New Application Integration Statistics and Trends for 2022 (elastic.io).

Apache Camel. N.d. Esittely camel.apache.org-verkkosivulla. Viitattu 3.3.2023. camel.apache.org.

Barney, N., Rouse M. & Mell, E. 2022. Legacy system (legacy application). Määritelmä techtarget.com -sivustolla. Viitattu 1.4.2023. https://www.techtarget.com/searchitoperations/definition/legacy-application.

Birchall, C. 2016. Re-Engineering Legacy Software. Manning Publications. Viitattu 1.4.2023. https://janet.finna.fi. Skillsoft Books ITPro.

Biscotti, F., Menon, S., Mehta, V. & Chibber, H. 2022. Marker Share: Integration Software Technologies, worldwide, 2021. Raportti Gartner-tutkimusyhtiön verkkosivulla. Viitattu 14.2.2023. Market Share: Integration Software Technologies, Worldwide, 2021 (gartner.com).

Developing integration tests. 2023. Artikkeli ibm.com verkkosivulla. Viitattu 9.4. 2023. https://www.ibm.com/docs/en/app-connect/12.0?topic=solutions-developing-integration-tests.

Digia iSuite tuotekuvaus. N.d. iSuiten tuotekuvaus Digian sisäisessä verkossa. Viitattu 1.3.2023. wiki.digia.com.

Digia yrityksenä. N.d. Yritysesittely digia.com-verkkosivulla. Viitattu 12.2.2023. www.digia.com/yritys.

Dolby, T. 2023. App Connect Enterprise (ACE) unit and component tests. Blogikirjoitus community.ibm.com verkkosivulla. Viitattu 7.4.2023. https://community.ibm.com/community/user/integration/blogs/trevor-dolby/2023/03/20/app-connect-enterprise-ace-unit-and-component-test.

Enterprise Application Integration Platforms Reviews and Ratings. 2023. Listaus Gartner- tutkimusyhtiön verkkosivulla. Viitattu 28.2.2023. https://www.gartner.com/reviews/market/application-integration-platforms.

Gordon-Byrne, G. 2014. Buying, Supporting, Maintaining Software and Equipment: An IT Manager's Guide to Controlling the Product Lifecycle. Auerbach Publications. Viitattu 26.3.2023. https://janet.finna.fi. Skillsoft Books ITPro.

Haikala, I. & Märijärvi, J. 2004. Ohjelmistotuotanto. Helsinki:Talentum.

Hautala, T. 2018. Integraatiot murroksessa API-hallinnan aikakaudella. Blogikirjoitus www.digia.com sivustolla. Viitattu 2.3.2023. https://blog.digia.com/integraatiot-murroksessa-api-hallinnan-aikakaudella

Heritage, I. 2014. Integration Throughout and Beyond the Enterprise. IBM Redbooks. Viitattu 12.2.2023. https://janet.finna.fi. Skillsoft Books ITPro.

Hirvonen, A. & Laaksonen, H. N.d. EDI:n esittely Ratkaisutuelle. Powerpoint-esitys Digian sisäisessä verkossa. Viitattu 27.12.2022.

Hirvonen, A. 2021. iSuite 6 yleisesittely. Powerpoint-esitys Digian sisäisessä verkossa. Viitattu 27.12.2022.

Hohpe, G. & Istvanick, W. 2002. Test-driven Development in Enterprise Integration Projects. Artikkeli enterpriseintegrationpatterns.com-verkkosivulla. Viitattu 7.4.2023. https://www.enterpriseintegrationpatterns.com/docs/TestDrivenEAI.pdf.

Hohpe, G. & Woolf, B. 2003a. Introduction. Artikkeli enterpriseintegrationpatterns.com sivulla. Päivitetty 2023. Viitattu 27.2.2023. https://www.enterpriseintegrationpatterns.com/patterns/messaging/Introduction.html.

Hohpe, G. & Woolf, B. 2003b. Solving Integration Problems Using Patterns. Artikkeli enterpriseintegrationpatterns.com sivulla. Päivitetty 2023. Viitattu 27.2.2023. https://www.enterpriseintegrationpatterns.com/patterns/messaging/Chapter1.html.

Hogg, R. 2008. SOA and ESB Architecture with BizTalk. Indianapolis: Wiley publishing. Viitattu 29.1.2023. https://janet.finna.fi. Skillsoft Books ITPro.

Integration technology categorization. N.d. Verkkomateriaali Digian sisäisessä verkossa. Viitattu 3.3.2023. wiki.digia.com

JSON. N.d. Esittely wikipedia.org verkkosivulla. Viitattu 2.3. fi.wikipedia.org/wiki/JSON.

Kivisaari, T. 2016. API:t ovat modernin integraatiostrategian ydin. Blogikirjoitus Digian verkkosivulla. Viitattu 2.3.2023. https://blog.digia.com/rest-api.

Källström J. N.d. Integration Patterns, verkkomateriaali Digian sisäisessä verkossa. Viitattu 7.4.2023. wiki.digia.com.

Källström, J. 2022. Viisi integraatiotrendiä vuodelle 2023. Blogikirjoitus Digian verkkosivuilla. Viitattu 28.12.2022. blog.digia.com/viisi-integraatiotrendia-vuodelle-2023.

Laaksonen, H. 2022. Integration specialist. Digia Oyj. Teams-keskustelu. 27.12.2022.

Manouvrier, B. & Menard, L. 2008. Application integration: EAI B2B BPM and SOA. Hoboken, NJ: John Wiley & Sons. Viitattu 17.1.2023. https://janet.finna.fi. ProQuest Ebook Central.

Nieminen, J. 2020. Event-pohjaiset integraatiot – muutakin kuin hypeä? Blogikirjoitus Digian verkkosivuilla. Viitattu 10.4.2023. https://blog.digia.com/event-pohjaiset-integraatiot.

Nieminen, J. & Rannikko, K. 2020. Tekoälyn käyttö yleistyy integraatioissa – esimerkkinä Dell Boomi. Blogikirjoitus Digian verkkosivuilla. Viitattu 10.4.2023. https://blog.digia.com/tekoalyn-kaytto-yleistyy-integraatioissa-esimerkkina-dell-boomi.

Open Source Dependency Management: Trends and Recommendations. N.d. Artikkeli sonatype.com verkkosivulla. Viitattu 2.4.2023. https://www.sonatype.com/state-of-the-software-supply-chain/open-source-dependency-management-trends-and-recommendations.

Open Source Supply, Demand and Strategy. N.d. Artikkeli sonatype.com verkkosivulla. Viitattu 2.4.2023. https://www.sonatype.com/state-of-the-software-supply-chain/open-source-supply-demand-security.

Project Quality Metrics. N.d. Artikkeli sonatype.com verkkosivulla. Viitattu 2.4.2023. https://www.sonatype.com/state-of-the-software-supply-chain/project-quality-metrics.

Rantanen, T. 2023. Integration Architect. Digia Oyj. Teams-keskustelu. 4.4.2023.

Samara, T. 2015. Erp and information systems : Integration or disintegration. Hoboken, NJ: John Wiley & Sons. Viitattu 17.1.2023. https://janet.finna.fi. ProQuest Ebook Central.

Sandborn, P. 2007. Software Obsolescence – Complicating the Part and Technology Obsolescence Management Problem. Viitattu 1.4.2023. http://escml.umd.edu/Papers/IEEE_SoftwareObs.pdf.

Schmidt, R. & Schmidt, R.F. 2013. Software Engineering: Architecture-Driven Software Development. Saint Louis: Elsevier Science & Technology. Viitattu 17.2.2023. https://janet.finna.fi. ProQuest Ebook Central.

Seth, A. & Seth, K. 2020. Understanding Service Oriented Architecture (SOA): Designing Adaptive Business Model for SMEs. BpPB Publications. Viitattu 29.1.2023. https://janet.finna.fi. Skillsoft Books ITPRO.

Sherif, M. 2010. Handbook of enterprise integration. Boca Raton, FL: Auerbach Publications. Viitattu 17.1.2023. https://janet.finna.fi. Skillsoft Books ITPRO.

System Overview. 2023. iSuiten yleisesittely Digian sisäisessä verkossa. Viitattu 1.3.2023. wiki.digia.com.

Tarkoma, S. 2012. Publish / subscribe systems: Design and principles. Hoboken, NJ: John Wiley & Sons. Viitattu 29.1.2023. https://janet.finna.fi. ProQuest Ebook Central.

TechIntro – Digia iSuite. 2023. iSuiten esittely Digian sisäisessä verkossa. Viitattu 1.3.2023. wiki.digia.com.

Toivanen, A. 2022. Integraatiot ja integraatioalustat -lyhyt oppimäärä. Artikkeli hiq.fi verkkosivulla. Viitattu 12.2.2023. HiQ Finland - Integraatiot ja integraatioalustat - lyhyt oppimäärä.

Tähtinen, S. 2005. Järjestelmäintegraatio: tarve, vaihtoehdot, toteutus. Jyväskylä: Talentum.

What is an iPaas. N.d. Esittely frends.com sivustolla. Viitattu 9.4.2033. https://frends.com/platform/what-is-an-ipaas.

What is service oriented architecture (SOA). 2020. Artikkeli redhat.com-sivustolla. Viitattu 29.1.2023. www.redhat.com/en/topics/cloud-native-apps/what-is-service-oriented-architecture.