Liiketoimintaprosessien kokonaisnäkyvyyden parantaminen järjestelmäintegraatioiden monitorointiratkaisulla

Atte Karhunen

Tampereen yliopisto
Luonnontieteiden tiedekunta
Tietojenkäsittelytieteiden tutkinto-ohjelma
Pro gradu -tutkielma
Ohjaaja: Marko Junkkari
Kesäkuu 2018

Tampereen yliopisto
Luonnontieteiden tiedekunta
Tietojenkäsittelytieteiden tutkinto-ohjelma
Atte Karhunen: Liiketoimintaprosessien kokonaisnäkyvyyden parantaminen järjestelmäintegraatioiden monitorointiratkaisulla
Pro gradu -tutkielma, 45 sivua, 1 liitesivu
Kesäkuu 2018

Tiivistelmä

Järjestelmäintegraatiot ovat tavallinen osa nykyaikaisen organisaation tietojärjestelmästrategiaa. Integraatioiden automatisoidessa tiedonsiirtoa järjestelmien välillä on näkyvyys integraatioiden toimintaan tärkeää. Tässä työssä esitellään järjestelmäintegraatioita ja niiden toteutukseen liittyviä tyypillisiä haasteita. Lisäksi esitellään kuinka ELK (ElasticSearch, Logstash, Kibana) -stackia käyttämällä organisaatioiden on mahdollista liittää yksi tai useampia tietojärjestelmiä yhden monitorointijärjestelmän alle, sekä visualisoida järjestelmien toimintalokeja. Tällä tavoin on mahdollista saada uuden tyyppistä näkyvyyttä järjestelmien toimintaan, josta voi olla hyötyä organisaation liiketoimintaprosessien kannalta olennaisten järjestelmien toiminnan seurannassa sekä virheenselvityksessä.

Avainsanat: järjestelmäintegraatiot, integraatioväylä, monitorointi

Sisällys

1 Johdanto

Organisaation kasvun myötä myös tiedon määrä organisaatioiden sisällä kasvaa, ja tietoa tallennetaan useisiin eri tietovarastoihin erilaisia käyttötarkoituksia varten. Tiedon määrän kasvaessa kasvaa usein myös tarve siirtää tietoa eri järjestelmien välillä, joka on organisaation kasvaessa (joko orgaanisesti tai organisaatioyhdistymisten kautta) varastoitu useisiin järjestelmiin eri menetelmiä käyttäen. Tällöin organisaation sisällä usein syntyy tarpeita erilaisille järjestelmäintegraatioille. Tietoa tulee voida siirtää, kerätä ja yhdistellä eri järjestelmien välillä automaattisesti, jotta olemassa olevaa tietoa voidaan hyödyntää tehokkaasti organisaation sisällä myös toisissa järjestelmissä sekä tätä tietoa käyttävissä prosesseissa. Chappell [2004] on aikaisemmin arvioinut että tarve erilaisille integraatioratkaisuille organisaatioissa tulee kasvamaan voimakkaasti, ja näin voidaan myös todeta tapahtuneen. Järjestelmäintegraatioiden kasvava tarve on synnyttänyt markkinan erilaisille ohjelmistoratkaisuille tämän tarpeen täyttämiseen. Tämän seurauksena on arvioitu että erilaisia tuotteita tähän tarkoitukseen tullaan tarjoamaan [Schulte, 2002], ja myös näin voidaan todeta tapahtuneen. Integraatiotarpeiden täyttämiseen on nykyään tarjolla useita erilaisia integraatioväyläratkaisuita, sekä kaupallisia että avoimen lähdekoodin tuotteita.

Järjestelmäintegraatioiden toteuttamisen ja käyttöönoton jälkeen siirtyvät toteutukset elinkaaressaan ylläpitovaiheeseen. Koko elinkaaren ajan on tärkeää pystyä seuraamaan ja tallentamaan järjestelmän toimintaa, monitoroinnin ja virheenselvityksen vuoksi. Tätä tarkoitusta varten kehitetyt erilaiset integraatioväyläratkaisut (ESB, Enterprise Service Bus) [Chappell, 2000] voivat integroida organisaation toiminnan kannalta tärkeää dataa, jolloin toimintaa täytyy jatkuvasti pystyä monitoroimaan, ja virheiden selvitystyössä on oleellista pystyä myös takautuvasti palaamaan väylän toimintahistoriaan (esimerkiksi järjestelmälokien avulla) ja seuraamaan kuinka erilaiset viestit ja data ovat liikkuneet väylän kautta järjestelmästä toiseen, minkälaisille tietomuunnoksille ne on mahdollisesti altistettu ja niin edelleen. Tällaisen operatiivisen monitoroinnin lisäksi on usein oleellista seurata myös bisnestoiminnan kannalta relevantteja tapahtumia integraatioissa.

Tässä pro gradu- työssä käsitellään olemassaolevan integraatioväylän monitorointiratkaisun uudistusta, sekä monitorointiratkaisun uudistuksen myötä saavutettavia etuja organisaation toiminnassa ja virheenselvityksessä, erityisesti operatiivisen monitoroinnin rinnalle toteutettavan bisnestoiminnan monitoroinnin kautta.

Luvussa 2 kuvataan järjestelmäintegraatioiden keskeisimpiä haasteita. Näistä erikseen keskitytään datan monimuotoisuuteen, tietomuunnoksiin sekä integraatioi-

den toiminnan kannalta oleellisten taustatietojen hallintaan.
Luvussa 3 kuvataan järjestelmäintegraatioiden tekniseen toteuttamiseen liittyviä yksityiskohtia ja yleisiä ratkaisumalleja. Kuvataan erilaisia integraatiotarpeita, sekä malleja integraatioiden toteutuksen ja ylläpidettävyyden kannalta kaikista yleisimpien ongelmien ratkaisemiseksi.
Luvussa 4 kuvataan todellisen organisaation toimintaa, toimintaan liittyvää integraatioväyläratkaisua, sekä integraatioväylän monitorointiratkaisua. Lisäksi käydään läpi integraatioväylän monitoroinnin kehitystarpeita, ja ratkaisut ja toimenpiteet monitoroinnin kehittämiselle, ja uudistetun monitorointijärjestelmän käyttöönottoa.
Luvussa 5 kuvataan käyttöönoton jälkeistä käyttäjätyytyväisyyttä uudistettuun monitorointiratkaisuun, sekä menetelmiä joilla käyttäjätyytyväisyyttä on arvioitu. Luvussa 6 esitellään integraatioväylän monitorointijärjestelmät ennen ja jälkeen uudistuksen, sekä vertaillaan niitä keskenään. Käydään läpi myös havaittuja seikkoja uudistuneen monitorointijärjestelmän puolesta ja vastaan.

2 Datan heterogeenisyys

Tietojärjestelmien integraatioihin liittyy useita haasteita [Halevy, 2005], yksi keskeisimpiä haasteita on datan heterogeenisyys. Organisaatioissa voi olla käytössä niiden toiminnan kannalta useita tärkeitä tietojärjestelmiä, joista jokainen käsittelee ja tallentaa dataa yksilöllisellä tavallaan omiin tietovarastoihinsa. Järjestelmät tallentavat käsittelemäänsä dataa eri tavoilla, tavallisesti joko tiedostopohjaisesti tai erilaisiin tietokantoihin. Järjestelmäintegraatioiden tarkoitus on pystyä siirtämään dataa järjestelmien välillä automaattisesti. Järjestelmät jotka voivat siirtää esteettömästi tietoa toistensa välillä, ovat interoperatiivisia [Wiederhold, 1999], ja onnistuneiden järjestelmäintegraatioiden seurauksena tietojärjestelmien interoperatiivisuus paranee. Integraatiotarve voi olla yhdestä yhteen tai useamman organisaation välillä, sekä yhteen tai kahteen suuntaan. Tallennustavan lisäksi olennainen seikka on tallennettavan tiedon syntaksi. Tällöin on luonnollisesti tärkeää, että tietomuunnokset eri järjestelmien käyttämien tietoformaattien välillä toimivat virheettömästi. Mikäli integraatiototeutuksen suorittama tietomuunnos tuottaa kohdejärjestelmän käsiteltäväksi dataa, joka ei vastaa täysin sen odottamaa muotoa tai syntaksia, kohdejärjestelmä ei pysty käsittelemään dataa.

Useista eri lähteistä kerättyä dataa joudutaan usein harmonisoimaan, sekä puhdistamaan. Datan harmonisoinnilla tarkoitetaan yksinkertaisimmillaan toimenpiteitä joilla datan laatua ja käyttökelpoisuutta pyritään parantamaan. Eri järjestelmistä kerääntyvä data voi olla hyvin monimuotoista, merkitykseltään samaa tietoa voidaan jäsentää hyvin vaihtelevin kuvauksin, jolloin dataa joudutaan muuntamaan tietorakenteista toisiin. Eri tietolähteistä kerättävä tieto voi myöskin olla "likaista"[Bernstein, 2008], jolloin tietoa voidaan joutua puhdistamaan esimerkiksi ylimääräisistä merkeistä tai kirjoitusvirheistä ennen kuin tiedon harmonisointi on mahdollista.

2.1 Tiedon jäsentäminen

Järjestelmäintegraatioiden mahdollistamiseksi eri järjestelmien on pystyttävä ymmärtämään ja käsittelemään automaattisesti muista järjestelmistä saatua tietoa. Järjestelmien välillä pitää pystyä käsittelemään erilaista tietoa, sekä järjestelmillä tulee olla jollain tavalla yhteneväinen määritys erilaiselle datalle, kuten esimerkiksi pankkitilin tai laskun tiedoille. Järjestelmien välillä kulkevien viestien tulee noudattaa jonkinlaista yhteneväistä muotoa (standardia), jotta tietoa voidaan

välittää eri järjestelmien välillä. Hasselbring [2000] on jo aiemmin todennut, että XML [W3Schools, 2018] on järjestelmäintegraatioiden yhteydessä vakiintunut tapa tiedon jäsentämiseksi. XML on kieli, jonka avulla voidaan kuvata erilaisia rakenteisia dokumentteja.
Esimerkki 1: Työntekijän tietojen kuvaus XML-formaatissa:

```
<Employee>
  <Name>John J. Doe</Name>
  <Position>Integration specialist</Position>
  <Salary>3500.00</Salary>
  <Team>Integration</Team>
  <EmploymentStarted>2017-01-10</EmploymentStarted>
</Employee>
```

Esimerkki 2: Työntekijän tietojen kuvaus XML-formaatissa

```
<Person>
  <Name>
    <First>John</First>
    <Middle>James</Middle>
    <Last>Doe</Last>
  </Name>
  <Salary>
    <NetPay>3500.00</NetPay>
    <IncomeTaxPct>19.3</IncomeTaxPct>
  </Salary>
  <Organization>
    <Title>Integration specialist</Title>
    <Team>Integration</Team>
  </Organization>
</Person>
```

Esimerkki 3: Työntekijän tietojen kuvaus XML-formaatissa

```
<EmployeeData>
  <IntegrationSpecialist Team="Integration" Salary="3500">
    <Name>John J. Doe</Name>
  </IntegrationSpecialist>
</EmployeeData>
```

Esimerkeillä 1, 2 ja 3 on pyritty havainnollistamaan, kuinka merkityksellisesti identtinen tieto voidaan XML-formaatissa kuvata usein eri tavoin. Vaikka kaksi järjestelmää käyttäsivät datan kuvaamiseen samaa formaattia ja syntaksia, tieto on silti usein rakenteellisesti eri tavalla kuvattua. XML-elementtien nimet voivat olla hyvin erilaisia, tietoa voidaan järjestelmästä riippuen joutua pilkkomaan tai yhdistämään eri elementeistä, tai tieto voi olla kuvattu eri tarkkuudella. Yhdessä järjestelmässä voi olla myös tallennettuna dataa, jota ei toisessa järjestelmässä tunneta laisinkaan.
Datan kuvaamiseen luodut syntaksit kuten XML ratkaisevat ainoastaan osan ongelmasta, sillä samaa standardia käyttämällä voidaan semanttisesti samankaltainen data kuitenkin kuvata usein eri tavoin. Eri tiedot voidaan järjestää hierarkkiseesti XML-dokumenteissa eri tavoin, ja samaa dataa sisältävät elementit voivat olla eri tavoin nimettyjä tai niihin voi liittyä eri tavalla muotoiltua tietoa kahden eri järjestelmän välillä. Vaikka kaksi eri järjestelmää käyttäisivätkin samaa standardia tiedon kuvaamiseen, joudutaan tietoa usein vielä muokkaamaan tai järjestelemään uudelleen tietoa järjestelmien kesken välitettäessä.
Tiedon jäsentämisen monimuotoisuuden lisäksi samaa tietoa voidaan kuvata eri tavoin, kuten esimerkiksi:

- 2008-01-12
- 12.1.2018
- Jan 12th, 2018

2.2 Datan muuntaminen

Datan ollessa tallennettuna eri järjestelmissä eri tavoin, ilmenee järjestelmäintegraatioissa oleelliseksi toimenpiteeksi erilaiset tietomuunnokset yhdestä muodosta toiseen (tai useampaan). Tietomuunnoksien tekninen toteuttaminen on mahdollista moninaisin tavoin, ja juuri tätä tehtävää varten on kehitetty esimerkiksi XSLT (Extensible Stylesheet Language Transformations), tai XQuery [Boag, 2002], joita käyttämällä on voidaan muuntaa rakenteinen XML-dokumentti toisen määrityksen (skeeman) mukaiseksi. XML:n ollessa yleinen tapa rakenteisen tiedon kuvaamiseen, on myös XSLT:n käyttö melko tavallista tietomuunnoksien yhteydessä. Erilaiset integraatioväylät tavallisesti myös tarjoavat omia työkalujaan tietomuunnoksien toteuttamisen helpottamiseksi.

Tietomuunnoksen teknisen toteuttamisen hahmottamiseksi esitellään seuraavaksi havainnollistava esimerkki tietomuunnoksesta. Esimerkki 4 sisältää päivämäärän, ja kyseinen tieto on kuvattu kahdella eri tavalla. Ensimmäinen XML-dokumentti A sisältää yhden juurielementin Epoch, ja Date-elementin, jossa on kuvattuna päivämäärä muodossa vvvv-kk-pp. Toinen dokumentti B sisältää juurielementin Date, jonka lapsielementtejä ovat Year, Month ja Day-elementit, joista jokainen sisältää nimensä mukaisesti vuoden, kuukauden ja päivän numeron.
Esimerkki 4:

```
Dokumentti A:
<Epoch>
  <Date>2018-02-06</Date>
</Epoch>

Dokumentti B:
<Date>
  <Year>2018</Year>
  <Month>02</Month>
  <Day>06</Day>
</Date>
```

Esimerkissä 5 kuvataan yksinkertainen, XML-rakenteinen muunnosdokumentti. Dokumentti sisältää kuvauksen vaadittavista tietomuunnoksista kun muunnetaan dokumentti A dokumentiksi B.
Esimerkki 5:

```
<DataMapping>
  <MapValue>
    <Source>Epoch/Date</Source>
    <Target>Date/Year</Target>
    <Transform>substring(0,4)</Transform>
  </MapValue>
  <MapValue>
    <Source>Epoch/Date</Source>
    <Target>Date/Month</Target>
    <Transform>substring(5,7)</Transform>
  </MapValue>
  <MapValue>
```

```
    <Source>Epoch/Date</Source>
    <Target>Date/Day</Target>
    <Transform>substring(8,10)</Transform>
  </MapValue>
</DataMapping>
```

Tietomuunnoskielen DataMapping-elementti sisältää MapValue-elementtejä, joissa kuvataan kuinka dokumentin A elementtien sisältämä tieto tulisi muuntaa dokumentin B elementteihin, sekä mahdolliset tietomuunnokset. Esimerkissä siis dokumentin A Date-elementin sisältö muunnetaan dokumentin B elementteihin, ja Transform-elementin sisällä on kuvattu tietomuunnos, jota datalle tehtäisiin. Tässä tapauksessa kuvitteellisella substring()-funktiokutsulla otetaan lähdekentän sisältämästä merkkijonosta haluttu kohta, ja lisätään tieto dokumentin B elementin sisältämäksi tiedoksi. Tällaista toimintaa kuvataan tavallisesti järjestelmäintegraatoissa englanninkielisellä nimellä "data mapping"[Shahbaz, 2015]. Suomenkielistä vastinetta sanalle ei ole olemassa ja järjestelmäintegraatioiden arjessa puhutaankin suomeksi tavallisesti lainasanaa käyttäen "mappauksista". Todellisuudessa käytössä olevat tietomuunnoksiin tarkoitetut ratkaisut ovat huomattavasti ilmaisuvoimaisempia kuin tässä esimerkin vuoksi kuvattu.

2.3 Perustietojen hallinta, Master Data Management (MDM)

Organisaation laajentuessa ja datan määrän kasvaessa muuttuu datan hallinta myös osaksi järjestelmäintegraatioiden haastetta. Organisaatioiden informaatiostrategiaan kuuluu usein jonkinlainen perustietojen hallintajärjestelmä, jonka avulla pyritään varmistamaan organisaation datan oikeellisuus, kattavuus, sekä hallitsemaan tätä tietoa keskitetysti yhden palvelun kautta. Tällaista tietoa kutsutaan usein perustiedoksi tai ydintiedoksi (englanniksi "master data"). Esimerkiksi organisaation työntekijöiden tilinumerot ovat palkanmaksun kannalta tärkeää perustietoa, jonka toivotaan olevan yksiselitteistä ja ajantasaista. Perustietojen oikeellisuus on tärkeää, ja perustiedot voivat olla järjestelmäintegraatioissa hyvin keskeisessä roolissa. Lisäksi perustietojen hallinta itsessään myös oleellinen osa organisaatioiden toimintaa, ja usein myös iso haaste isoille organisaatioille [Loshin, 2010].

Eri järjestelmät käyttävät usein omia tietovarastojaan käyttämänsä datan tallentamiseen, jolloin organisaation data hajautuu useisiin eri järjestelmiin. Organisaation kannalta tärkeää dataa voidaan tallentaa tai päivittää useiden eri järjestelmien kautta, jolloin jokin tieto eri järjestelmien ja perustiedon välillä voi olla

synkronoimaton. Tällöin syntyy tarve usein myös tehdä muutoksia tai päivityksiä järjestelmäintegraatioiden yhteydessä organisaation perustietoihin. On myös mahdollista, että syntyy tilanne, jossa järjestelmäintegraation avulla kuljetetaan viestejä järjestelmästä A järjestelmään B. Järjestelmä B kuitenkin vaatii/olettaa viestissä olevan jotain tietoa, jota ei ole tallennettuna järjestelmään A. Tällöin esiin voi nousta tarve rikastaa viestiä integraatioväylällä, ja rikastamisen suorittamiseksi tarvittava data hyvin usein löydetään master data-järjestelmästä.
Suurten organisaatioiden tapauksissa on siis hyvin mahdollista, ettei varsinaisia kahden järjestelmän välisiä suoria (point-to-point) integraatioita ole ollenkaan olemassa. Monimutkaisten integraatiotarpeiden seurauksena viestien kulkiessa integraatioväylän kautta kahden järjestelmän välillä, täydennetään ja rikastetaan viestiä integraatioväylällä tiedolla, jota noudetaan useammasta tietojärjestelmästä.

2.4 Integraatioväylän toiminnan monitorointi

Integraatioväylän automatisoi järjestelmien välistä datan siirtoa ja tietomuunnoksia. Toiminnan seurannan ja oikeellisuuden varmistamisen vuoksi on pystyttävä tarkkailemaan järjestelmän toimintaa. Organisaation koon, integraatioiden määrän ja järjestelmien käyttöasteiden kasvaessa myös integraatioväylän käsittelemien viestien määrä nousee. Integraatioväylä pystyy siirtämään ja käsittelemään dataa merkittävästi nopeammin kuin manuaalista tiedonsiirtoa tekevät työntekijät. Tästä seuraa että käsiteltävien viestien määrän ollessa suuri, järjestelmien toimintahäiriöit aiheuttavat hyvin nopeasti myös suuria määriä virheitä. Tällaisten virhetilanteiden korjaaminen tavallisesti vaatii suuria määriä manuaalista työtä tilanteen korjaamiseksi.
Tavallisimpia virheitä järjestelmäintegraatioissa ovat erilaiset tietoliikennehäiriöt, lähdejärjestelmiin tallennettu virheellinen tai puutteellinen data, tai automaattisien tietomuunnoksien toimintalogiikassa olevat virheet tai puutteet. Tietomuunnoksien puutteet voivat johtua puutteellisesta määrittelystä, tai muunnettavan lähtödatan poikkeavuuksista. Mikäli toiminnassa havaitaan virhe, on järjestelmien oikeellisen toiminnan kannalta kriittistä havaita mahdolliset toimintahäiriöt mahdollisimman pian, jotta toimintahäiriöstä pystytään toipumaan nopeasti sekä minimoidaan virheiden määrä.
Yksinkertaisimpia monitorointimenetelmiä ovat erilaiset hälytykset, joita integraatioväylä voi nostaa poikkeuksien tapahtuessa, kun tiedoston käsittelyssä jossain vaiheessa (vastaanotto, käsittely, lähetys) tapahtuu virheitä. Joissain tapauk-

sissa voi olla tärkeää myös tarkkailla integraation läpi kulkevien viestien määrää. Mikäli integraation läpi kulkee tapauskohtaisiin raja-arvoihin verraten liian vähän tai liian useita sanomia jollain tarkasteluvälillä, voi kyseessä olla toimintahäiriö. Integraatio saattaa toimia teknisesti ilman virheitä, jolloin sanomat kulkevat järjestelmien välillä, ja tietomuunnokset onnistuvat, mutta tietomuunnoksissa saattaa olla puutteita tai virheitä (bugeja). Tällöin tietomuunnokset voivat aiheuttaa puutteellisen sekä virheellisen datan automaattista siirtymistä järjestelmien välillä. Tämä voi aiheuttaa nopeasti ongelmia toisaalla, mikäli toiset prosessit nojaavat integraatioiden automaattisesti siirtämään dataan toiminnassaan.

3 Järjestelmäintegraatiot

Järjestelmäintegraatioilla tähdätään organisaation toiminnan tehostamiseen datan käsittelyä automatisoimalla. On tavallista, että eri järjestelmien välillä joudutaan siirtämään dataa manuaalisesti. Tämä tarkoittaa olemassa olevan tiedon kopioimista järjestelmästä toiseen, sekä useimmiten myös tietomuunnoksien tekemistä kun kopioitua dataa joudutaan muokkaamaan toisen järjestelmän ymmärtämään muotoon. Manuaalinen työ on aikaa vievä prosessi, joka lisää virheiden syntymisen mahdollisuutta. Mikäli tällaisia datan siirtoja joudutaan tekemään säännöllisesti tai useiden järjestelmien välillä, on toiminnan automatisointi usein kannattavaa.

Toistuvien prosessien automatisoinnin seurauksena organisaation työntekijät pystyvät käyttämään työaikansa tehokkaammin, sekä virheiden syntymisen mahdollisuudet pienenevät. Onnistuneiden järjestelmäintegraatioiden avulla data liikkuu eri järjestelmien välillä tehokkaammin ja virheettömämmin, jolloin pystytään tarjoamaan ajantasaisempaa sekä oikeellisempaa dataa eri tietojärjestelmille ja niiden käyttäjille.

3.1 Esimerkki järjestelmäintegraatiosta ja sen toteutuksesta

Oletetaan että organisaatiolla A on käytössään tietojärjestelmä X, johon työntekijät kirjaavat päivittäin työpäivän aikana käytetyt työtunnit palkanlaskentaa varten. Kyseessä on web-käyttöliittymä, jonka kautta käyttäjät syöttävät päivittäin henkilökohtaisilla tunnuksillaan työtunnit järjestelmään, jotka tallennetaan relaatiotietokantaan, joka toimii sekä palkanlaskun työkaluna, että organisaation raportointityökaluna. Organisaatio A yhdistyy toisen organisaation, organisaatio B kanssa, jossa työtunnit kirjataan erilaiseen järjestelmään Y. Tämä järjestelmä tallentaa tiedot XML-tiedostoihin palkanlaskentaa varten. Myös organisaation B tuntikirjaustiedot halutaan saada organisaatio A:n käyttämään palkanlaskentajärjestelmään, mutta erinäisistä syistä johtuen organisaatio B ei voi luopua nykyisen järjestelmän käytöstä. Tällaisessa tilanteessa syntyy tarve järjestelmäintegraatiolle.

Järjestelmäintegraation toteuttaminen vaatii seuraavia toimenpiteitä: Organisaation B tiedostoihin tallennettu tuntikirjausdata pitää siirtää tietojärjestelmästä toiseen. Koska tieto on lähdejärjestelmässä Y XML-tiedostoissa, ei sitä voida sellaisenaan viedä kohdejärjestelmään X, joka käyttää tietovarastonaan relaatiotietokantaa. Tarvitaan myös tietomuunnos.

Lähdejärjestelmä Y tallentaa tiedostot levylle, tuntikirjausdatan formaatti on esitetty esimerkissä 6.
Esimerkki 6:

```
<HourReport>
  <Name>John Doe</Name>
  <Date>01/11/2017</Date>
  <StartWork>7:28 AM</StartWork>
  <EndWork>3:34 PM</EndWork>
  <LunchBreakMin>36<LunchBreakMin>
</HourReport>
```

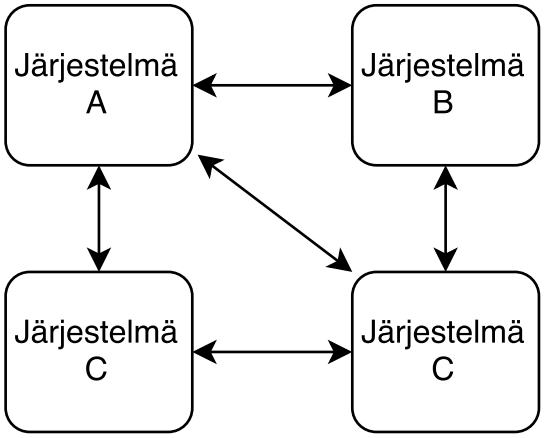
Lähdejärjestelmässä on yksi kenttä nimelle, päivämäärälle, työn alku- ja loppuajalle sekä lounastauon pituus minuutteina. Kohdejärjestelmässä X tieto on jäsennelty eri tavalla: Kirjauksessa on etu- ja sukunimi, päivämäärä (formaatti yyyy-MM-dd), ja tehdyt työtunnit kirjataan täysinä tunteina ilman lounastaukoa. Jotta järjestelmän Y tarjoamaa XML-dataa voidaan tallentaa tietokantaan, tulee sille tehdä seuraavat tietomuunnokset:

- Nimi-tieto pitää jakaa Etu- ja sukunimeksi
- Päivämäärä täytyy muuntaa 01/11/2017 -> 2017-11-01
- Työtuntien määrä pitää laskea alku- ja loppukellonajan perusteella, ja vähentää saadusta tuloksesta lounastauko, sekä pyöristää tulos lähimpään tasatuntiin

Yllä kuvatussa skenaariossa on tarve point-to-point integraatiolle, jonka toteuttaminen on mahdollista kolmessa vaiheessa:

- Tiedoston nouto lähdejärjestelmästä, ja sen sisällön lukeminen
- XML-tiedoston sisällön jäsentäminen, sekä tietomuunnokset kohdejärjestelmän vaatimaan muotoon
- Muunnetun tiedon tallentaminen kohdejärjestelmän tietokantaan

Jokaisessa vaiheessa on olemassa myös virheiden mahdollisuuksia. Tietoyhteyksissä voi olla tilapäisiä tai jatkuvia häiriöitä, lähdedatassa voi olla virheitä tai puutteita jotka aiheuttavat tietomuunnoksen epäonnistumisen. Myös muutokset tietojärjestelmien toiminnassa voivat aiheuttaa muutostarpeen myös integraation toteutukseen.



Kuva 3.1 Point-to-point integraatioita

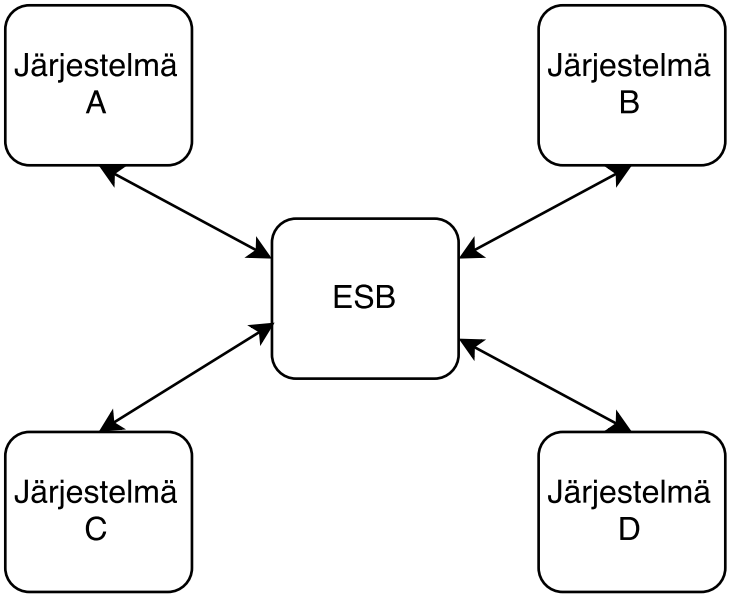
3.2 Point-to-point integraatiot

Järjestelmäintegraatioiden toteuttaminen yksinkertaisin malli on suora yhteys kahden tietojärjestelmän välillä. Tässä tapauksessa tietojärjestelmällä voidaan tarkoittaa esimerkiksi tietovarastoa tai muuta järjestelmää, joka lähettää tai vastaanottaa dataa. Tällainen suora järjestelmäintegraatio toteutuu kahdessa vaiheessa: dataa siirretään lähde- ja kohdejärjestelmän välillä, sekä datan muokkaaminen kohdejärjestelmän käyttämään formaattiin. Tällaisen mallin mukaan toteutettuja integraatioita kutsutaan point-to-point integraatioiksi.
Point-to-point integraatio voi toimia yksisuuntaisesti tai molempiin suuntiin. Yksisuuntaisessa integraatiossa data liikkuisi ainoastaan järjestelmästä A järjestelmään B, ja kaksisuuntaisessa integraatiossa data voi kulkea järjestelmien välillä molempiin suuntiin. Yksi järjestelmä voi myös lähettää tai vastaanottaa useampaa erilaista dataa, jolloin mahdollisesti joudutaan kahden järjestelmän välille rakentamaan useampia tietomuunnoksia. Käytännössä tämä tarkoittaa, että kahden järjestelmän välinen integraatio voi koostua useasta erilaisesta tietomallista, joita kahden järjestelmän välillä siirretään ja joille tehdään tietomuunnoksia.
Point-to-point integraatio toteutaan aina kahden järjestelmän välillä. Lähde- ja kohdejärjestelmät voivat olla toteutettu eri teknologioilla, ja viestinvälitys täytyy toteuttaa kyseiset teknologiat huomioiden. Integraatiototeutus on myös räätälöity juuri näiden kahden tietojärjestelmän välille, ja toteutus keskustelee käyttämällä viestiformaatteja ja syntakseja jotka ovat näiden järjestelmien määrittelemiä.
Point-to-point -suunnittelumallia käyttäessä integraatioiden määrä kasvaa eksponentionaalisesti sitä mukaa kun integroitavien järjestelmien määrä kasvaa. Tämän seurauksena mahdollinen muutos yhden järjestelmän tuottamaan tai käyttämään

dataan edellyttää myös muutoksia jokaiseen integraatiototeutukseen, jotka liittyvät muutoksien kohteena olevaan järjestelmään. Point-to-point -integraatioiden ylläpito ja hallinnoiminen muuttuu nopeasti hyvin vaikeaksi, jos integroitavien järjestelmien määrä kasvaa.

3.3 Integraatioväylä

Järjestelmäintegraatioiden toteuttaminen on mahdollista usein eri tavoin. Integraatioiden toteuttajat voivat rakentaa usealla eri tavalla teknisen kokonaisuuden, joka pystyy toteuttamaan aiemmassa esimerkissä kuvatun kaltaisen point-to-point integraation eri vaiheet (tietojen haku tai vastaanotto lähdejärjestelmästä, tietomuunnos eri järjestelmien välillä ja muunnetun tiedon vieminen kohdejärjestelmään). Tarve toteutetulle integraatiolle voi jatkua pitkään sen jälkeen, kun alkuperäisen integraation toteuttanut kehittäjä tai kehittäjät ovat poistuneet organisaatiosta. Integraatiototeutusten kattava tekninen dokumentointi on tärkeää tällaisia tilanteita varten. Integraatioiden määrän kasvaessa, hyvinkin kattavasti dokumentoidut point-to-point integraatiot muuttuvat nopesti hyvin vaikeasti hallittavaksi kokonaisuudeksi, mikäli integraatiot ovat toteutettu vaihtelevia suunnittelumalleja ja teknisiä komponentteja käyttäen.
Koska järjestelmäintegraatioiden perusluonne noudattaa usein samanlaista kaavaa (nouda tai vastaanota data järjestelmästä A, muunna se järjestelmän B hyväksymään muotoon, tallenna tai vie tieto järjestelmään B), on luonnollista, että samoja tai hyvin samankaltaisia toimivia ohjelmistokomponentteja on kannattavaa uudelleenkäyttää uusia järjestelmäintegraatioita kehittäessä. Kun esimerkiksi tiedostojen noutamiseen SFTP-palvelimelta käytetään toistuvasti samaa komponenttia, muuttuu järjestelmäintegraatioiden toiminnan kokonaisuuden hallinta helpommaksi.
Integraatioväylä eli ESB (Enterprise Service Bus) -ratkaisut helpottavat juuri tällä tavoin järjestelmäintegraatioiden toteuttamisen ja kokonaisuuden hallintaa. ESB-ratkaisut tarjoavat usein kattavan työkalupakin järjestelmäintegraatioiden toteuttamiseen vaadittavien askeleiden toteuttamiseen. Tällaisia työkaluja ovat esimerkiksi erilaiset adapterit, joiden avulla pystytään viemään tai noutamaan tiedostoja tiedostopalvelimilta, tekemään kyselyitä relaatiotietokantoihin, tai välittämään sanomia muihin liitettävien järjestelmien rajapintoihin. Integraatioväylään liitettävien järjestelmien määrän kasvaessa on kokonaisuus helpommin hallittavissa, sillä integraatiot on toteutettu integraatioväylän tarjoamia komponentteja käyttäen. Integraatioväylän ollessa keskitetty kokonaisuus, on myös in-



Kuva 3.2 Hub-and-spoke

tegraatioiden toiminnan ja virhetilanteiden seuraaminen helpompaa.

3.4 Palvelukeskeinen arkkitehtuuri

Point-to-point -integraatioiden suuren määrän ylläpitoon ja muutosten hallintaan liittyvien ongelmien ratkaisun helpottamiseksi on kehitetty palvelukeskeisen arkkitehtuurin malli (Service Oriented Architechture, SOA) [Erl, 2008] ja palveluväylätuotteet (Enterprise Service Bus, ESB) [Chappell, 2004]. Palvelukeskeisessä arkkitehtuurissa kaikki integroitavat järjestelmät liittyvät palveluväylään vain kertaalleen, ja viestinvälitys kaikkien integroitavien järjestelmien välillä kulkee aina tämän palveluväylän kautta. Tätä arkkitehtuuria voidaan kuvata myös hub-and-spoke-mallina, jota havainnollistetaan kuvassa 3.2. ESB eli Enterprise Service Bus toimii keskitettynä integraatioväylänä, johon muut integroitavat järjestelmät liittyvät sen sijaan, että järjestelmät liittyisivät suoraan toisiinsa kuten point-to-point integraatioissa. Tällöin integraatioitavien järjestelmien määrän kasvaessa, nousee toteutettavien integraatioiden määrä lineaarisesti. Tämä tekee integraatioiden ylläpidosta ja muutosten hallinnasta helpompaa. Integraatiotoimintojen keskittämisen myötä myös integraatoiden toiminnan seuranta ja virheiden hallinta on helpompaa.

Palveluväylä ei kuitenkaan itsessään ratkaise järjestelmäintegraatioihin liittyviä ongelmia ja onkin mahdollista rakentaa edelleen point-to-point integraatioita pal-

veluväylää käyttäen. Joidenkin järjestelmien integraatioväylän kautta lähettämä tai vastaanottama data voi olla käyttötarkoitukseltaan tai muodoltaan hyvin spesifiä, jolloin tiedon muuntamisella yleismuotoiseen (kanoniseen) malliin ei saavuteta lisäarvoa. Järjestelmät voivat myös itsessään tarjota työkaluja lähetettävän tai vastaanotettavan datan formaatin konfigurointiin, jolloin integraatioväylällä ei välttämättä ole edes tarvetta toteuttaa erillisiä tietomuunnoksia, vaan suora tiedonsiirto voi jo riittää järjestelmäintegraation toteuttamiseen. Kun suora tiedostonsiirtokin kulkee integraatioväylän kautta, on tällöin myöten myös keskitetysti seurattavissa ja hallittavissa.

3.5 Järjestelmäintegraation totetuksen käytäntö

Aiemmissa kohdissa kuvatuissa point-to-point (kuva 3.1) sekä hub-and-spoke (kuva 3.2) -integraatioarkkitehtuuria visualisoivat kaaviot ovat abstraktioita todellisuudessa vallitsevista tilanteista, joita tarkennetaan seuraavaksi. Kaavioissa laatikot kuvasivat järjestelmiä, ja nuolet laatikoiden välillä kuvasivat järjestelmäintegraatioita eri järjestelmien välillä. Seuraavaksi tarkennetaan, mitä tarkoitetaan "järjestelmällä"ja mitä tarkoitetaan "integraatiolla".
Järjestelmien välille piirretyt nuolet kuvaavat järjestelmäintegraatioita, eli automatisoitua datan siirtoa ja muuntamista näiden järjestelmien välillä. Integraatio sisältää siis datan siirron - tämä voi siis käytännössä tarkoittaa esimerkiksi joko tiedostojen kopiointia SFTP-palvelimelta, web service-kutsua tai tietokantakyselyä. Lisäksi yhden järjestelmän tarjoama data pitää integraatiossa muuntaa toisen järjestelmän ymmärtämään muotoon. Tämän tietomuunnoksen toteuttavaa komponenttia kutsutaan ohjelmistokehityksessä adapteriksi [Gamma, 1995]. Nämä integraatiot voivat järjestelmistä riippuen olla yksi- tai kaksisuuntaisia. Tarkoittaen sitä, että integraatiotarpeesta riippuen järjestelmästä joko luetaan, tai sinne kirjoitetaan dataa, tai molempia.
Järjestelmä-laatikot kuvaavat erilaisia tietojärjestelmiä. Kyseessä voi olla esimerkiksi levyjako, SFTP-palvelin, SQL-tietokanta, tai jokin laajempi organisaation sisäisessä käytössä oleva tietojärjestelmä tai ohjelmistokokonaisuus. Järjestelmäintegraatioiden kannalta järjestelmien oleelliset osat ovat dataformaattien lisäksi liittymät, eli rajapinnat. Kuinka järjestelmiin liitytään, tai kuinka päästään käsiksi niiden tarjoamaan dataan ja kuinka niihin saadaan talletettua dataa. Järjestelmät, jotka ovat olleet käytössä pitkän aikaa, eikä niitä ole alunperin suunniteltu järjestelmäintegraatioita tai liitettävyyttä ajatellen, saattavat tarjota esimerkiksi omassa formaatissaan tallentamiaan tiedostoja (esimerkiksi .csv-

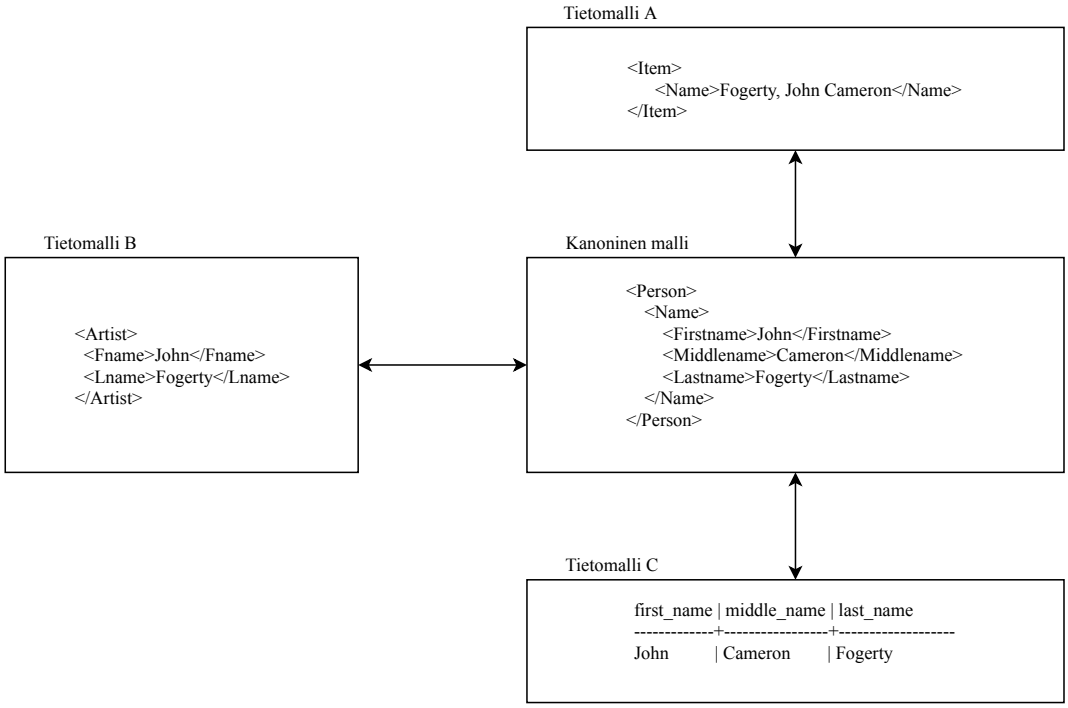
tiedosto) jotka kopioidaan levyjakoon tai SFTP-palvelimelle saataville. Nykyaikaiset järjestelmät tarjoavat usein erilaisia SOAP- tai REST-rajapintoja joita voidaan käyttää HTTP-kutsuilla internetin välityksellä, ja ne palauttavat kyselijälle vastauksena tarjoamansa datan esimerkiksi XML (Extensible Markup Language)- tai JSON (JavaScript Object Notation)-formaatissa. [Shafranovic, 2005] [Bray, 1997] [Bray, 2014] [Rodriguez, 2008] [Barrett, 2001]

3.6 Kanoninen malli

Olennainen osa palveluväylän viestinvälitystä on kanoninen tietomalli, joka on integroitavista sovelluksista ja niiden omista tietomalleista riippumaton tiedon esitysmalli. Integroitaessa järjestelmää integraatioväylään, suoritetaan tietomuunnos sovellusriippuvaisesta tiedosta kanoniseen malliin, ja toisin päin. Täten integroitava järjestelmän liittämiseksi palveluväylään riittää tietomuunnos järjestelmäspesifistä tietomallista kanoniseen malliin ja takaisin. Koska kaikki palveluväylälle lähetettävät viestit muunnetaan kanoniseen malliin, pystytään jokaisen palveluväylään liitetyn järjestelmän välillä kuljettamaan viestejä palveluväylän kautta. Palveluväylä huolehtii viestimuunnoksen lähdejärjestelmästän formaatista kanoniseen malliin, ja kanonisesta mallista kohdejärjestelmän formaattiin. Mikäli palveluväylään integroidun järjestelmän muoto tai syntaksi muuttuu, riittää ainoastaan tietomuunnoksen päivittäminen järjestelmän pitämiseksi mukana integraatioväylän toiminnassa.

Muutosten hallitseminen integroitavissa järjestelmissä helpottuu, sillä ainoa muuttuva osa on integroitavan järjestelmän ja palveluväylän välillä ja/tai muutos palveluväylän kanoniseen tietomalliin. Palvelukeskeinen arkkitehtuuri mahdollistaa myös soveltuvilta osin jo olemassa olevien palvelujen uudelleenkäyttämisen. Jotta palvelukeskeisestä arkkitehtuurista saataisiin irti suurin mahdollinen hyöty, toiminnan ohjauksen pitää olla suunnitelmallista. Palveluita ei tulisi toteuttaa vain yksittäisiä projekteja tai tietojärjestelmiä ajatellen, vaan tulee aina pyrkiä ottamaan huomioon suurempi kokonaisuus ja mahdollisesti tulevaisuudessa samoja palveluita käyttävät muut järjestelmät.

Kanonisen mallin määrittelyyn liittyy myös paljon haasteita. Kanonisen tietomallin tulee olla riittävän spesifi, jotta se palvelee tarkoitustaan ja sen avulla pystytään kuvaamaan tarvittavia tietoja. Kanonista mallia määrittäessä ei kuitenkaan hyvin usein voida tietää, minkä kaltaisten tietojärjestelmien tuottamaa tai tarvitsemaa dataa sen avulla tulisi tulevaisuudessa kuvata. Täten on mahdollista, että integraatiotarpeiden myötä myöhemmin ilmenee, että kanoniseen mal-



Kuva 3.3 Kanonisen mallin havainnollistaminen

liin vaaditaan muutoksia. Muutos kanoniseen malliin voi siis myös edellyttää, että useisiin tai kaikkiin kanonisesta mallista järjestelmäkohtaisiin tietomuunnoksiin vaaditaan myös muutoksia.

Kanonista tietomallia luodessa on myös oleellista huomioida konteksti, jossa toimitaan. Vaikka olisikin mahdollista luoda yleiskäyttöinen tietomalli, joka olisi niin monipuolinen ja kattava, että samaa tietomallia käyttäen voidaan kattavasti kuvata vaikkapa työntekijän, palkkakuitin kuin ajoneuvon huoltokirjan tiedot, tulisi tällainen tietomalli todennäköisesti olemaan tarpeettoman monimutkainen ja vaikeaselkoinen, eikä siten palvelisi tarkoitustaan.

Palveluväyläarkkitehtuuri tai kanonisen mallin käyttöönotto järjestelmäintegraatoissa ei siis itsessään vielä ratkaise useiden järjestelmien välisiin integraatioihin liittyviä haasteita. Näiden haasteiden helpottamiseksi on ensisijaisesti tärkeintä onnistua järjestelmäintegraatioiden määrittelyssä ja sitä myötä suunnittelussa.

Kuvassa 3.3 on pyritty havainnollistamaan tilannetta, jossa semanttisesti samaa tietoa on tallennettu eri järjestelmiin eri tavoin. Kuvassa keskimmäisessä laatikossa, johon kaikki muut laatikot on yhdistetty kaksisuuntaisilla nuolilla, on henkilötiedon kanoninen kuvaus. Kanoninen kuvaus sisältää kaikki tiedot, joilla henkilön nimitiedot saadaan yksiselitteisesti kuvattua. Järjestelmissä A ja B on

kuvattu samalle tiedolle erilainen XML-kuvaus. Järjestelmä C sisältää tietokantataulun, jossa henkilön nimitiedot ovat omissa sarakkeissaan. Kaksisuuntaiset nuolet kuvassa esittävät integraatioiden yhteydessä toteutettavia tietomuunnoksia, eli kuinka automaatiolla muunnetaan järjestelmän A tuottama data kanonisen muotoon, ja vastaavasti toisin päin. Oletetaan että kuvaan lisättäisi vielä uusi järjestelmä D. Tällöin määritellään ja toteutetaan tietomuunnokset järjestelmän D ja kanonisen mallin välillä. Näin toimittaessa olisi kanonista mallia käyttävän integraatioväylän kautta suoraan mahdollista siirtää dataa järjestelmän D ja kaikkien muiden integraatioväylään liitettyjen järjestelmien välillä.

4 Asiakascase

Aiakas on teollisuuden kunnossapitoon erikoistunut yritys. Integraatioratkaisuna käytössä avoimen lähdekoodin Mule ESB -integraatioväyläratkaisu [Dossot, 2004] [Rademakers, 2008]. Asiakas toimii useissa eri maissa, ja vastaa useiden eri teollisten toimintalaitosten kunnossapitojärjestelmien toiminnasta, työn seurannasta sekä raportoinnista. Asiakasorganisaation yhteistyö integraatioväyläratkaisun toimittajan kanssa on jatkunut jo useita vuosia. Nykyistä integraatiovälyän toiminnan monitorointituotetta ei enää kehitetä, joten uusia ominaisuuksia ei ole luvassa, ja integraatioden määrän ja monimutkaisuuden kasvaessa on syntynyt tarve kokonaisvaltaisemmalle monitoroinnille. Monitoroinnin osalta on syntynyt tarve prosessien kokonaisnäkyvyydelle järjestelmärajojen yli.

Järjestelmärajat ylittävällä kokonaisnäkyvyydellä tarkoitetaan sitä, että monitorointijärjestelmän avulla on mahdollista seurata integraatioväylän lisäksi myös muita integraatioväylään liitettyjä järjestelmiä. Operatiivisen toiminnan kannalta on olennaista pystyä seuraamaan sanoman kulkua lähtöjärjestelmästä integraatioväylän kautta kohdejärjestelmään. Tällaisia sanomia ovat esimerkiksi erilaiset työtilausten tietojen kirjaussanomat. Sanoman kulkiessa järjestelmien välillä pystytään yksittäiseen työtilaukseen liittyvät sanomat tunnistamaan esimerkiksi työtilaustunnisteen avulla. Vanhan monitoroinnin kautta voidaan kuitenkin löytää tietoa yksittäiseen työtilaukseen liittyen ainoastaan siltä osin kun sanoma on integraatioväylän käsiteltävänä. Käytännössä siitä lähtien kun sanoma luetaan sisään, siihen asti kunnes sanoma lähtee väylältä eteenpäin. Monitorointiratkaisun uudistuksena on myös tarjota näkyvyyttä järjestelmärajojen yli. Tämä tarkoittaa sitä, että myös muut järjestelmät voivat lähettää toimintalokeistaan sanomia uuden monitorointiratkaisun tietovarastoon. Tällöin käyttöliittymän kautta on mahdollista tutkia lokitapahtumia tiettyyn työtilaukseen liittyen jo ennen ja sen jälkeen, kun sanoma on integraatioväylän käsiteltävänä.

4.1 Nykytilanne

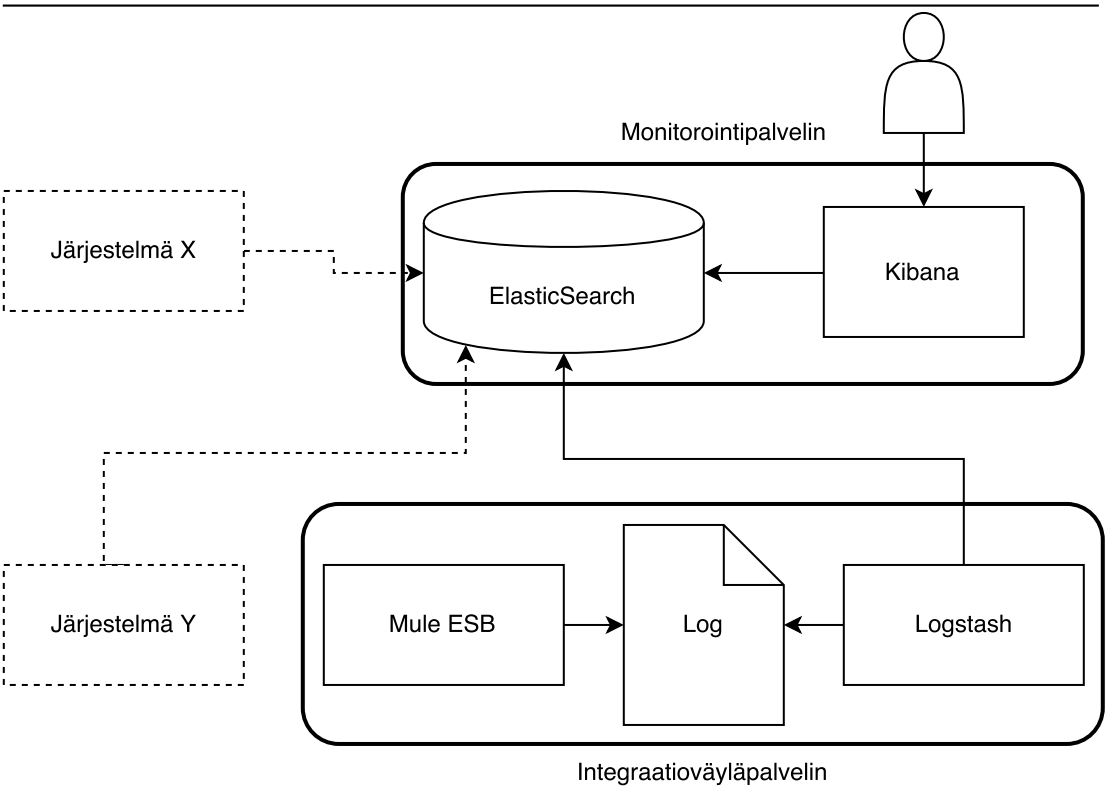
Nykyiseltään integraatioväylän logitus ja monitorointiratkaisu on Solitan toimittama Pulse-monitorointiratkaisu, joka on toiminnassa samalla palvelimella, kuin Mule ESB-integraatioväyläsovellus. Integraatioväylällä on käynnissä kymmeniä järjestelmäintegraatioita, joiden kautta välitetään useita tuhansia viestejä päivittäin eri järjestelmien välillä. Integraatioväylän kautta on yhdistetty asiakkaan toiminnan kannalta useita kriittisiä toiminnan ohjaus- ja raportointijärjestelmiä.

Nykyisen ratkaisun avulla pystytään ainoastaan monitoroimaan integraatioväylän toimintaa. Erilaisten sanomien käsittelyä ei pystytä seuraamaan monitorointijärjestelmän avulla enää sen jälkeen kun viesti on lähetetty väylältä eteenpäin. Vastaavasti ei myöskään voida seurata mitä toimenpiteitä viestille on tehty ennen sen saapumista integraatioväylän käsiteltäväksi. Pulse-monitorointiratkaisu on tekninen prosessien monitorointityökalu, joka kirjoittaa integraatioiden suoritusten aikana erilaisia selkokielisiä, integraatoiden toimintaa kuvaavia sanomia relaatiotietokantaan. Selainkäyttöliittymän avulla pystytään hakemaan ja suodattamaan tietokannasta rivejä, ja hakutuloksia tutkimalla selaamaan integraatioiden suorituksen aikana kirjoitettuja lokitussanomia. Monitorointiratkaisun avulla on mahdollista hakea sanomia halutulta aikaväliltä, halutuista integraatioista, sekä suodattaa tuloksia myös vapaamuotoisen haun muodossa. Pulse-monitorointiratkaisun hakutulokset ovat tarkasteltavissa ainoastaan rivitason tietoina, jossa monitoriin kirjoitetut sanomat ovat luettavissa, yksi rivi per kirjoitettu monitorisanoma. Taulukko sisältää tiedot integraatiosta, sanoman kirjoituksen hetkestä, sekä selkokielisen sanoman, joka integraatiototeutuksessa on määritetty. Pulse-ratkaisu ei tarjoa muunlaista kokonaisnäkymää integraatioväylän toiminnasta kuin alkunäkymän johon on koostettu onnistuneiden ja epäonnistuneiden integraatioden lukumäärät, sekä varoitukset määrätyltä ajanjaksolta. Kyseisen ajanjakson monitorointiväylän tapahtumien visualisointiin ei ole tarjolla muunlaista näkymää. Monitorointiuudistuksen tarkoituksena on tarjota asiakkaalle BPM (Business Performance Monitoring) [Frolick, 2006] mukaista näkyvyyttä integraatioväylän ja muiden järjestelmien toimintaan. Tämä mahdollistaa sen että monitorointijärjestelmän avulla saadaan näkyvyyttä pelkän prosessimonitoroinnin lisäksi myös bisnestoimintaan. Tämän näkyvyyden avulla pystytään mahdollisesti reagoimaan ongelmiin nopeammin sekä havaitsemaan erilaisia ongelmakohtia järjestelmien toiminnassa. Nykyisen monitorointiratkaisun toiminta ei myöskään tue suorituksen aikaisen metatiedon tallentamista jokaiseen lokitettuun sanomaan. Tällöin integraation päätyessä virheeseen joudutaan tarkastelemaan suorituksen aikana aiemmin kirjoitettuja viestejä, jotta voidaan saada tietoon esimerkiksi kyseisen työtilauksen tunniste, tai muuta virheen selvityksen kannalta oleellista tietoa.

4.2 Integraatioväylän monitorointiratkaisun uudistus

Integraatioiden määrän kasvaessa ja sitä myöden integroitujen järjestelmien määrän kasvaessa, on yhä suurempi tarve seurata integraatioväylän toimintaa. On-

gelmatilanteiden ratkaisun nopeuttamiseksi pitää kerätä enemmän dataa ja metadataa integraatioväylällä käsiteltävistä viesteistä.
Näihin tarpeisiin vastauksena on suunniteltu integraatioväylän monitorointiratkaisun uudistusta. Uudistuksen tarkoituksena on erottaa monitorointijärjestelmä ja integraatioväylä toisistaan eriyttämällä monitorointi integraatioväylältä eri palvelimelle omaksi erilliseksi järjestelmäkseen. Tämän saavuttamiseksi on muodostettu seuraavanlainen kokonaisratkaisu:
Integraatioväylän keräämät järjestelmälogit kerätään ja lähetetään eteenpäin erillisen keskitetyn monitorointijärjestelmän luettavaksi. Täten eriytetään integraatioväylän toiminta ja sen monitorointi toisistaan. Tähän uuteen monitorointijärjestelmään voidaan jatkossa kerätä logeja myös muista järjestelmistä. Eri järjestelmien lokeja yhdistämällä saadaan muodostettua keskitetty monitorointiratkaisu, jonka avulla voidaan monitoroita integraatioväylän kautta kulkevia prosesseja alusta loppuun asti myös järjestelmärajojen yli. Näin saadaan aikaan kokonaisvaltaisempi kuva järjestelmien toiminnasta.
Keskittämällä eri järjestelmien lokeista kerättyä dataa yhteen järjestelmään, saa järjestelmänvalvoja nopeasti kattavamman kokonaiskuvan integraatioväylän sekä integraatioväylän toisiinsa liittämien järjestelmien toiminnasta. Koska lokeista kerätty data on keskitetty yhteen järjestelmään ja yhden käyttöliittymän taakse, ei käyttäjä tarvitse erillistä pääsyä jokaisen tarkkailtavan järjestelmän lokeihin, eikä hänen tarvitse pystyä tulkitsemaan useissa eri formaateissa olevia lokeja. Teknologiaratkaisuna tässä tapauksessa käytetään niin kutsuttua ELK-stackia, joka koostuu kolmesta teknisestä ohjelmistokomponentista: ElasticSearch, Logatash, Kibana. [Kleindienst, 2016]
Kibana-käyttöliittymän tarjoamat visualisointimahdollisuudet tarjoavat nykymuotoisen operatiivisen monitoroinnin lisäksi mahdollisuuden tarkastella tietoa eri tavoin. Samoista integraatioista kirjoitettuihin lokisanomiin pystytään tekemään monimuotoisempia hakuja kuin aikaisemmin, sekä saadut hakutulokset pystytään visualisoimaan erilaisiin graafeihin. Täten teknisen monitoroinnin lisäksi pystytään saamaan näkyvyyttä myös bisnestoimintaan, ja tarkastelemaan esimerkiksi erilaisia kuormapiikkejä integraatioväylän, sekä muiden bisnestoiminnan kannalta tärkeiden järjestelmien toiminaan.
Kuvassa 4.1 on kuvattu integraatioväylän ja monitorointiratkaisun rakennetta uudistuksen jälkeen. Monitorointi irroitetaan integraatioväyläpalvelimelta, jossa se alkutilanteessa sijaitsee. Integraatioväyläpalvelimelle asennetaan ja konfiguroidaan Logstash-ohjelmisto, joka lukee Mule ESB integraatioväylän kirjoittamaa logia, ja lähettää eteenpäin monitorointipalvelimelle. Monitorointipalvelimella saa-



Kuva 4.1 Monitorointiratkaisun arkkitehtuurikuvaus

puvien logien tietovarastona toimii ElasticSearch [Gormley, 2015]. Monitorointipalvelimella on asennettuna myös Kibana-sovellus, joka esittää ElasticSearchista saadun hakutuloksen vastauksena saadut logitiedot. Käyttäjä seuraa monitoria Kibanan kautta, ja tätä kautta hän saa tehtyä erilaisia hakuja eri järjestelmien lähettämistä lokeista. Näin hän pystyy seuraamaan integraatioväylän sekä siihen liitettyjen järjestelmien toimintaa. Koska tietovarasto on eriytetty integraatioväylältä, voidaan tietovarastoon lähettää logitietoja myös muista järjestelmistä (Järjestelmät Y ja X), joiden välille on mahdollisesti rakennettu integraatioita integraatiopalvelimen kautta. Täten pystytään metatietojen avulla yhtenäistämään lokeja, ja seurata esimerkiksi saman työtilauksen kulkua koko prosessin läpi yli järjestelmärajojen.

4.3 Lokidatan luokittelu

Mule ESB-integraatioväylä käyttää järjestelmän toiminnan sisäiseen lokittamiseen log4j-lokituskomponenttia [Gupta, 2003]. Toiminnan aikana järjestelmään

kirjoitetaan erilaisia lokisanomia, jotka kertovat järjestelmän ja sen käyttämien komponenttien toiminnasta suorituksen aikana, sekä mahdollisista virheistä. Integraatioväylän toiminnan aikana yksittäinen liittymä kirjoittaa paljon rivejä esimerkiksi käytettävien adapterien toiminnasta ja statistiikasta. Integraatioiden operatiivisen monitoroinnin kannalta suurin osa sanomista ei ole olennaista tietoa. Esimerkiksi integraation lähdejärjestelemän ollessa SQL-tietokanta, ei integraation operatiivisen toiminnan monitoroinnin kannalta ole oleellista lokittaa tietoa jokaisesta onnistuneesta yhteydenottoyrityksestä ja onnistuneesta kyselystä tietokantaan - epäonnistuneet tietokantayhteyden muodostamiset sitä vastoin ovat operatiivisen toiminnan seuraamisen kannalta oleellista tietoa. Log4j-lokituskomponentti kirjoittaa lokiin eri tasoisia sanomia, lokitustasot määritellään avainsanoilla kuten INFO, WARN, ERROR tai DEBUG. INFO-tasoiset sanomat ovat nimensä mukaan järjestelmän toiminnasta tiedoksi saatettavaa tietoa, WARN-tason sanomat ovat varoituksia tilanteista jotka voivat aiheuttaa virheitä, ERROR-sanomat kertovat virheistä ja niin edelleen.
Integraatioväylän ollessa käynnissä, useat erilaiset yhteyskomponentit, sekä ajettavat integraatiot kirjoittavat jatkuvasti eri tasoisia sanomia lokiin. Myös integraation ajojen aikana lokiin kirjoitettavat sanomat kirjoitetaan samaan lokiin, ja useiden integraatiosuoritusten ollessa yhtäaikaa ajossa, muodostuu lokin lukeminen suoraan lokitiedostoista haasteelliseksi. Tähän halutaankin tukeutua useimmien ainoastaan virheiden selvitystilanteissa.
Aiemmin käytössä ollut Pulse-monitorointikomponentti on Mule-lokin lisäksi kirjoittanut sanomat relaatiotietokantaan, josta web-käyttöliittymän avulla pystyttiin helpommin hakemaan integraation operatiivisen monitoroinnin kannalta halutut tiedot. Näin kannan avulla pystyttiin helposti selvittämään missä integraatioissa on tapahtunut virheitä halutun ajanjakson aikana. Esimerkiksi Mule ESB-järjestelmän käyttämien adapterien toiminnan seuraaminen ei vanhan Pulse-käyttöliittymän kautta ollut mahdollista. Uudessa monitorointiratkaisussa Logstash voidaan konfiguroida lukemaan myös adapterien toimintaan liittyvät lokisanomat, jolloin integraatioiden operatiiviseen toimintaan liittyvien erikseen kirjoitettujen sanomien lisäksi on mahdollista seurata myös adapteritasolla tapahtuvien toimintojen määriä ja niissä tapahtuvia virheitä.

4.4 Käytännön toimenpiteet

Monitorointiratkaisun uudistus edellyttää muutoksia sekä integraatioväylän toteutukseen että koko integraatioratkaisun arkkitehtuuriin.

Integraatioväylä lokittaa useita eri tyyppisiä viestejä lokitiedostoihin, ja suurin osa väylän lokittamista viesteistä ei ole integraatioväylän operatiivisen toiminnan monitoroinnin kannalta olennaisia. Integraatiototeutukset kirjoittavat erilaisia viestejä integraatioiden suorituksen aikana, mutta lisäksi lokeissa on paljon erilaista järjestelmän komponenttien suoritukseen liityvää diagnostiikkadataa, joka ei useimmiten ole operatiivisen toiminnan kannalta kriittistä tietoa. Tätä tarkoitusta varten toteutukseen on lisättävä oma erillinen lokituskomponenttinsa, joka kirjoittaa prosessin eri vaiheissa (alku, loppu, virhe, tai jokin muu vaihe) operatiivisesti relevantit tiedot lokiin omassa formaatissaan, joka sisältää myös suorituksen aikaisia metatietoja. Täten saadaan helposti lokeista prosessoitua ainoastaan ne viestit, jotka ovat operatiivisen monitoroinnin kannalta relevantteja.

4.4.1 Logstash

Ylempänä kuvatussa monitorointiratkaisun kuvauksen mukaisesti integraatioväylän palvelimelle (sekä testi- että tuotantoympäristöön) on asennettu Logstash-ohjelmisto, joka kerää ja jäsentää rivejä lokitiedostoista, sekä tekee mahdollisesti erilaisia muunnoksia lokeista kerätylle datalle, sekä lähettää ne eteenpäin tietovarastoon, joka tässä casessa on siis ElasticSearch. Logstashiin konfiguroidaan luettavat lokit sekä niin kutsuttu grok-pattern, johon sopivat lokirivit poimitaan lokeista, muunnetaan JSON-muotoon ja lähetetään ElasticSearch-tietovarastoon REST-kutsua käyttäen [Rodriguez, 2008].

4.4.2 ElasticSearch

ElasticSearch on avoimen lähdekoodin ohjelmisto, joka toimii Logstashin jäsentämien lokitapahtumien hakukoneena ja tietovarastona. Tämän projektin yhteydessä ElasticSearch asennettiin omalle palvelimelle, jota käytetään sekä testi- että tuotantoympäristön lokivarastona. [Gormley, 2015]. ElasticSearch on todettu suorituskyvyltään vahvaksi järjestelmäksi suurten datamäärien käsittelyssä [Bai, 2013]. Kyseisessä asiakasprojektissa päivittäisiin lokitiedostoihin kirjoitetaan satoja tuhansia rivejä dataa, eikä ElasticSearchin toiminnassa ole toistaiseksi havaittu toimintaa häiritsevää hitautta, kun lokidataa on tallennettuna tietovarastoon yli kuuden kuukauden ajalta. Tavallisesti operatiivisen toiminnan kannalta yli kuusi kuukautta vanha data harvoin on relevanttia, ja tätä vanhempaa tietoa ElasticSearchin tietovarastosta tullaan myöhemmin hallitusti poistamaan.

4.4.3 Kibana

Kibana on web-käyttöliittymä, jonka avulla voidaan tehdä hakuja ElasticSearchiin tallennettuihin lokitapahtumiin, sekä luoda erilaisia visualisointeja haettavasta datasta helppokäyttöisen käyttöliittymän kautta. Käyttäjät pystyvät luomaan ja tallentamaan ElasticSearchiin useita erilaisia visualisointeja koostamaan luoduista visualisoinneista erilaisia näkymiä (dashboardeja). Käyttöliittymän avulla käyttäjät pystyvät tutkimaan lokitietoja ja niistä piirrettäviä visualisointeja, sekä porautumaan dataan aika- ja muilla rajauksilla.

4.5 Monitoroitavien viestin määrittely

Integraatioväylän monitorointiratkaisun uudistukseen lähtiessä tilanne oli se, että olemassaolevaa järjestelmää väylän toiminnan monitorointiin ja virhetilanteiden selvitykseen on käytetty jo yli vuoden ajan. Tänä aikana oli jouduttu selvittämään useita erilaisia virhetilanteita, joiden aiheuttajina olivat esimerkiksi häiriöt tietoliikenteessä, virheet integraatioiden toiminnan määrittelyssä sekä bugit (virheet) integraatioiden toteutuksissa. Näiden myötä on tunnistettu erilaisia tilanteita, joita on ollut hankalaa ennakoida tapahtuvaksi. Jälkijättöisesti ollaan pystytty lokitietoja tutkimalla tunnistamaan erilaisia tilanteita, jotka ovat edeltäneet virhetilanteita, mutta Pulse-monitorointiratkaisu ei pysty visualisoimaan virhetilanteita vaan ainoastaan tarjoamaan numeropohjaisia yhteenvetoja väylän lokeista valitulta ajanjaksolta (esimerkiksi tietynlaiset virheet edellisen 12 tunnin ajalta).

Koska Kibana-käyttöliittymä tarjoaa mahdollisuuden myös rakentaa erilaisia visualisointeja integraatioväylän lokimerkinnöistä, on tätä myöten tunnistettu tarve määritellä integraatioiden lokittamiin viesteihin myös muuta ajonaikaista metadataa jonka avulla olisi mahdollista tunnistaa ja korjata integraatioväylän operaatioissa tapahtuvia virhetilanteita nopeammin. Esimerkiksi havainnoimalla integraatioväylän läpi kulkevia viestimääriä, yllättäviä nousuja tai laskuja jonkin integroidun järjestelmästä saapuvien viestien määrässä, tai järjestelmien välisessä viiveessä (latency) viestien lähettämisen tai vastaanottamisen välissä.

Kehittäminen on ollut iteratiivinen prosessi, jossa erilaisia visualisointeja tutkimalla on todettu, että jonkin lisämääreen tallentaminen lokiviestien metatietoihin olisi hyödyllistä. Täten integraatiototeutuksiin on lisätty erilaista metatiedon tallentamista integraation suorituksen aikana, jotta ElasticSearchiin tallennetusta datasta saataisiin rakennettua mahdollisimman hyvin vianselvitystä tukevia

tietoja.

4.6 Logstashin konfigurointi ja lokiviestien formaatti

Mule-toteutuksia varten on kehitetty erillinen lokituskomponentti, jonka avulla Mule lokeihin kirjoitetaan määrätyn muotoisia lokirivejä integraatioprosessin aikana, josta Logstash jäsentää niin sanottuun grok-patterniin osuvat lokirivit, poimii lokiriviltä metadatat, ja muuntaa sanoman JSON -muotoon ja lähettää sen eteenpäin ElasticSearchille.
Alla on esimerkki Mule-toteutuskoodista, jossa asetetaan 'monitorParameters' -nimiseen muuttujaan avain-arvo pareina kyseisen integraation suoritukseen liittyviä parametreja ja metadataa, sekä kirjoitetaan lokisanomalle viesti integraation alkamisesta. Muuttujan sisältämät avain-arvoparit kirjoitetaan prosessin aikana jokaiselle lokitetulle sanomalle, jolloin yksittäisestä ERROR-virherivistä voidaan lukea myös kaikki prosessin aikana avain-arvopareiksi tallennetut tiedot, jolloin on mahdollista pelkästä virheviestistä lukea, mihin työtilaukseen kyseinen virhe on liittynyt, tai vaikkapa sanoman lähde- ja kohdejärjestelmät.
Esimerkki 7:

```
<set-variable variableName="monitorParameters"
  value="#[['woid':1234,'senderSystem':ESB]]" />

<monitor:start message="Integration XYZ started">
  <monitor:parameters ref="#[monitorParameters]" />
</monitor:start>
```

Esimerkin 7 mukaisesta toteutuskomponentista kirjoitetaan integraatioväylän alla olevan (esimerkki 8) mukainen lokirivi (luettavuuden vuoksi rivinvaihtoineen, varsinaisessa lokitiedostossa tieto on yhdellä rivillä)
Esimerkki 8:

```
INFO 2017-12-04 08:56:56,727 [[msb].wostatusupdate/msb.from.stage1.1787]
fi.solita.mule.connector.monitor.MonitorConnector:
Service=wostatusupdate%2Fmsb.from&
woid=1234&senderSystem=ESB&InstanceId=514709e3-d8c0-11e7-ad46-00505685722d&
LogMessage=Received+work+order+status+update&Application=msb&State=Start
```

Logstash konfiguroidaan tässä tapauksessa niin, että se lukee Mulen kirjoittamaa lokia, ja poimii ja käsittelee sieltä ylläolevaa muotoa noudattavat rivit. Esimerkissä 9 on esitty vaadittu konfiguraatio:

Esimerkki 9:

```
1 input {
2   file {
3     sincedb_path => "/path/to/mule-log-sincedb"
4     path => "/var/log/mule.log"
5     start_position => "beginning"
6 }
7 filter {
8   grok {
9     match => { "message" =>
10     "^%{LOGLEVEL} %{TIMESTAMP_ISO8601:LogTimestamp}
11     (?<SourceThread>.*)
12     fi.solita.mule.connector.monitor.MonitorConnector:
13     (?<Metadata>.*)"
14     }
15   }
16   date {
17     match => ["LogTimestamp",
18               "yyyy-MM-dd'T'HH:mm:ss,SSSZ",
19               "yyyy-MM-dd HH:mm:ss,SSS"]
20   }
21   mutate {
22     strip => "Metadata"
23   }
24   kv {
25     field_split => "&"
26     source => "Metadata"
27   }
28   if "_grokparsefailure" in [tags] {
29     drop {}
30   }
31 }
32 output {
33   elasticsearch {
34     action => "index"
35     workers => 5
```

```
36      document_id => "%{LogTimestamp}%{Service}%{InstanceId}%{State}"
37      hosts => "12.34.56.78:1234"
38      index => "logstash-%{+YYYYMM}"
39    }
40    stdout {
41      codec => rubydebug
42    }
43 }
```

Rivillä 1 määritetään input-elementti, eli tässä tapauksessa luettava lokitiedosto Logstashille (polku ja tiedoston nimi). Start_position parametrilla määritetään että tiedosto luetaan aina alusta alkaen. Mule-lokit muodostuvat niin, että nykyisen päivän loki kirjoitetaan tiedostoon nimeltä "mule.log". Vuorokauden vaihtuessa edellisen päivän tiedosto uudelleennimetään noudattaen muotoa "mule-YYYY-mm-DD.log". Tätä toimepidettä kutsutaan lokien rotaatioksi, ja Logstash jatkaa edelleen lokien lukemista päivän vaihtuessa tiedostosta "mule.log", tunnistaa, että kyseessä on uusi tiedosto ja alkaa lukemaan sitä alusta lähtien. Näin lokit arkistoidaan myös väylän levylle päiväkohtaisesti omiin tiedostoihinsa. Sincedb_path-parametrilla määritetään polku niinkutsuttuun sincedb-tiedostoon, jonka avulla Logstash pitää sisäisesti kirjaa siitä, mihin asti sen päivän lokitiedostoa on luettu. Mikäli jostain syystä Logstash halutaan sammuttaa hetkeksi, sincedb-tiedoston avulla lokitiedoston parsimista voidaan uudelleen käynnistämisen jälkeen jatkaa samasta kohtaa, mihin asti tiedosto oli luettu Logstash sammutettaessa.

Riviltä 7 eteenpäin filter-elementti, eli millaiset rivit lokitiedostosta halutaan poimia. Riveillä 10-13 on määritelty grok-hahmo, johon sopivat lokirivit suodattuvat joukosta esiin. Lokirivin ensimmäisenä odotetaan olevan lokituksen aste (kuten INFO, DEBUG, ERROR), jonka jälkeen on yksi välilyönti ja aikaleima. Aikaleiman jälkeisestä välilyönnistä seuraavana oleva merkkijono poimitaan arvoksi nimeltä SourceThread, jonka jälkeen odotetaan merkkijonoa “fi.solita.mule.connector.monitor.MonitorConnector:”, jonka jälkeisestä välilyönnistä loppuun odotetaan löytyvän lokisanomalle avain-arvopareina kirjoitettu metadata. “Mutate”-konfiguraatio poistaa Metadata-kentästä whitespacen. “Date”-elementin match muuntaa “LogTimestamp” -elementtiin jäsennetyn aikaleiman yksiselitteiseksi aikaleimaksi, kentän jälkeen annettavia mahdollisia muotoja käyttäen (voidaan antaa useita eri formaatteja, joiden avulla tulkita aikaleima). “Kv” muuntaa metadatan sisällön avain-arvopareiksi, käyttäen &-merkkiä erottaakseen

avain-arvoparit toisistaan. Lopuksi if-ehto jättää huomiotta rivit, joita ei saada tulkittua määriteltyä filtteriä käyttämällä.
"Output"-elementissä on määritelty output, johon filtteriin osuvat rivit lähetetään, eli IP-osoite jossa ElasticSearch kuuntelee saapuvia JSON-sanomia ja indeksi johon sanoma halutaan talletettavan. Stdout-elementissä on määritetty muoto, jossa Logstash kirjoittaa omaa toimintalokiaan.
Täten Logstash jäsentää ja normalisoi logeista halutut rivit, ja jättää huomiotta (drop) rivit, joiden tulkinta grok-hahmoa vastaan ei onnistu. Näitä rivejä ei logiteta Logstashin omiin lokeihin, eikä niitä lähetetä myöskään eteenpäin ElasticSearchiin. On tilanteita, joissa voi olla tarpeellista lähettää ElasticSearchiin haettaviksi myös lokirivejä, joiden jäsentäminen ennalta määrättyjen sääntöjen mukaan ei onnistu.
Onnistuneen jäsentämisen myötä Logstash saa jäsennettyä lokirivit, ja muodostaa niistä sen jälkeen JSON-muotoisen sanoman, joka lähetetään REST-kutsuna eteenpäin ElasticSearchille. Esimerkissä 10 on esitetty tällainen JSON-sanoma:
Esimerkki 10:


```
{
    "message": "INFO  2018-01-18 16:02:22,381
    [[msb_prod].woupdate/msb.from.stage1.3397]
    fi.solita.mule.connector.monitor.MonitorConnector:
    Service=workorderstatusupdate%2Fmsb.from&woid=ID-12345
    &InstanceId=33807e33-fc58-11e7-8b33-00505685722d
    &LogMessage=Received+work+order+status+update&
    Application=msb-prod-strict&State=Success",
    "@version": "1",
    "@timestamp": "2018-01-18T14:02:22.381Z",
    "path": "/path/to/logfile.log",
    "host": "MULE_ESB",
    "LogTimestamp": "2018-01-18 16:02:22,381",
    "SourceThread": "[[msb-prod].workorderstatusupdate/msb.from.stage1]",
    "Metadata": "Service=woupdate%2Fmsb.from&woid=ID-12345
    &InstanceId=33807e33-fc58-11e7-8b33-00505685722d
    &LogMessage=Received+work+order+status+update&Application=msb_prod
    &State=Success",
    "Service": "woupdate%2Fmsb.from",
    "woid": "ID-12345",
```

```
    "InstanceId": "33807e33-fc58-11e7-8b33-00505685722d",
    "LogMessage": "Received+work+order+status+update",
    "Application": "msb_prod",
    "State": "Success"
  }
```

Edellä mainittujen askelien seurauksena Logstashin avulla on saatu Mule-lokeista jäsennettyä tapahtumat ja normalisoitua ne niin, että ne on pystytty lähettämään ElasticSearchiin, ja lokeista kerätty tieto on nyt haettavissa kannasta. Käyttäjät voivat täten suorittaa hakuja dataan sekä muodostaa erilaisia visualisointeja integraatioväylän toiminnasta.

4.7 Datan visualisointi

Kun data on saatu kirjoitettua lokeille, normalisoitu ja lähetetty ElasticSearchiin, päästään dataa käyttämään monitorointitarkoituksessa. Kibana -käyttöliittymän avulla ElasticSearchiin talletettua dataa voidaan hakea nopeasti, ja halutuloksista pystytään muodostamaan helposti erilaisia visualisointeja. Asiakkaan operatiivisen toiminnan kannalta on pystytty tässä vaiheessa muodostamaan visualisointeja, joiden avulla saadaan reaaliaikaista näkymää työtilausliikenteen määriin, mahdollisiin virheisiin, sekä pystytään nopeasti havaisemaan mikäli johonkin tiettyyn työkohteeseen liittyen integraatioväylällä alkaa tapahtua epätavallisia määriä toimintaa tai virheitä.
Visualisointeja luodaan suoraan Kibana -käyttöliittymän kautta ja visualisointeja voidaan koostaa useisiin yleisnäkymiin, jotta saadaan nopeammin ja selkeämmin muodostettua yleiskuva integraatioväylän sekä muiden monitoroitavien järjestelmien operatiivisesta toiminnasta. Luotujen visualisointien avulla on myös mahdollista porautua syvemmin näkyvillä olevaan dataan, sekä tehdä erilaisia rajauksia. Muutokset näkyvät ElasticSearchin tietovarastosta tehdyissä hauissa, ja tilanteen selvittämisen kannalta olennaiset lokiviestit on mahdollista haarukoida nopeasti ja helposti visuaalisen käyttöliittymän kautta.
Käyttöönoton aikana monitorointijärjestelmien asentamisen ja käyttöönoton lisäksi on järjestetty yhteisiä työpajoja, joiden aikana Kibanan ja Logstashin käyttöä koulutettiin asiakasorganisaation työntekijöille. Täten asiakasorganisaatiolla on valmiudet luoda omia visualisointeja ElasticSearchiin lähetetyistä logiriveistä, sekä lisätä muita järjestelmiä monitoroinnin piiriin asentamalla ja konfiguroimalla Logstash muihin järjestelmiin.

4.8 Monitoroinnin vaiheittainen käyttöönotto

Monitorointiuudistuksen käyttöönottovaiheessa nykyinen integraatioväylä on ollut asiakkaan toiminnassa käytössä jo lähes 4 vuoden ajan. Tänä aikana olemassaoleviin integraatioihin on tullut useita muutoksia, sekä kokonaan uusia integraatiota. Yhteensä erilaisia integraatioita on käytössä jo lähes 100 kappaletta. Kyseiseen joukkoon sisältyy kriittisyysasteeltaan erilaisia integraatioita. Liiketoiminnan kannalta kriittisiä integraatioita pitää pystyä monitoroimaan koko käyttöönoton ajan, tästä johtuen uuden monitoroinnin käyttöönotto projektissa on tehtävä hallitusti ja vaiheittain.
Ensimmäinen vaihe monitoroinnin uudistamisessa on lisätä uusi monitorointikomponentti tärkeimpiin integraatioihin. Tämän lisäksi on luotava Kibanaan kyseisten integraatioiden toiminnan seurannan kannalta soveltuvia visualisointeja, jotta integraatioiden toimintaa ja virheiden määrää pystytään seuraamaan sen kautta. Monitorointiuudistuksen tavoitteena on pystyä nopeammin havaitsemaan ja tunnistamaan virheet aiheuttava juurisyy, sekä selvittää virheen korjaamiseksi vaadittavat avaintiedot.

4.9 Jatkokehitys

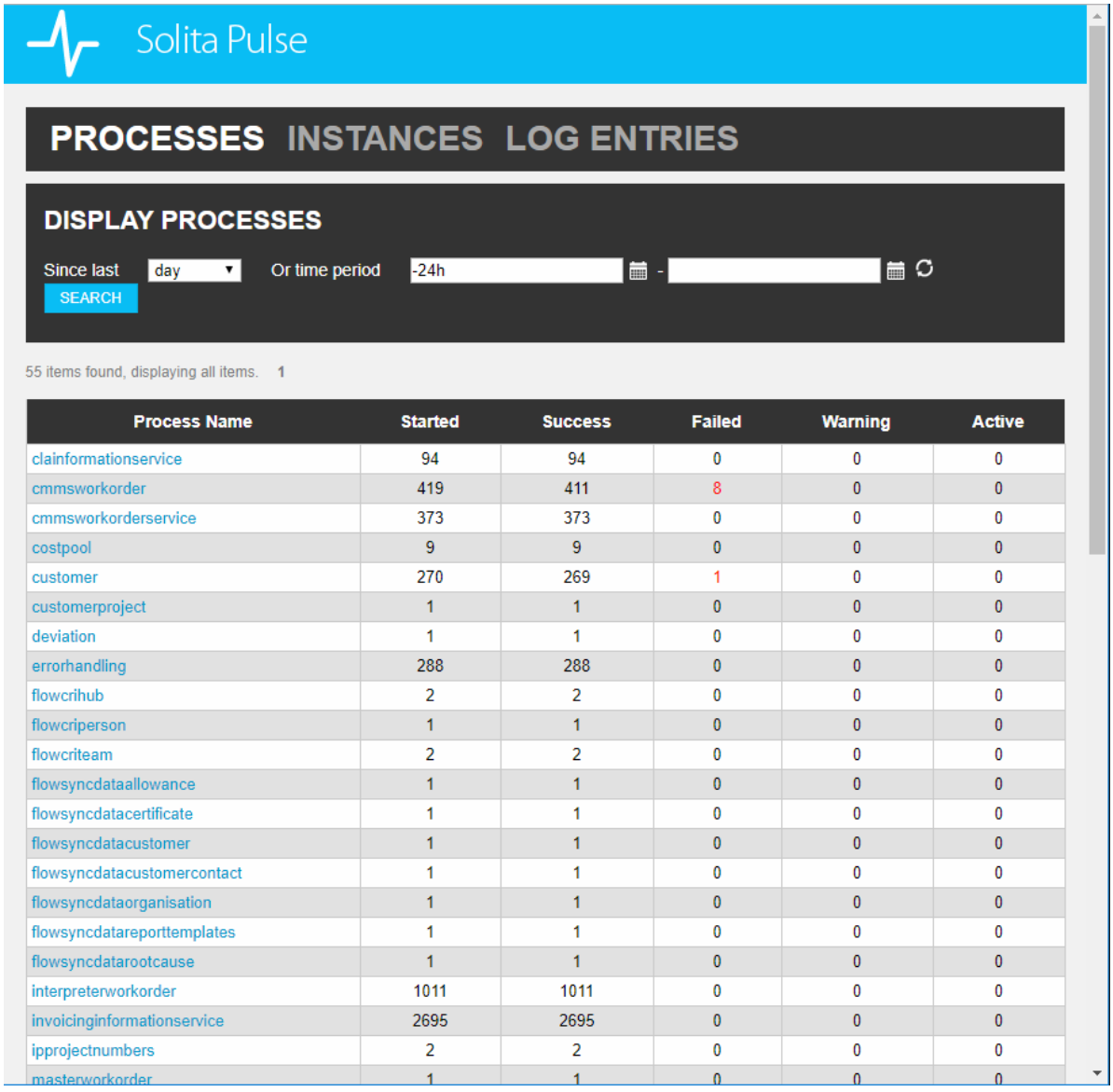
Tämän projektin aikana integraatioväylän monitorointiuudistuksessa tehdyt toimenpiteet ovat olleet uudistuksen vaatimien teknisten edellytyksien käyttöönotto, sekä integraatioväylän monitorointikomponentin käyttöönotto ja lisääminen osaan olemassaolevin integraatioita. Monitorontia tullaan jatkossa laajentamaan siten, että kaikissa integraatioväylälle toteutettavissa uusissa integraatioissa käytetään tämän projektin aikana käyttöön otettua uutta monitorointikomponenttia, ja täten integraatioiden suorituksen aikana kirjoitettavat lokimerkinnät ovat uuden monitorointiratkaisun piirissä.
Olemassaoleviin integraatioihin muutoksia toteuttaessa tullaan niin tarvittaessa muun muutostyön ohessa myös lisäämään uusi monitorointikomponentti, jolloin olemassaolevat integraatiot saadaan uuden monitoroinnin piiriin. Näitä integraatioita on olemassa suuri määrä, jolloin on integraatiokohtaisesti pohdittava, saadaanko tällä muutoksella riittävästi hyötyä, jotta muutokset kannattaa toteuttaa. Lisäksi ajan saatossa osa integraatiosta on muuttunut tarpeettomiksi, ja tullaan mahdollisesti poistamaan kokonaan. Monitorointiuudistuksia tullaan siis toteuttamaan muun refaktoroinnin yhteydessä tarpeen mukaan.
Lisäksi nyt on mahdollista kerätä uuden monitorointipalvelimen lokivarastoon

lokitapahtumia myös järjestelmärajojen yli. Tässä kontekstissa ainoastaan integraatioväylä ja monitorointipalvelin ovat järjestelmiä, joihin minulla on ollut pääsy. Muut asiakkaan liiketoiminnassa mukana olevat järjestelmät ovat toisten osapuolten toimittamia, joten lokitusmuutokset näihin tullaan toteuttamaan kolmannen osapuolen tai asiakkaan toimesta. Uuden monitorointiratkaisun myötä avautuu mahdollisuus saada näkyvyyttä myös muiden järjestelmien lokeista, ja tämän mahdollisuuden hyödyntäminen on ollut keskusteluissa mukana.

5 Vertailu

Tässä luvussa esitellään aikaisemman Pulse-monitorointiratkaisun tarjoamia seurantanäkymiä kuvakaappauksien avulla, sekä uuden monitorointijärjestelmän avulla luontuja näkymiä.

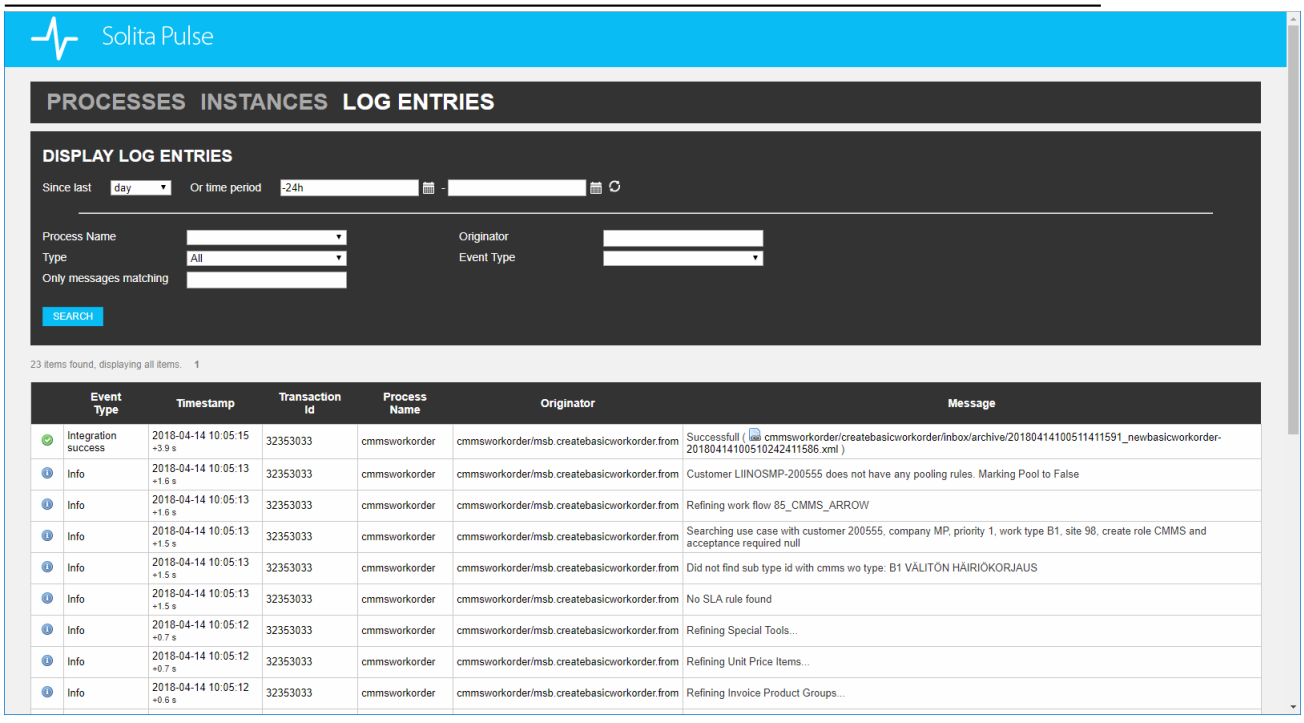
5.1 Alkuperäisen monitorointijärjestelmän yleisnäkymä



Kuva 5.1 Alkuperäisen monitorointijärjestelmän overview-näkymä

Kuvassa 5.1 on esitetty alkuperäisen monitorointijärjestelmän yleisnäkymä, jossa näytetään yhteenveto halutun ajanjakson ajetuista integraatioprosesseista. Sarakkeisiin on tarkennettuna integraatiokohtaisesti kuinka monta kertaa integraatio on käynnistynyt valitulla ajanjaksolla, kuinka monta kertaa prosessi on mennyt läpi onnistuneesti, montako prosessia on ajossa juuri tällä hetkellä, montako virhettä integraatiossa on tapahtunut, sekä montako varoitusta kyseisen integraation suorituksien aikana on noussut. Rajaamalla kuvassa näkymää alasvetovalikkoa, tai haluttua alku- ja loppuhetkeä muokkaamalla järjestelmä päivittää näytetyn näkymän valitun ajanjakson tiedoilla. Yksittäisen integraation nimestä klikkaamalla pääsee tarkastelemaan tarkemmin rivitasolla kyseisen integraation lokittamia monitorisanomia.

5.2 Rivitason näkymä

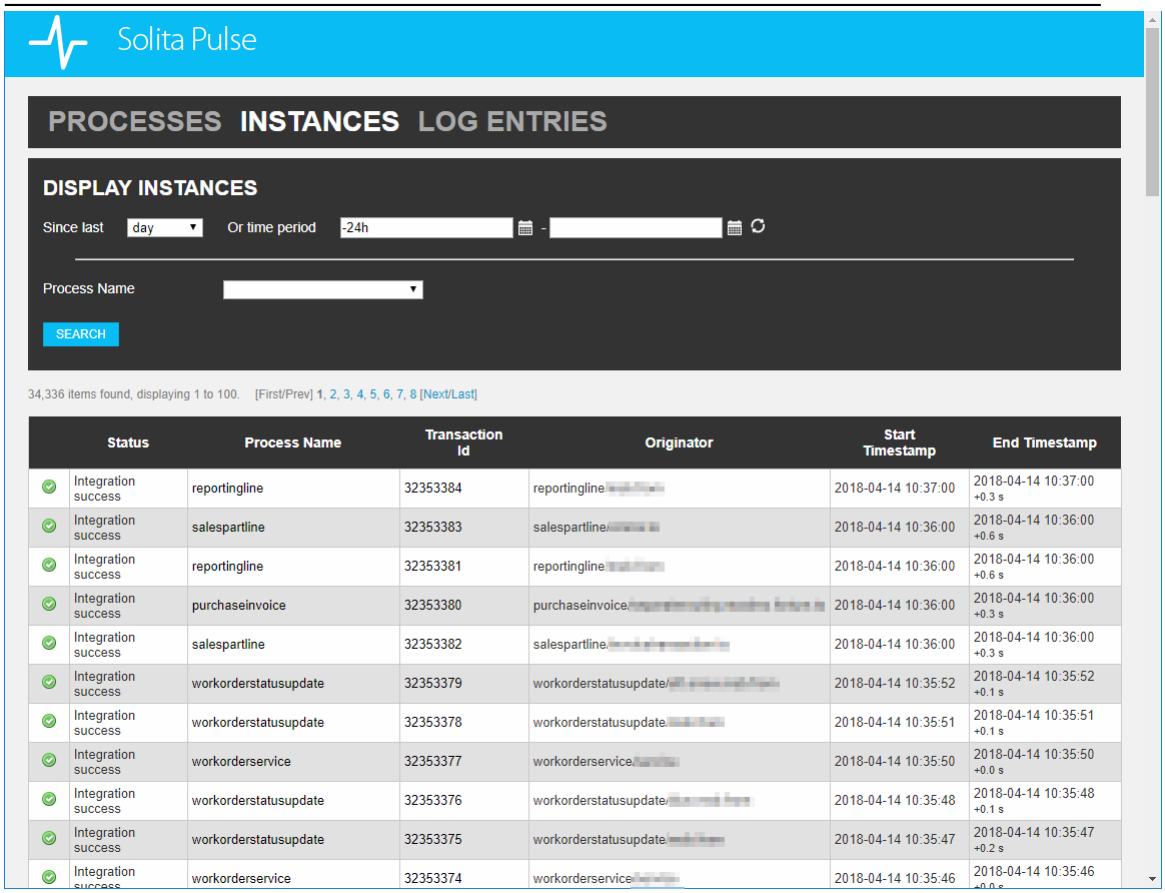


Kuva 5.2 Alkuperäisen monitorointijärjestelmän rivitason näkymä

Kuvassa 5.2 on esitetty yksittäisen integraation rivitason sanomia. Mikäli integraatiosuoritus on mennyt virheeseen, on tätä kautta mahdollista tutkia integraatioprosessin ajon aikana monitoriin kirjoitettuja sanomia, ja tutkia integraation

suoritusta, ja mikä virheen on mahdollisesti aiheuttanut. Näkymässä on mahdollista rajata rivejä niiden tyypin mukaan, integraation suorituksen aikaisten yksittäisten komponenttien (originator) mukaan, tai hakea sanomia joissa esiintyy jokin tietty merkkijono.

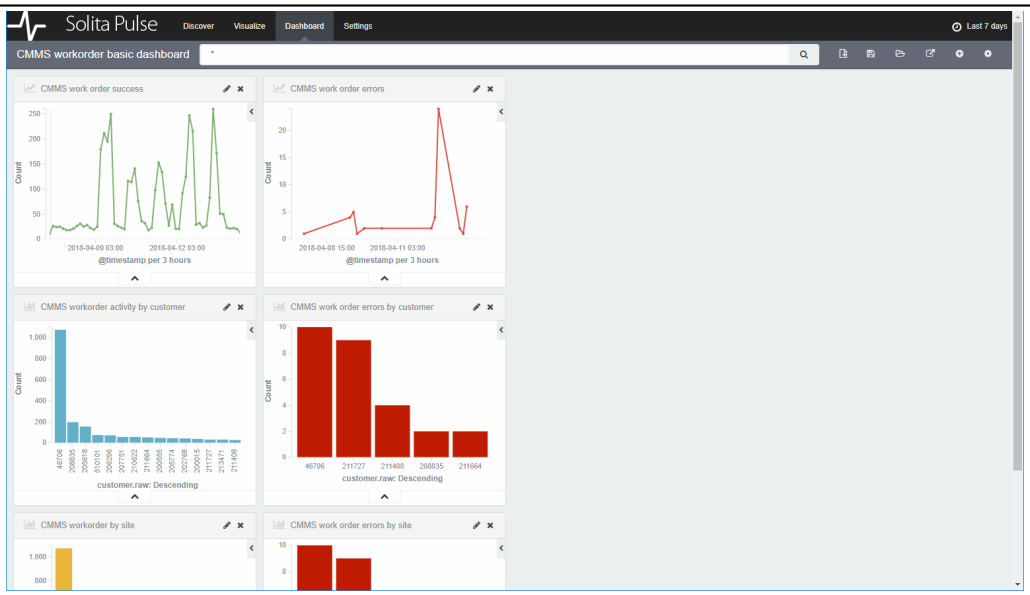
5.3 Prosessinäkymä



Kuva 5.3 Pulse-monitorointijärjestelmän instanssitason yhteenvetonäkymä

Kuvassa 5.3 on esitetty prosessinäkymä, jossa on eriteltynä prosessikohtaisesti viimeisin sanoma (Success tai Error). Pulse-monitorointijärjestelmä tarjoaa ensisijaisesti helppokäyttöisen käyttöliittymän, jonka avulla tiettyjen integraatioiden toiminnan seurantaan on helppo tehdä erilaisia hakuja, ja esittää hakujen tulokset rivitettynä lokitietoja. Lisäksi tarjotaan yksinkertaisia yhteenvetoja integraatioväylän toiminnasta, mutta monitorointijärjestelmän tarjoama näkyvyys integraatioväylän toimintaan tai toiminnan analytiikkaan on hyvin rajoitettua.

5.4 Uudistetun monitorijärjestelmän dashboard

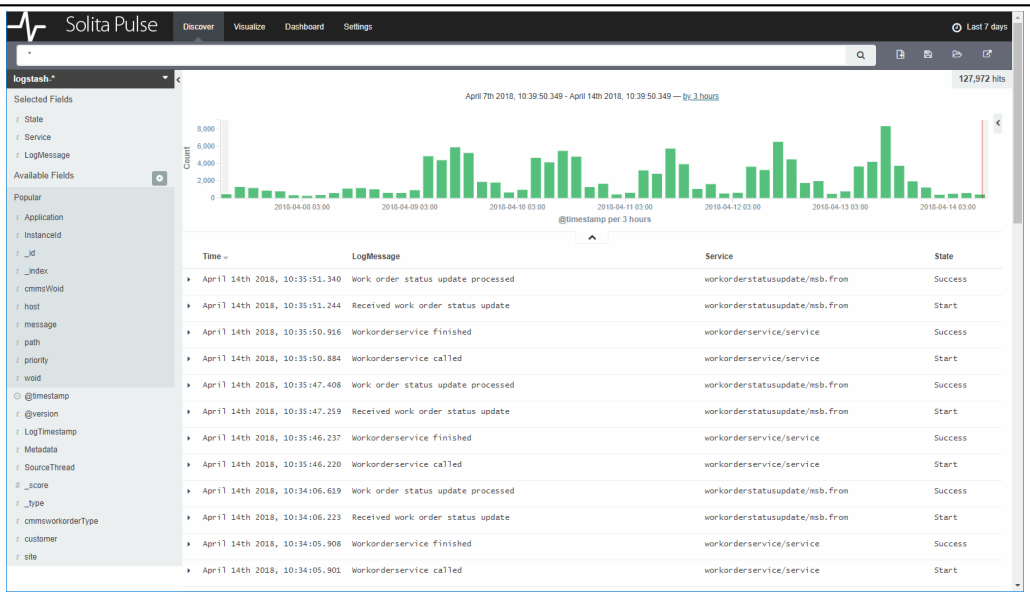


Kuva 5.4 Uudistetun monitorointijärjestelmän dashboard-yleisnäkymä

Kuvassa 5.4 on esitetty uudistetun monitorointijärjestelmän yksi dashboard-näkymä, johon on koostettu eri visualisointeja. Visualisoinneissa näkyy onnistuneet ja epäonnistuneet integraatiosuoritukset, ja suorituksista piirretään graafeja. Myös virheet on visualisoitu per asiakas, jonka kohdalla virhe on tapahtunut. Monitorointijärjestelmään voidaan luoda uusia visualisointeja, ja koostaa luoduista visualisoinneista uusia dashboard-näkymiä, joiden avulla saadaan uudenlaista näkyvyyttä integraatiojärjestelmän toimintaan. Mikäli väylän operatiivisen toiminnan monitoroinnissa tapahtuu muutoksia, ei tämä välttämättä edellytä lainkaan muutoksia integraatioväylän toimintaan. Olemassa olevasta datasta voidaan tehdä uusia hakuja ja näitä hakutuloksia voidaan visualisoida tarpeen mukaan.
Kuvan 5.4 mukainen dashboard on osoittautunut hyödylliseksi esimerkiksi sellaisessa käyttötilanteessa, jossa vanhassa monitorissa on huomattu tietyssä integraatioprosessissa tapahtuneen epätavallisen suuri määrä virheitä 24 tunnin ajanjaksolla. Oikean ylänurkan "CMMS work order errors-graafista pystyttiin nopeasti toteamaan kellonaika, jolloin virheily oli alkanut, sekä rajaamaan virheiden syntyneen noin viiden minuutin mittaisella ajanjaksolla sekä että virheiden syntyminen oli tämän jälkeen loppunut ja integraation toimineen tästä eteenpäin normaalisti. Rajaamalla käyttöliittymästä kyseinen viiden minuutin ajanjakso tarkasteluajanjaksoksi pystyttiin myös nopeasti toteamaan virheiden joh-

tuneen SFTP-palvelimen liittyvästä yhteysvirheestä. Vanha monitori ei tarjonnut mahdollisuutta vastaavan kaltaiseen näkyvyyteen integraatioväylän toimintaan, ja samojen asioiden toteaminen vaatii integraation toimintalokin läpikäyntiä, sekä virheiden aikaleimojen erityistä tarkastelua. Tällä tavoin uusi monitorointijärjestelmä tarjoaa uudenlaista näkyvyyttä integraatioväylän toimintalokiin, joka nopeuttaa virhetilanteiden selvittämistä.

5.5 Uudistetun järjestelmän Discover-näkymä

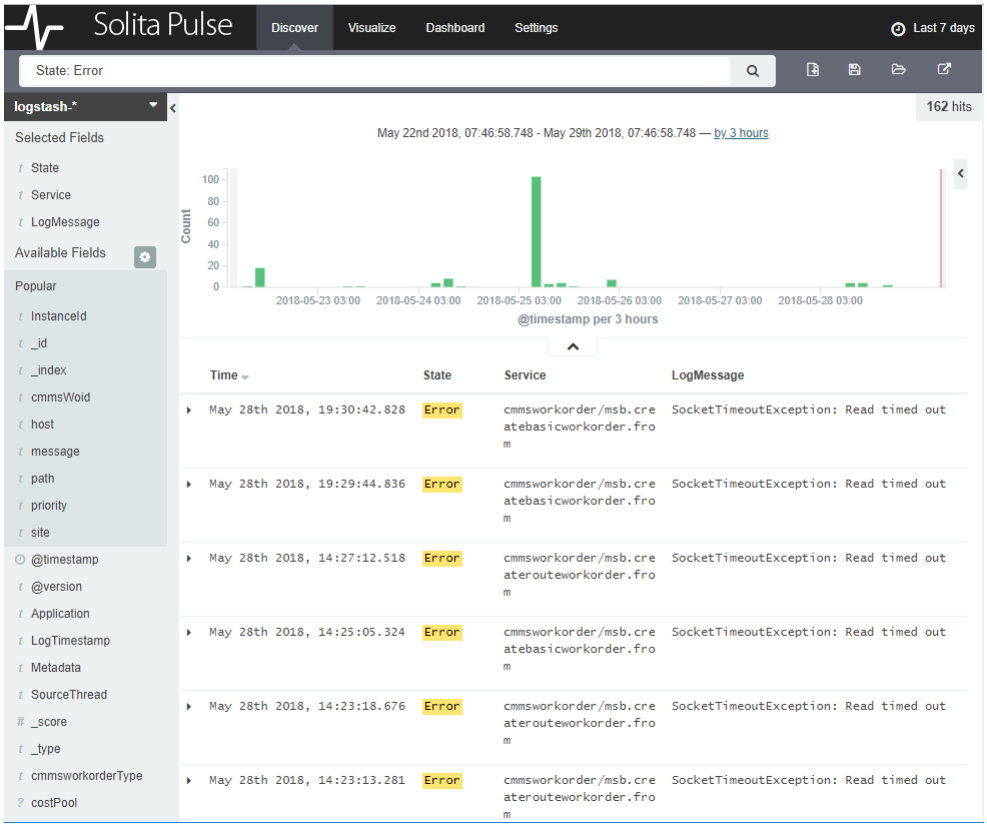


Kuva 5.5 Uudistetun monitorointijärjestelmän discover-näkymä

Kuvassa 5.5 on esitetty uuden monitorointijärjestelmän discover-näkymä, jossa on graafi johon piirtyy yksinkertaisesti ElasticSearch-järjestelmän vastaanottamat lokirivit aikajanalla, sekä listanäkymä, jonka avulla voidaan selata ja tutkia järjestelmän vastaanottamia lokitapahtumia rivitasolla.
Uusi monitorointijärjestelmä siis tarjoaa monipuolisia visualisointimahdollisuuksia, joiden avulla integraatioväylän kirjoittamaa prosessilokia voidaan tarkastella. Uusia visualisointeja voidaan helposti luoda tarpeen mukaan, jolloin monitorointijärjestelmää käyttämällä voidaan helpommin luoda esimerkiksi muuttuviin bisnestarpeisiin soveltuvia visualisointeja. Erilaiset visualisoinnit tarjoavat uudenlaista näkyvyyttä samalla tavalla lokitettuun dataan.
Kuitenkin Discover-näkymä tarjoaa vastaavat mahdollisuudet integraatioväylän monitorointiin, kuin aiempi monitorointiratkaisu. Integraatioväylän kirjoittamia

sanomia pystytään rajaamaan aikavälin perusteella, sekä sanomat kirjoittaneen integraatioprosessin perusteella. Uusi monitorointijärjestelmä siis tarjoaa mahdollisuuden tarkastella integraatiojärjestelmän toimintaa myös rivitasolla, mikäli näin halutaan.
Uusi monitorointijärjestelmä tarjoaa käyttäjälle mahdollisuuden räätälöidä rivitasolla näkyviä tietoja. Kuvakaappauksella vasemmassa laidassa olevaan listaan populoidaan hakutuloksissa esiintyvät eri kentät, joita käyttäjä voi tarpeen mukaan lisätä tai poistaa listattaviin riveihin. Täten käyttäjillä on aikaisempaan monitorointiratkaisuun verrattuna paremmat mahdollisuudet myös räätälöidä hakuun osuvien hakutulosten esitystapaa käyttötarpeen mukaan. Esimerkissä näkyy, että esitettäviksi sarakkeiksi on valittu aikaleima, lokitettu selkokielinen sanoma, palvelu, sekä viestin tila.

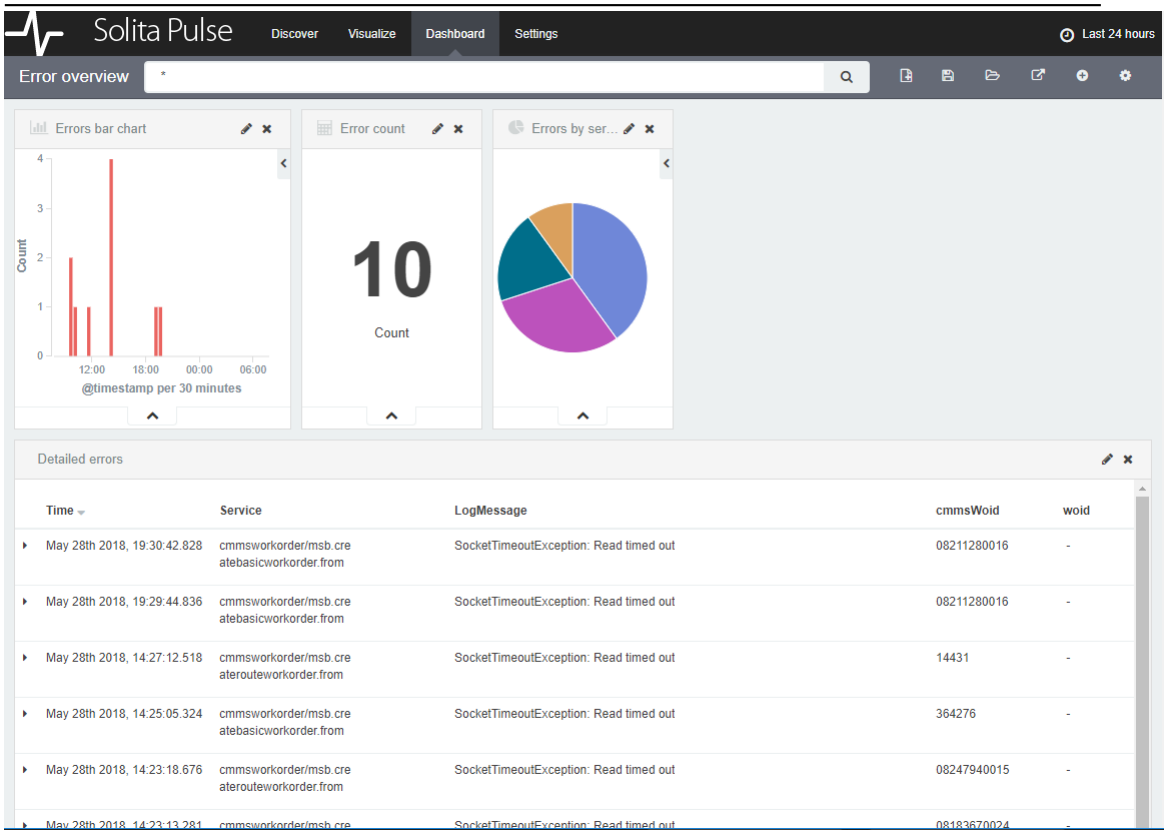
5.6 Uudistetun järjestelmän hakunäkymä



Kuva 5.6 Virheiden haku Discover-näkymässä

Kuvassa 5.6 esitetään, kuinka Kibana-käyttöliittymän hakukenttää käyttämällä

voidaan hakea ElasticSearchista haluttuja lokirivejä. Hakukenttä käyttää Lucene-hakukielen syntaksia [McCandless, 2010] ja esimerkkikuvassa on haettu kaikki lokirivit ElasticSearchista halutulta tarkasteluväliltä, jotka löytyvät hakuehdolla "State: Error". Discover-näkymän pylväsdiagrammi päivittyy automaattisesti, ja kuvaa käyttäjälle milloin ja minkä verran hakutulokseen osuvia tapahtumia löytyy tietovarastosta, sekä esittää myös löytyneet tiedot käyttäjälle listausnäkymässä. Tehty haku voidaan tallentaa Kibanaan, ja hakua voidaan käyttää visualisointien pohjana.



Kuva 5.7 Yleiskäyttöinen dashboard virheille

Kuvassa 5.7 on luotu 4 eri visualisointia kuvassa 5.6 esitellyn haun perusteella. Samaa hakua käyttäen pystytään virheistä esittämään erilaista dataa, jonka avulla on mahdollista saada nopeasti selkeä yhteenveto tietyn aikavälin virheistä, joita integraatioväylällä on tapahtunut. Näkymä esittää selkeästi käyttäjälle pylväsdiagrammin (aika y-akselina), josta voi nähdä milloin virheitä on tapahtunut ja kuinka paljon. Lisäksi ylärivin oikeanpuoleisin visualisointi "Errors by service"kertoo kuinka virheet ovat jakautuneet eri prosesseihin. Vanha monitorointi

ei tarjonnut vastaavanlaista näkyvyyttä integraatioväylän virheisiin, ainoastaan listauksen virheiden määrästä per integraatio. Operatiivista toimintaa monitoroivat käyttäjät voivat pitää tämän näkymän auki näyttöpäätteillään, ja uudet virheet eri integraatiossa sekä erilaiset piikit virheiden syntymisen määrässä pystytään havaitsemaan nopeasti. Lisäksi saadaan nopeasti vahvistus, kun virheiden määrä lähtee laskemaan tai loppuu kokonaan. Täten monitorointijärjestelmän uudistus tuo uudenlaista näkyvyyttä monitoroitavaan järjestelmään, sekä lisäarvoa operatiivisen toiminnan seurantaan.

6 Jatkotutkimus

Integraatioväylän uudistettu monitorointijärjestelmä on operatiivista toimintaa seuraaville loppukäyttäjille uusi tietojärjestelmä joka on tarkoitettu työssä käytettäväksi, jolloin on aiheellista tutkia millaiseksi loppukäyttäjät kokevat uuden järjestelmän. Technology acceptance model (TAM) on teoria jonka avulla pyritään mallintamaan kuinka käyttäjät omaksuvat ja käyttävät uusia tietojärjestelmiä [Davis, 1985]. Mallin mukaan uusia tietojärjestelmiä arvioitaessa käyttäjiin vaikuttavat useat eri tekijät jotka vaikuttavat käyttäjien asenteisiin järjestelmää kohtaan. Nämä asenteet vuorostaan vaikuttavat siihen millä tavoin ja kuinka paljon tietojärjestelmää tullaan käyttämään.

Tulevaisuudessa käyttäjätyytyväisyyttä uuteen monitorointijärjestelmään tullaan tutkimaan Davisin kehittämää kyselyä käyttämällä [Davis, 1989]. Kysymykset esitellään liitteessä 1. Kyselykaavakkeeseen käyttäjät arvioivat 12 kysymykseen vastaamalla kuinka hyödylliseksi (perceived usefulness) ja helppokäyttöiseksi (perceived ease of use) järjestelmä koetaan. Davisin tietojärjestelmän hyödyllisyyttä ja helppokäyttöisyyttä mittaavan kyselyn kysymykset on jaoteltu tasan kahteen eri osuuteen, kuusi per osuus. Vastaajat vastaavat kuinka todennäköisinä tai epätodennäköisinä pitävät kyselyn väitteiden toteutumista omassa työskentelyssään (vastaukset numeroarvoilla yhdestä seitsemään). Osuudet ovat "perceived usefulness"ja "perceived ease of use"vapaasti käännettynä "koettu hyödyllisyys"ja "koettu helppokäyttöisyys". Koetulla hyödyllisyydellä Davisin mukaan mitataan kuinka vahvasti vastaaja uskoo järjestelmän parantavan heidän työsuoritustaan. Koetulla helppokäyttöisyydellä tarkoitetaan kuinka vahvasti käyttäjä uskoo että järjestelmän käyttö työtehtävissä ei vaadi lisäponnistelua.

Monitorointijärjestelmän uudistus tarjoaa uudenlaista näkyvyyttä siihen liitettyjen järjestelmien toimintaan. Mikäli operatiivista toimintaa valvovat loppukäyttäjät eivät koe että uusi järjestelmä tarjoaa etua operatiivisen toiminnan seurantaan ja on riittävän helppokäyttöinen, ei järjestelmäuudistuksella ole saavutettu haluttua lisäarvoa järjestelmien monitorointiin. Kaikkia uudistetun monitorointijärjestelmän tarjoamia mahdollisuuksia ei ole vielä tässä vaiheessa voitu ottaa käyttöön. Monitorointijärjestelmä monitoroi tällä hetkellä ainoastaan Mule ESB-integraatioväylän lokeja. Käyttäjätyytyväisyyden arviointi on ajankohtaista sitten kun monitorointiratkaisuun on saatu liitettyä useampia järjestelmiä, eli kun järjestelmärajat ylittävä näkyvyys on saavutettu.

7 Yhteenveto

Tässä pro gradu- työssä olen esittänyt kuinka organisaation toiminnassa keskeisessä osassa toimivan integraatioväylän toiminnan monitorointia on mahdollista hajauttaa ELK-stackin (ElasticSearch, Logstash, Kibana) teknologioita käyttämällä. Tietojärjestelmän toimintalokien monitoroinnin hajauttaminen, sekä useamman kuin yhden tietojärjestelmän monitorointi yhden käyttöliittymän kautta on mahdollista toteuttaa tällä tavoin. Olen myös havainnollistanut, kuinka tietojärjestelmän kirjoittamiin toimintalokeihin on näin mahdollista saada uudenlaista näkyvyyttä, sekä esittänyt kuinka organisaatio voi mahdollisesti hyötyä tällaisesta näkyvyydestä.

Olen osoittanut että vastaavan monitorointijärjestelmän lisääminen jo olemassa oleviin järjestelmiin on mahdollista toteuttaa ilman muutoksia itse järjestelmään, jonka toimintaa tällä tavalla halutaan seurata.

Viiteluettelo

[Bai, 2013] Jun Bai. Feasibility analysis of big log data real time search based on hbase and elasticsearch. In *Natural Computation (ICNC), 2013 Ninth International Conference on*, pages 1166–1170. IEEE, 2013.

[Barrett & Silverman, 2001] Daniel J Barrett & Richard E Silverman. *SSH, the Secure Shell: the definitive guide*. "O'Reilly Media, Inc.", 2001.

[Bernstein & Haas, 2008] Philip A Bernstein & Laura M Haas. Information integration in the enterprise. *Communications of the ACM*, 51(9):72–79, 2008.

[Boag *et al.*, 2002] Scott Boag, Don Chamberlin, Mary F Fernández, Daniela Florescu, Jonathan Robie, Jérôme Siméon, & Mugur Stefanescu. Xquery 1.0: An xml query language. 2002.

[Bray *et al.*, 1997] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, & François Yergeau. Extensible markup language (xml). *World Wide Web Journal*, 2(4):27–66, 1997.

[Bray, 2014] Tim Bray. The javascript object notation (json) data interchange format. 2014.

[Chappell, 2004] David Chappell. *Enterprise service bus*. "O'Reilly Media, Inc.", 2004.

[Davis, 1985] Fred D Davis. *A technology acceptance model for empirically testing new end-user information systems: Theory and results*. PhD thesis, Massachusetts Institute of Technology, 1985.

[Davis, 1989] Fred D Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319–340, 1989.

[Dossot *et al.*, 2014] David Dossot, John d'Emic, & Victor Romero. *Mule in action*. Manning Publications Co., 2014.

[Frolick & Ariyachandra, 2006] Mark N Frolick & Thilini R Ariyachandra. Business performance management: One truth. *IS Management*, 23(1):41–48, 2006.

[Gamma *et al.*, 1995] Erich Gamma, Richard Helm, Ralph Johnson, & John Vlissides. Design patterns: Elements of reusable software components. 1995.

[Gormley & Tong, 2015] Clinton Gormley & Zachary Tong. *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine*. "O'Reilly Media, Inc.", 2015.

[Gupta, 2003] Samudra Gupta. *Logging in Java with the JDK 1.4 Logging API and Apache log4j*. Springer, 2003.

[Halevy *et al.*, 2005] Alon Y Halevy, Naveen Ashish, Dina Bitton, Michael Carey, Denise Draper, Jeff Pollock, Arnon Rosenthal, & Vishal Sikka. Enterprise information integration: successes, challenges and controversies. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 778–787. ACM, 2005.

[Hasselbring, 2000] Wilhelm Hasselbring. Information system integration. *Communications of the ACM*, 43(6):32–38, 2000.

[Kleindienst, 2016] Patrick Kleindienst. Building a real-world logging infrastructure with logstash, elasticsearch and kibana. 2016.

[Loshin, 2010] David Loshin. *Master data management*. Morgan Kaufmann, 2010.

[McCandless *et al.*, 2010] Michael McCandless, Erik Hatcher, & Otis Gospodnetic. *Lucene in action: covers Apache Lucene 3.0*. Manning Publications Co., 2010.

[Rademakers & Dirksen, 2008] Tijs Rademakers & Jos Dirksen. *Open-source ESBs in action*. Manning Publications Co., 2008.

[Rodriguez, 2008] Alex Rodriguez. Restful web services: The basics. *IBM developerWorks*, 2008.

[Schulte, 2002] R Schulte. Predicts 2003: Enterprise service buses emerge. *Report, Gartner, December*, 2002.

[Shafranovich, 2005] Yakov Shafranovich. Common format and mime type for comma-separated values (csv) files. 2005.

[Shahbaz, 2015] Qamar Shahbaz. *Data mapping for data warehouse design.* Elsevier, 2015.

[W3Schools, 2018] W3Schools. Xml tutorial. `https://www.w3schools.com/xml/`, 2018. Tarkistettu 27.5.2018.

[Wiederhold, 1999] Gio Wiederhold. Mediation to deal with heterogeneous data sources. In *Interoperating Geographic Information Systems.* Springer, 1999.

Liite

Perceived usefulness

1. Using the system in my job would enable me to accomplish tasks more quickly	1	2	3	4	5	6	7
2. Using the system would improve my job performance	1	2	3	4	5	6	7
3. Using the system in my job would increase my productivity	1	2	3	4	5	6	7
4. Using the system would enhance my effectiveness on the job	1	2	3	4	5	6	7
5. Using the system would make it easier to do my job	1	2	3	4	5	6	7
6. I would find the system useful in my job	1	2	3	4	5	6	7

Perceived ease of use

7. Learning to operate the system would be easy for me	1	2	3	4	5	6	7
8. I would find it easy to get the system to do what I want it to do	1	2	3	4	5	6	7
9. My interaction with the system would be clear and understandable	1	2	3	4	5	6	7
10. I would find the system to be flexible to interact with	1	2	3	4	5	6	7
11. It would be easy for me to become skillful at using the system	1	2	3	4	5	6	7
12. I would find the system easy to use	1	2	3	4	5	6	7