Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Tietotekniikan koulutusohjelma

TILANNEKUVAJÄRJESTELMÄN TIETOKANNAN TOTEUTTAMINEN PILVIYMPÄRISTÖSSÄ

Joonatan Rantanen

Työn tarkastaja: Apulaisprofessori Antti Knutas

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT

School of Engineering Science

Tietotekniikan koulutusohjelma

Joonatan Rantanen

Tilannekuvajärjestelmän tietokannan toteuttaminen pilviympäristössä

Kandidaatintyö 2020

33 sivua, 4 kuvaa

Työn tarkastaja: Apulaisprofessori Antti Knutas

Hakusanat: tilannekuvajärjestelmä, tietokanta, pilvilaskenta

Keywords: situational awareness system, database, cloud computing

Pilvilaskentaa hyödynnetään nykyaikaisessa tietotekniikassa runsaasti sen tuomien etujen takia. Tilannekuvajärjestelmät ovat yksi järjestelmätyyppi, joka voi hyötyä pilvilaskennan käytöstä. Tilannekuvajärjestelmät ovat järjestelmiä, jotka luovat tilannekuvaa ympäristöstä esimerkiksi erilaisten mittalaitteiden avulla. Tietokanta on oleellinen osa tällaista järjestelmää. Tämän työn tavoitteena oli selvittää, miten tilannekuvajärjestelmän tietokanta tulisi toteuttaa pilvialustalla. Tavoitetta lähestyttiin tutustumalla aiheeseen liittyviin tieteellisiin julkaisuihin ja case-järjestelmään tehtävän kehityksen avulla. Työn case-järjestelmä on Observis Oy:n ObSAS-ohjelmistotuote. Järjestelmän kehitystyön lisäksi suunniteltiin yleinen malli toteutusten tekemiseen. Työ rajattiin käsittelemään relationaalisia tietokantoja, koska case-järjestelmässä oli käytössä vain relaatiotietokanta. Tarkasta työn aiheesta tilannekuvajärjestelmiin liittyen ei löytynyt tieteellisiä julkaisuja, mutta läheisistä aiheista sensorijärjestelmiin ja pilvilaskentaan liittyen löytyi paljonkin kirjallisuutta ja tutkimusta, jota voitiin hyödyntää työssä. Case-järjestelmää kehitettiin suunnittelemalla tietokantaan liittyvä rakenne pilvitoteutusta varten ja tekemällä vaadittavat muutokset järjestelmään proof of concept -version luomiseksi. Yleisille toteutuksille luotu malli esitettiin kahden päätöksentekopuun avulla, joissa käsitellyt kysymykset ja vastaukset johdettiin case-järjestelmän kehityksen tärkeimmistä kysymyksistä sekä lähimmin aiheeseen liittyvistä järjestelmätoteutuksista ja kirjallisuudesta. Mallin aiheet liittyvät erityisesti käytettävään pilvipalvelumalliin sekä valintaan yhden tai useamman tietokannan välillä.

ABSTRACT

Lappeenranta-Lahti University of Technology LUT

School of Engineering Science

Degree Programme in Software Engineering

Joonatan Rantanen

Implementing the database of a situational awareness system in cloud environment

Bachelor's Thesis 2020

33 pages, 4 figures

Examiner: Assistant Professor Antti Knutas

Keywords: situational awareness system, database, cloud computing

Cloud computing is utilized abundantly in modern IT. Situational awareness systems are a type of system that can benefit from using cloud computing. Situational awareness systems are systems that create situational awareness from the environment with different kinds of measuring devices, for example. A database is an essential part of this kind of system. The aim of this work was to figure out how the database of a situational awareness system should be implemented on a cloud platform. The objective was approached by studying scientific publications related to the subject and by doing development for a case system. The case system of this work is ObSAS, a software product of Observis Oy. In addition to the development work for the system a general model for implementations was designed. This work was limited to relational databases because the case system only had a relational database. Scientific publications about the exact subject of this work relating to situational awareness systems were not found, but a lot of literature and research was found about similar subjects relating to sensor systems and cloud computing, which could be utilized in this work. The case system was developed by designing the structure relating to the database for cloud implementation, and by making the required modifications to the system for creating a proof-of-concept version. The model created for general implementations was presented with two decision trees, the questions and answers in which were derived from the most important questions of the case system development and the system implementations and literature that most closely related to the subject. The subjects in the model especially relate to the cloud service model being used and the choice between one or more databases.

ALKUSANAT

Työ on tehty Mikkelissä Observis Oy:llä kesällä 2020 ja viimeistelty syksyllä Lappeenrannassa. Tahdon kiittää kaikkia Observis Oy:llä mahdollisuudesta työn tekemiseen ja avustuksesta toteutuksessa. Tahdon myös kiittää kaikkia muita, jotka ovat auttaneet työn tekemisessä tai olleet tukena siinä.

SISÄLLYSLUETTELO

SYMBOLI- JA LYHENNELUETTELO

CBRN	Chemical, Biological, Radiological, Nuclear
DBaaS	Database as a Service
DSR	Design Science Research
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a service
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
NATO	North Atlantic Treaty Organization
NIST	National Institute of Standard and Technology
NoSQL	Not only SQL/No SQL
PaaS	Platform as a Service
REST	Representational State Transfer
SaaS	Software as a Service
SQL	Structured Query Language
VPN	Virtual Private Network
WSN	Wireless Sensor Network

1 JOHDANTO

1.1 Tausta

Pilvilaskennan kasvattaessa suosiotaan viime vuosikymmenen ajan niin teolliset kuin akateemiset tahot ovat kiinnittäneet siihen runsaasti huomiota (Al-Dhuraibi et al, 2017). Pilvilaskennan NIST:n (National Institute of Standard and Technology) määritelmästä löytyy useita olennaisia ominaisuuksia, joiden houkuttelevuus nykyaikaisessa tietotekniikan käytössä on helppo nähdä: tietojenkäsittelykyvykkyyksien varaaminen itsepalveluna tarpeen vaatiessa, laaja saatavuus verkon avulla, käytettävissä olevien resurssien säätely joustavasti ja nopeasti sekä kuluttajan näkökulmasta lähes rajatta (Mell & Grance, 2011).

Varsinaisesti pilvilaskennan voidaan määritellä olevan “malli, joka mahdollistaa kaikkialla helposti ja tarvittaessa saatavilla olevan pääsyn verkon kautta jaettuihin konfiguroitaviin tietojenkäsittelyresursseihin, joita voidaan toimittaa ja jakaa minimaalisella hallinnallisella työllä tai vuorovaikutuksella tarjoajan kanssa”. Resursseilla voidaan tarkoittaa esimerkiksi palvelimia, tallennustilaa tai palveluita. Lisäksi NIST määrittelee pilvipalveluille useita mahdollisia palvelumalleja, kuten SaaS (Software as a Service), PaaS (Platform as a Service) ja Iaas (Infrastructure as a service), sekä käyttöönottomalleja, kuten yksityinen, julkinen ja hybridipilvi. (Mell & Grance, 2011)

Pilvilaskennalle ominaista ovat käytön mukainen maksaminen resursseista sekä varsinaisen fyysisen laitteiston peittäminen hyödyntämällä virtualisoituja alustoja (Fehling et al, 2011). Joustavuus on tärkeä käsite pilvilaskennassa, ja pilvilaskennan suurimpia etuja perinteisiin menetelmiin nähden onkin mahdollisuus muuttaa resurssien määrää dynaamisesti tarpeiden mukaan (Al-Dhuraibi et al, 2017). Viime vuosikymmenen aikana sekä suuret että pienet ja keskisuuret yritykset ovat yhä suuremmissa määrin siirtyneet pilvilaskennan käyttöön. Joustavuuden ja skaalautuvuuden lisäksi tärkeimpiä syitä on sen tarjoama yksinkertaisuus, ja ostamalla tietojenkäsittelyresursseja pilvipalvelujen tarjoajilta yritykset voivat keskittyä ydinaktiviteetteihinsa. (El-Gazzar, 2014) Yritykset saavat siis toimintaansa paljon vapautta, kun tallennustilaa ja laskentatehoa voidaan ostaa vain kulloisenkin tarpeen mukaan, ja määrää voidaan kasvattaa helposti paljonkin. Resurssien jako verkon kautta ei myöskään sido mihinkään tiettyyn fyysiseen sijaintiin.

Pilvilaskennan lisäksi tässä työssä olennaisessa osassa ovat tietokannat, ja näistä erityisesti relaatiotietokannat. Tietokannalla tarkoitetaan sellaista tiedon varastoimisvälinettä, josta on mahdollista hakea tietoa. Yksinkertaisimmillaan relaatiotietokantoja voi kuvailla tietokannoiksi, jotka esittävät tiedon rivejä ja sarakkeita sisältävissä taulukoissa. Relaatiotietokantojen kanssa käytettäväksi on suunniteltu SQL-kieli (Structured Query Language), jonka peruskomentoja voi käyttää kaikissa relaatiotietokantojen hallintajärjestelmissä. (Oracle, 2020) Uudempana vaihtoehtona relaatiotietokannoille ovat nousseet niin sanotut NoSQL-tietokannat, joiden nimi tulkitaan usein “not only SQL” eli “ei ainoastaan SQL”, eikä “no SQL” eli “ei SQL”. NoSQL-tyyppejä on useita, ja niiden malli on kehitetty hyvin suuriin datamääriin skaalautuvaksi, jossa relaatiomallia toteuttavilla tietokannoilla on ollut ongelmia. (Nayak et al, 2013)

Kolmantena työn tärkeimmistä käsitteistä ovat tilannekuvajärjestelmät. Stanton et al (2001) esittelevät kolme eri lähteistä peräisin olevaa tärkeää määritelmää tilannekuvalle (engl. situational awareness tai situation awareness). “Ympäristön elementtien havaitseminen tietyssä aika- ja tilamäärässä, niiden tarkoituksen ymmärrys ja projektio lähitulevaisuuteen.” “Tietoinen yksilön dynaaminen reflektio tilanteesta, joka tarjoaa dynaamisen orientaation siitä ja mahdollisuuden reflektoida menneisyyttä, nykyhetkeä ja tulevaisuutta, sekä tilanteen potentiaalisia ominaisuuksia. Se sisältää loogis-käsitteellisiä, mielikuvituksellisia, tietoisia ja tiedostamattomia osia, jotka mahdollistavat yksilön henkisen mallin kehittämisen ulkoisista tapahtumista.” “Muuttumaton osa toimija-ympäristö-systeemissä, joka luo hetkellisen tiedon ja käyttäytymisen suorituksen välittäjän määrittelemien tavoitteiden saavuttamiseksi ympäristössä.” Määritelmät painottavat eri asioita, mutta yhdessä ne voivat auttaa käsitteen kokonaisuuden hahmottamisessa. Työssä käsiteltävä tilannekuvajärjestelmä voidaan ymmärtää nimensä mukaisesti järjestelmäksi, joka auttaa tilannekuvan luomisessa.

Observis Oy on vaativiin tilannekuvajärjestelmiin ja erityisesti CBRN-sensoreista (engl. Chemical, Biological, Radiological, Nuclear) koostuvien järjestelmien hallintaohjelmistoon erikoistunut yritys, jonka päätuotteena olevan ObSAS-ohjelmiston kehitykseen tämä opinnäytetyö on tehty. Observis Oy:n kehittämä ObSAS on juuri erilaisista mittalaitteista koostuvan systeemin hallintaan käytettävä tilannekuvajärjestelmä. ObSAS:ia ollaan laajentamassa tekemällä sille toteutus pilvialustalle, ja tämän kandidaatintyön tavoite on

viedä kehitystä eteenpäin suunnittelemalla ja toteuttamalla osa pilviversiosta. Aiemmat ObSAS:n asennukset ovat olleet paikallisia, mutta työn aloittamisen aikaan Google Cloud -pilveen on jo tehty yksinkertainen alustava versio. Työ keskittyy ohjelmiston tietokantakomponentin toteutukseen.

1.2 Tavoitteet ja rajaukset

Tässä työssä tutkittava ongelma on tilannekuvajärjestelmän tietokannan siirtäminen pilveen, ja aihetta lähestytään käsiteltävänä olevan case-järjestelmän kautta. Työn päätavoite on selvittää eri vaihtoehdot ja mahdollisuudet tilannekuvajärjestelmän tietokannan toteuttamiseksi pilviympäristössä, tehdä tutkimusta parhaista ratkaisuista, sekä suunnitella ja tehdä käytännön malli ja toteutus selvityksen avulla saavutettua tietoa pohjana käyttäen. Selvitys tehdään tutkimalla aihetta käsitteleviä tieteellisiä julkaisuja. Tarkoituksena on löytää vastaus tutkimuskysymykseen

“Miten tilannekuvajärjestelmän tietokanta tulisi toteuttaa pilvialustalla?”

Case-järjestelmän sisäisen rakenteen yksityiskohdat ovat Observis Oy:n omistamaa tietoa, joten kaikkia siihen tehtyjä muutoksia ei voida kuvata tarkasti. Yleisenä käytännön tuotoksena työssä luodaan myös case-järjestelmän toteutuksen ja tieteellisen kirjallisuuden pohjalta muodostettu malli pilvialustalla toimivan tilannekuvajärjestelmän tietokannan toteuttamiseen.

Tärkeänä rajauksena työn toteutusosuudessa tullaan käsittelemään vain relaatiotietokantoja, koska muutokset tietokantamalliin aiheuttaisivat case-järjestelmään liian suuria muutoksia työn laajuuteen nähden. Lisäksi relaatiotietokannat todennäköisesti soveltuvat NoSQL-malleja paremmin case-järjestelmään esimerkiksi sen raportointivaatimusten takia. Järjestelmällä on useita asiakkaita, ja suunnittelussa on olennaista ottaa huomioon rakenne eri käyttäjien tai ryhmien tiedoille ja niiden käytölle. Tietokantaa tullaan myös käsittelemään osana muuta pilvialustalle rakennetun järjestelmän kokonaisuutta, eikä kyse ole siis paikallisen systeemin erillisestä pilvitietokannasta. Työ on rajattu järjestelmän tietokantaosuuteen, koska kokonaisuuden käsitteleminen olisi liian suuri alue työn

laajuudelle. Tietokanta on erittäin olennainen osa tilannekuvajärjestelmää, ja muodostaa itsessään selkeästi rajatun ja sopivan kokonaisuuden.

Työn case-järjestelmään tehdyssä toteutuksessa otetaan huomioon järjestelmän nykytilanne sekä verrattavissa olevissa järjestelmissä käytössä olevat teknologiat. Vaihtoehtoja harkitaan niin käsiteltävän ohjelmiston kuin pilvilaskennan nykytilanteessa yleisesti käytettyjen teknologioiden kontekstissa, ottaen huomioon myös ohjelmiston erityispiirteet ja nykyisten arkkitehtuuriratkaisujen tilanne. Varsinkin jos arkkitehtuurille lähdetään harkitsemaan suurempien muutoksien tekemistä pilviversiota varten, huomioitavia kriteerejä ovat, kuinka paljon lisää työtä mitkäkin ratkaisut tuottavat ja kuinka paljon ne toisaalta tuovat hyötyä. Muutokset voivat kuitenkin myös avata tapoja hyödyntää pilven tarjoamia mahdollisuuksia suuremmalla tasolla. Lisäksi suunnittelussa huomioitava piirre voi olla järjestelmän eteenpäin kehitettävyys alueen nopean muuttumisen ja kehittymisen takia, jolloin tulevaisuuden muutosten helpottaminen on tärkeää.

Pilvilaskenta ja varsinkin asiakkaiden mahdollisesti arkaluonteisten tietojen tallennus pilveen herättää usein kysymyksiä tietoturvasta. Case-järjestelmässä todennäköisesti suurin osa asiakkaista tulee käyttämään perinteistä asennusta juuri tietoturvan aiheuttamien huolien takia. Asiakkaiden ollessa esimerkiksi eri maiden viranomaisia on ymmärrettävää, että tärkeiden tietojen tallentaminen ja käsitteleminen toisessa maassa sijaitsevalla laitteistolla ei ole houkutteleva ratkaisu. Vaihtoehtoisia ratkaisuja ongelmaan voisivat olla yksityisen pilven hyödyntäminen tai sellaiset asiakkaat, joille tietojen täydellinen turvaus ei välttämättä ole niin tärkeää, kuten mahdollisesti pelastuslaitokset.

1.3 Työn rakenne

Työn rakenne tulee jatkumaan siten, että seuraavassa luvussa käsitellään aiheesta tehtyä kirjallisuutta, ja sitten kuvaillaan käytettävää tutkimusmenetelmää. Tämän jälkeen selostetaan työn käytännön osuuden toteutus, keskustellaan tuloksista ja päätetään työ yhteenvetoon.

2 KIRJALLISUUSKATSAUS

2.1 Aiempi tutkimus

Pilvilaskennan käytöstä ja sen monista eri sovelluksista on tehty runsaasti tutkimusta. Niin markkinoilla olevista pilvipalvelujen tarjoajista, toimintatavoista, olemassa olevien järjestelmien siirtämisessä pilveen, eri tavoista hyödyntää pilvilaskentaa tietokannoissa kuin sensori- ja IoT-datan (engl. Internet of Things eli esineiden Internet) yhdistämisestä pilvisovelluksiin löytyy useita tieteellisiä julkaisuja ja artikkeleita.

Vaikka erilaisista sensoriverkoista ja tilannekuvasysteemeistä löytyy tieteellisiä julkaisuja, tehdyn tiedonhaun perusteella nämä ovat yleensä keskittyneet ja rajoittuneet spesifeihin käyttötarkoituksiin, kuten esimerkiksi sovelluksiin älykkäissä sähköverkoissa (engl. smart grid) (Saunders et al, 2016; Su et al, 2016). Käsiteltävän case-järjestelmän tyyppisistä erilaisten CBRN-sensorien hallintaan käytettävistä järjestelmistä löytyy hyvin niukasti aiempaa tutkimusta pilvilaskennan hyödyntämisen näkökulmasta, mutta työssä voidaan hyödyntää tutkimusta pilvilaskennan käytöstä erilaisissa dataa keräävissä, käsittelevissä ja tallentavissa sovelluksissa. On mahdollista, että työtä varten suoritettua tiedonhakua ei osattu kohdistaa sopivasti haluttujen julkaisujen löytämiseksi, mutta tämä oletetaan epätodennäköiseksi. Haut suoritettiin Google Scholarissa sekä tieteellisten julkaisujen tietokannoissa, kuten IEEE Xploressa, käyttäen useita erilaisia hakusanoja ja niiden yhdistelmiä.

WSN- (engl. Wireless Sensor Network, langaton sensoriverkko) ja IoT-sovelluksista pilvessä ovat kirjoittaneet esimerkiksi Hromic et al (2015), Fortino et al (2014), Ji et al (2014) ja Hou et al (2016). Kaikissa edellä mainituissa tapauksissa käytettiin NoSQL-tietokantoja tallentamiseen hyvin suurten datamäärien takia. Tämä näyttäisi olevan yleistä sensorisovelluksissa, kuitenkin monessa tapauksessa hyödynnetään myös perinteisiä relationaalisia järjestelmiä ainakin jossain osassa, kuten esimerkiksi tapauksissa, jotka Grothe et al (2016), Phan et al (2015) ja Barros et al (2018) esittelivät. Joissakin järjestelmissä näyttäisi olevan käytössä pelkkä relationaalinen tietokanta, jollaisia Majumdar et al (2012), Chung et al (2013), Alamdar et al (2016) ja Dias et al (2018) käsittelivät. Lisäksi

Rajkumar ja Iyengar (2013) kirjoittivat maaseudun terveydenhuollon hätätilanteiden hoidossa hyödynnettävästä pilvisovelluksesta, jossa käytettiin relationaalista tietokantaa. Monessa sensorisovellusten pilvikäyttöä käsittelevässä julkaisussa ei ole tarkempaa mainintaa tietokantojen tyypeistä (Hassan et al, 2009; Hummen et al, 2012; Abawajy & Hassan, 2017; Suciu et al, 2013; Dash et al, 2012). Aiheeseen liittyvää tutkimusta on myös eri IoT-pilvialustojen tarjoajista (Ray, 2016).

Van der Veen et al (2012) vertailivat SQL- ja NoSQL-tietokantoja sensoridatan kanssa pilvilaskennan näkökulmasta sekä virtuaalikoneessa että perinteisemmällä fyysisellä alustalla. Tulokset eivät nostaneet mitään kolmesta testatusta tietokannasta selväksi voittajaksi, vaan toivat esiin niiden vahvuuksia tietyissä tilanteissa. Esimerkiksi ainoa testattu relaatiotietokanta, PostgreSQL, oli selvästi hitain pienissä kirjoitusoperaatioissa, mutta paras useita lukuoperaatioita tehdessä, ja virtualisoinnilla oli muita vähemmän negatiivista vaikutusta siihen. Hammes et al (2014) vertailivat NoSQL- ja SQL-tietokantoja yleisesti pilvikäytössä, eivätkä tulleet ratkaisevaan johtopäätökseen kummankaan testattuna olleen tietokannan kattavasta paremmuudesta. Tutkimusten perusteella pilvitietokannan tyyppi tulisi siis valita käyttötarkoituksen perusteella.

Olemassa olevien applikaatioiden tai tietokantojen siirtämisestä pilveen löytyy suuri määrä julkaisuja. Esimerkiksi Strauch et al (2013b ja 2014), Yu et al (2012), Maenhaut et al (2013), Vodomin & Andročec (2015) ja Balalaie et al (2015) ovat kirjoittaneet aiheesta relaatiotietokantojen osalta, ja Adewojo et al (2015) sekä Chauhan & Babar (2011) NoSQL:n kannalta. Useissa julkaisuissa myös käsitellään molempia tietokantatyyppejä (Zou et al, 2013; Strauch et al, 2013a; Tran et al, 2011; Andrikopoulos et al, 2012; Ellison et al, 2018; Zhao et al, 2012).

Koska pilvikäytössä tietojen käsittely tapahtuu ulkopuolisen tarjoajan arkkitehtuurilla, herättää datan tietoturva usein huolia. Tietoturvakysymyksiä tiedon salauksen näkökulmasta käsittelivät esimerkiksi Wang et al (2017) ja Ferretti et al (2014). Toinen lähestymistapa ongelmaan on yksityis- tai hybridipilviratkaisujen käyttö, jolloin mahdollisesti arkaluonteiset tiedot eivät päädy lainkaan muiden tiloihin. Suciu et al (2012) vertailivat julkista ja yksityistä pilviratkaisua, ja tietoa löytyy myös hybridipilven käytöstä (Helmi et al, 2018). Kaikissa näissä julkaisuissa keskityttiin relaatiotietokantoihin.

Pilvisovelluksissa tärkeässä asemassa ovat usean asiakkaan käytössä olevat järjestelmät (engl. multi-tenancy), ja myös case-järjestelmällä on tarkoitus olla useita pilvisovellusta erillisesti käyttäviä asiakkaita. Aiheesta löytyviä esimerkkejä ovat tiedonhallinnan suuren skaalautuvuuden saavuttamiseksi luotu viitekehys (Maenhaut et al, 2015a), vanhasta koodista Dockeria hyödyntäen rakennettu usean asiakkaan pilvipalvelu (Slominski et al, 2015), sekä käyttäjien tietoa joustavasti allokoiva järjestelmä (Maenhaut et al, 2015b).

Tutkimusta löytyy myös relaatiomallin käytöstä ja sen ongelmista pilvessä (Feuerlicht & Pokorný, 2013; Kiefer & Lehner, 2011), pilvisovellusten siirrettävyyden ongelmista (Gonidis et al, 2013), sekä yleisesti pilvitietokannoista ja niiden arkkitehtuurista (Sakr, 2013; Bogdanov & Lwin, 2015; Kulkarni et al, 2012). Bogdanov ja Lwin (2015) keskittyivät relaatiotietokantoihin, mutta Sakr (2013) sekä Kulkarni et al (2012) käsittelivät lisäksi NoSQL-tietokantoja.

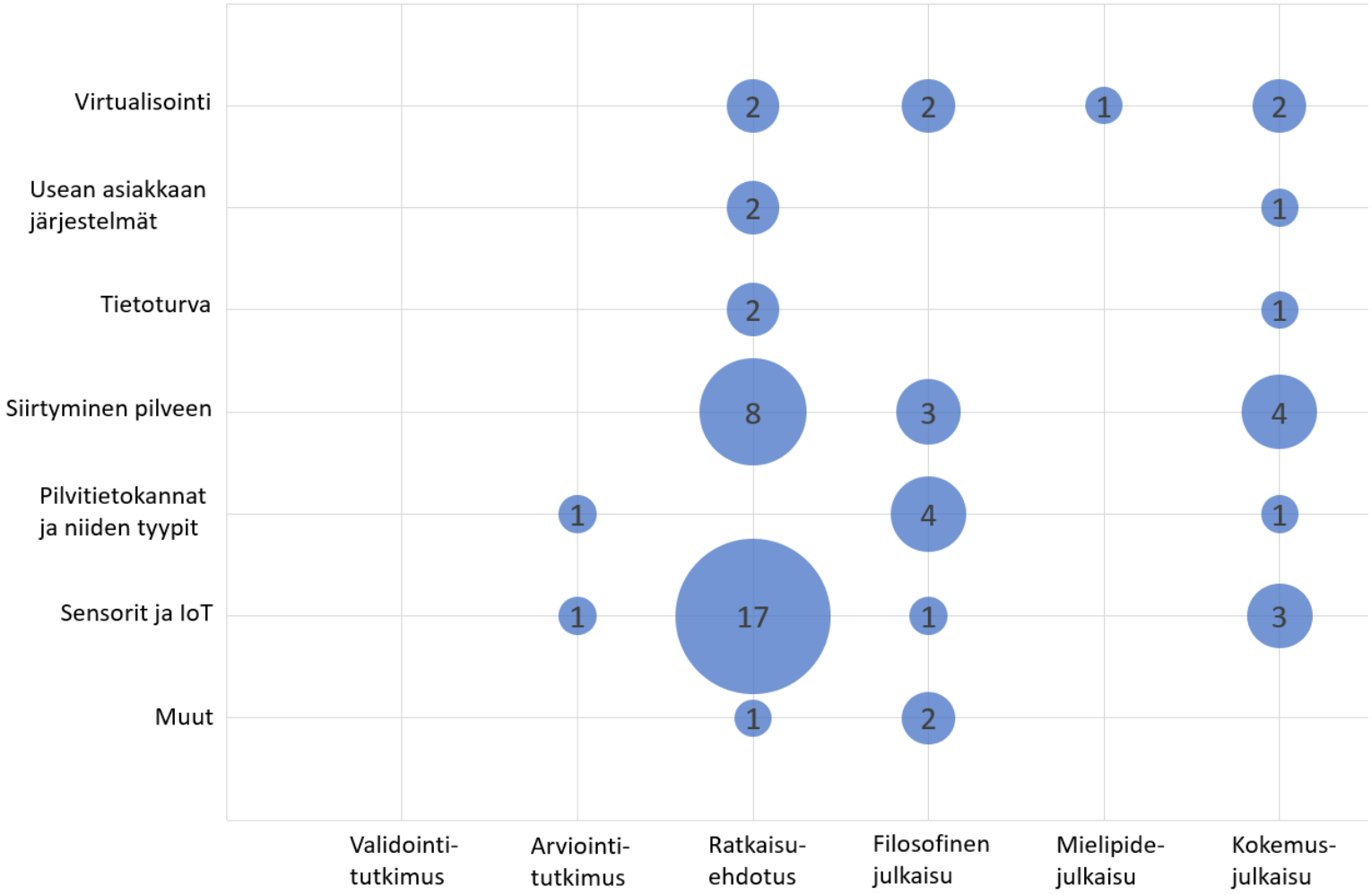
Hyvin oleellinen käsite pilvilaskennasta puhuttaessa on virtualisointi, ja suurin osa pilvipalveluiden tarjoajista hyödyntääkin sitä tarjoamissaan resursseissa. Voidaankin sanoa, että pilvilaskenta on riippuvainen virtualisointiteknologiasta. Virtualisoinnilla mahdollistetaan useiden käyttöjärjestelmien ajo samalla palvelimella, resurssien nopea toimitus ja skaalaus sekä abstraktoidaan laitteiston yksityiskohdat. Perinteisesti virtualisointi on toteutettu virtuaalikoneiden avulla (engl. virtual machine, VM). (Al-Dhuraibi et al, 2017) Virtuaalikoneet ovat muusta järjestelmästä eristettyjä “tietokoneita tietokoneen sisällä”, joita voidaan kuvailla tietokoneen lailla käyttäytyvinä tiedostoina (Microsoft, 2020). Virtuaalikoneiden raskauden takia uudempana keksintönä kehitetty säiliöinti (engl. containerization), joka on huomattavasti kevyempi ja nopeampi virtualisoinnin muoto, on noussut vaihtoehdoksi. Toisin kuin virtuaalikoneet, säiliöt eivät tarvitse omaa käyttöjärjestelmäänsä, mutta haittapuolena niillä on vaikeampi asettaa yhtä korkea eristyneisyys muusta järjestelmästä, ja ne ovat yhä kehittyvä teknologia. (Al-Dhuraibi et al, 2017)

Case-järjestelmässä hyödynnetään jo valmiiksi säiliöintiteknologiaa, mikä tekee siitä luonnollisen myös pilvikäytössä. Tässä työssä keskitytään Docker-säiliöintiin sen tuoreuden, monipuolisten ominaisuuksien sekä case-järjestelmässä käytön takia. Erityisesti Dockerista

pilvilaskennan kontekstissa, myös tietokantoihin liittyen, ovat kirjoittaneet esimerkiksi Peinl et al (2016), Shah & Dubaria (2019) ja Nadgowda et al (2017). Dockerin ja pilven käyttöä sensorisovelluksissa on myös käsitelty useassa lähteessä (Debauche et al, 2018; Sun et al, 2017; Chaturvedi & Kolbe, 2018). Tutkimusta löytyy lisäksi Docker-järjestelmien hallinnasta (Modak et al, 2018), tietoturvasta (Combe et al, 2016) sekä Dockerin ja mikropalveluiden käyttöön siirtymisestä (Balalaie et al, 2015).

Alueella on tapahtumassa edelleen nopeaa kehitystä eteenpäin. Meneillään on jatkuvia muutoksia, ja uusia trendejä pilvilaskennassa ovat esimerkiksi Multi-cloud, Fog ja Edge Computing, Hybrid Cloud ja Function-as-a-service (Varghese & Buyya, 2018). Pilvilaskentaa sovellettaessa voikin olla tärkeää huomioida tämä hyvin eteenpäin kehitettävissä olevilla ratkaisuilla.

Kuten runsaasta tieteellisten julkaisujen määrästä voi huomata, pilvilaskentaa ja sen ominaisuuksia, erityisesti tietokantoihin liittyen, on tutkittu monipuolisesti useista näkökulmista. Suosittuja aiheita ovat esimerkiksi sensori- ja IoT-käyttö, tietokantatyypit, sovellusten ja tietokantojen siirtäminen pilveen, tietoturva, usean asiakkaan järjestelmät, arkkitehtuuri ja virtualisointi. Kuluneen vuosikymmenen aikana tutkittavana on ollut paljon erilaisia pilvilaskentaan liittyviä teknologioita, ja kehitystä sekä uusia teknologisia ratkaisuja on yhä tapahtumassa. Kuvassa 2.1.1 on esitetty kaikki tässä luvussa käsitellyt artikkelit aihealueen ja julkaisun tyypin mukaan jaoteltuna. Julkaisutyyppien jaottelu ja kuvaajan valinta on tehty Petersen et al (2008) esimerkin mukaisesti. Artikkeleiden aiheiden jakautumiseen on vaikuttanut tiedonhaun painotukset.



Kuva 2.1.1. Käsitellyt artikkelit aiheen ja julkaisun tyypin mukaan.

2.2 Pilvilaskennan palvelumallit

Pilvipalveluita voidaan ostaa ja tarjota eri virtualisoinnin tasoilla, jotka on yleisesti jaettu IaaS-, PaaS- ja SaaS-palvelumalleiksi. Myös tarkempaa jaottelua esiintyy, mutta nämä kolme ovat yleisessä käytössä ja löytyvät esimerkiksi NIST:n pilvilaskennan määritelmistä. SaaS- eli Software as a Service -mallissa asiakkaalle tarjotaan pilven infrastruktuurilla ajettavia applikaatioita, joihin pääsee käsiksi esimerkiksi selaimella tai ohjelmistorajapinnalla, ja enimmillään asiakas voi hallita käyttäjäkohtaisia ohjelman konfiguraatioita. PaaS- eli Platform as a Service -mallissa asiakas voi ottaa pilven infrastruktuurilla käyttöön kehittämiään tai hankkimiaan applikaatioita käyttäen tarjoajan tukemia ohjelmointikieliä, kirjastoja, palveluita ja työkaluja. Tässä mallissa asiakas hallitsee

käyttöön otettavia ohjelmistoja sekä mahdollisesti ympäristön konfiguraatioita, mutta ei infrastruktuuria kuten käyttöjärjestelmää, muistia tai palvelimia. IaaS- eli Infrastructure as a Service -mallissa taas asiakkaalle toimitetaan perustavanlaatuisia tietojenkäsittelyresursseja, kuten muistia, verkkoja tai prosessointitehoa, joita käyttäen voi ajaa mielivaltaisia ohjelmistoja tai käyttöjärjestelmiä. Siinä asiakas hallitsee esimerkiksi muistia, käyttöjärjestelmiä ja mahdollisesti rajattuja verkon komponentteja, kuten palomuureja, mutta ei taustalla olevaa pilven infrastruktuuria. (Mell & Grance, 2011)

Tämän työn ja sen kanssa tapahtuvan case-järjestelmän kehityksen lopputuotteena asiakkaille tullaan tarjoamaan pilvialustalla ajettavaa ohjelmistoa eli SaaS-palvelua. Tietokanta voitaisiin teoriassa itsekin ostaa SaaS-palveluna (tunnetaan tässä tapauksessa myös DBaaS eli Database as a Service (IBM Cloud Education, 2019)), mutta tämä ei olisi järkevää, koska kaikki järjestelmän muutkin osat tulevat olemaan pilvessä. Lisäksi hallinta olisi paljon rajoitetumpaa ja tietoturvaan liittyisi lisää huolia. Käytännössä harkittavana ostettavaksi ovat siis IaaS- tai PaaS-mallin palveluita, joita tullaan jalostamaan SaaS-palveluksi.

Eri pilvipalvelutyyppien hintaa on hyvin vaikea verrata, koska eri palveluntarjoajat käyttävät erilaisia hinnoittelutapoja ja malleja, eri tuotteissa tulee erilaisia palveluita ja komponentteja mukana, esimerkiksi muistia tai prosessointitehoa voi olla eri määriä ja hinnoissa myös tapahtuu jatkuvia muutoksia. Kuten Wu et al (2019) toteavat, eri hinnoittelutapojen määrä on valtava, ja artikkelissa listataan suuri määrä käytettyjä strategioita ja tyyppejä hinnoittelulle. Pääpiirteisesti hinnat voivat mennä esimerkiksi usein pilvilaskennan yhteydessä korostetulla käytön mukaisella maksulla tai vaikkapa tilauksella. Lisäksi jos esimerkiksi ostaa IaaS-palvelua, on hyvin vaikea selvittää, mikä hinta on lisätyöllä, jonka joutuu tekemään verrattuna PaaS-palveluun, ja toisaalta minkä arvoista saatava vapaus kustomoinnissa on. Perinteisten palvelumallien lisäksi nykyään kehitys on myös hinnoittelussa kulkemassa Function as a Service -tyyppisen mallin suuntaan (Wu et al, 2019).

IaaS-mallissa pilvipalvelun ostajalla on itse vastuu infrastruktuurin hallinnoimisesta (Gibson et al, 2012), mutta toisaalta sen mukana tulee muita malleja suuremmat vapaudet tehdä asiat niin kuin haluaa. Hinta on usein tärkeä tekijä palvelua harkitessa. On kuitenkin tutkimusta,

joka viittaisi, että raskaalla laskenta- ja verkkokäytöllä oma infrastruktuuri saattaa olla halvempaa kuin IaaS-pilvipalvelun ostaminen (Gibson et al, 2012). IaaS-mallissa tietoturva on tärkeässä roolissa. Palvelun asiakkaat toimivat jaetussa ympäristössä käyttäen omia pilvialustalla ajettavia virtuaalikoneita, ja niitä hallitsevaan hypervisoriin murtautuminen voi vaarantaa muistin ja tietoliikenteen, vaikka ylläpitäjällä ei olisikaan niihin pääsyä. (Gibson et al, 2012) Myös virtuaalikone, johon on murtauduttu voi hyökätä kaikkiin muihin jaetun ympäristön takia (Freet et al, 2015). Hyvänä puolena tässä mallissa voi muita paremmin kerätä yksityiskohtaisia lokitietoja ja todistusaineistoa (Gibson et al, 2012; Freet et al, 2015). Muita mahdollisesti hyödyllisiä työkaluja tietoturvan parantamiseen ovat pienimmän etuoikeuden periaatteen käyttäminen, osassa hypervisoreissa olevan etuoikeutetun rakenteen muuttaminen sekä tietojen salaus esimerkiksi käyttämällä VPN-teknologiaa (Virtual Private Network) (Gibson et al, 2012).

PaaS-mallissa työkalut ja ympäristö ovat palvelussa valmiina, mutta tästä seuraavina ongelmina voi seurata lukittautuminen tiettyyn tarjoajaan, koska ympäristöt ja työkalut eroavat ja yhteensopivuudet ovat epävarmoja. Ongelma korostuu varsinkin, jos kyseessä on “täysi” PaaS, jossa kaikki tehdään verkkorajapinnan kautta, mutta silloin asiakkaan tarvitsee tehdä kaikkein vähiten hallinnoimista itse. Toinen aiheeseen liittyvä haaste on uusien alustojen ja rajapintojen opettelu, joissa tapahtuu jatkuvaa kehitystä ja muutosta. (Gibson et al, 2012) PaaS-mallissa on siis vähemmän vapautta, mutta vähemmän työtä. Tämä malli ei todennäköisesti ole hyvä ratkaisu, jos ohjelmisto vaatii alemman tason ohjelmiston tai laitteiston kustomointia (Freet et al, 2015). PaaS-mallissa tietoturvan kannalta ongelmana ovat pilvialustalla usein tehokkuuden takia ylennetyillä oikeuksilla toimivat palvelut, jotka voivat vaarantaa turvallisuutta, jos niihin päästään murtautumaan. Rajoitusten tulee olla tarkat, jotta toisten dataan ei voi päästä käsiksi. Hyvänä puolena PaaS-mallissa olemassa olevat avoimen lähdekoodin ratkaisut voi varmistaa ja testata tietoturvan kannalta perusteellisesti kolmansien osapuolien toimesta. (Gibson et al, 2012) Myös lokeja on mahdollista tallentaa, ja käsiteltävät tiedot voidaan salata. Toisaalta salaus tuo kuitenkin usein paljon ongelmia esimerkiksi indeksoinnin ja lajittelun kanssa. (Freet et al, 2015)

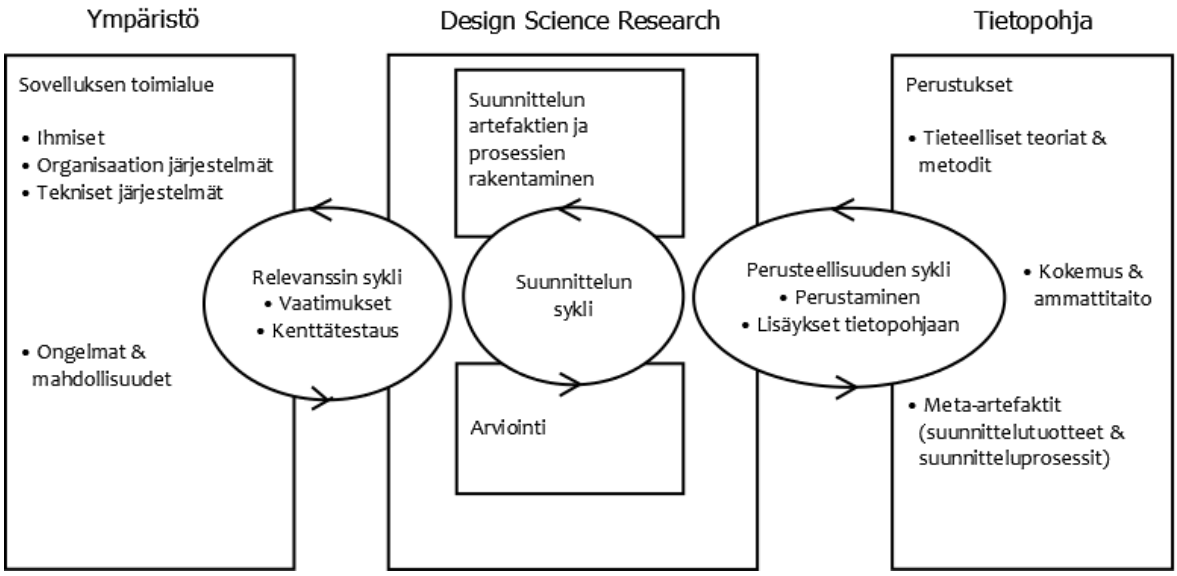
SaaS-mallissa asiakkaalle tarjotaan ohjelmistoa, jota jaetaan verkon kautta (Gibson et al, 2012). Se on siis palvelumalleista korkeimmalla tasolla, ja alustan sijaan asiakkaalle tarjotaan valmista applikaatiota. Etuna SaaS-ohjelmistoissa perinteisiin ratkaisuihin

verrattuna on niiden halpuus (Gibson et al, 2012) sekä helpompi ylläpito, yhteensopivuus ja laaja pääsy (Freet et al, 2015). Koska palvelu on niin korkealla tasolla, tietoturvan eteen ei itse voi tehdä kovin paljoa, ja asia on suurelta osin riippuvainen tarjoajasta (Freet et al, 2015). Tietoturvan voi kuitenkin nähdä joiltain osin jopa parantuvan, kun tärkeitä tietoja ei voi enää esimerkiksi varastaa kannettavalta tietokoneelta (Gibson et al, 2012).

3 RATKAISUMENETELMÄT

Tutkimuskysymyksen ratkaisemiseksi ja asetettujen tavoitteiden saavuttamiseksi tullaan käyttämään Design Science Research -tutkimusmenetelmää (DSR). DSR:ssä suunnittelija vastaa ihmisten ongelmille relevantteihin kysymyksiin luomalla innovatiivisia artefakteja ja tietoa niiden avulla. Artefaktilla tarkoitetaan jotain keinotekoista, ihmisen luomaa asiaa, ja IT:n kontekstissa niiden voidaan yleisesti määritellä olevan käsitteitä, malleja, metodeja, toteutettuja järjestelmiä tai parannettuja suunnittelun teorioita. (Hevner ja Chatterjee, 2010) Tässä työssä luotavat artefaktit ovat tilannekuvajärjestelmän tietokannan pilvialustalle toteuttamista ohjaava malli sekä case-tilannekuvajärjestelmälle tehty toteutus. Case-järjestelmän kaikkia yksityiskohtia ei voida kuvata työssä tarkasti, koska tiedot kuuluvat Observis Oy:lle.

Toisin kuin ammattitaitoinen suunnittelu, DSR ei tarkoita olemassa olevan tiedon käyttöä ongelmien ratkaisemiseksi, vaan uniikkia tai innovatiivista ratkaisemattoman ongelman käsittelyä tai tehokkaampaa vastausta jo ratkaistuun ongelmaan. DSR-menetelmään kuuluu selvä lisäys tietopohjaan ja sen kommunikointi. (Hevner ja Chatterjee, 2010) Tutkimus suoritetaan iteratiivisissa sykleissä, jotka varmistavat, että työ on relevantti käyttöympäristössään, suunnittelu ja tulosten arviointi tehdään huolellisesti, ja työ perustetaan ja sen tulokset lisätään tietopohjaan. Kuva 3.1 kuvaa prosessin syklejä.



Kuva 3.1. DSR-tutkimuksen syklit (mukailtu lähteestä Hevner 2007).

Työ on relevantti käyttöympäristössään, koska se tehdään yrityksen käytännön tarpeeseen. Perustaminen tietopohjaan tehdään ottamalla huomioon alueella aikaisemmin tehdyt työt, ja tulokset tullaan kommunikoimaan julkaisemalla kandidaatintyö.

4 TULOKSET

4.1 Suunnittelun lähtökohdat

Työn käytännön osuutta varten tehdään alustava toteutus case-järjestelmän tietokannalle pilviympäristöä varten. Pilvialustan palvelumalleista vaihtoehtoja toteutukselle ovat PaaS- ja IaaS-palveluiden käyttäminen. Tällä hetkellä case-järjestelmästä on Google Cloud -palvelimella alustava toteutus, johon ei ole vielä tehty rakenteellisia muutoksia. Turvallisuuden kannalta kriittisimmissä tapauksissa asiakkaat haluavat todennäköisesti yhä käyttää paikallista järjestelmää. Case-järjestelmän tapauksessa on siksi tärkeää huomioida, että pilvitoteutus tulee olemaan vaihtoehto paikallisen asennuksen rinnalla eikä korvaa sitä. Tämä piirre tekee PaaS-mallista IaaS-mallia vähemmän houkuttelevan, koska siinä ei voi hyödyntää yhtä hyvin olemassa olevaa järjestelmää varten tehtyä kehitystyötä. Toisin kun IaaS-palveluissa, PaaS-palveluissa käytössä olevat teknologiat ovat rajoittuneet tarjoajan ennalta päättämään kokoelmaan. IaaS-mallissa kustomoinnin ja hallinnan aste on suurempi, ja järjestelmässä hyödynnetään valmiiksi Docker-säiliöintiä, minkä ansiosta sen toteuttaminen IaaS-alustalla voidaan tehdä melko pienellä työmäärällä. Ylläpidettävyys paranee huomattavasti paikallisen järjestelmän ja sen pilviversion ollessa mahdollisimman samanlaisia. Tärkeää olisi huomioida myös mahdollisimman pieni sitoutuminen tiettyyn alustaan tai teknologiaan. Liian suuri rajoittuminen voi aiheuttaa tulevaisuudessa paljon lisää työtä alustassa tapahtuvien muutosten takia, varsinkin jos myöhemmin halutaan lähteä muuttamaan järjestelmän kehityksen suuntaa. PaaS-mallin käyttäminen tarkoittaisi vähemmän vapautta, enemmän tiettyyn alustaan sitoutumista ja tässä tapauksessa se vaatisi myös enemmän kehitystyötä, minkä takia pilvitoteutusta aletaan kehittämään IaaS-alustalle.

Mahdollisia arkkitehtuurimalleja pilvitoteutukselle on useita. Tärkeimpiä muutoksia tietokannalle on siirtyminen yhden organisaation rakenteesta skaalautuvaan, usean asiakkaan käytössä olevaan versioon. Yksi mahdollisuus on tehdä koko systeemille yksi suuri tietokanta, jossa kaikkien asiakasorganisaatioiden tiedot olisivat. Toinen vaihtoehto on toteuttaa skaalautuminen käyttämällä erillistä tietokantaa jokaiselle asiakkaalle. Tärkeimmät edut yhdellä tietokannalla ovat helpompi hallittavuus ja mahdollisten rakenteellisten muutosten tekeminen tulevaisuudessa, koska huomioon ei tarvitse ottaa kuin yksi tietokanta.

Suurimpia ongelmia taas ovat useat rakenteelliset muutokset case-järjestelmän tämänhetkiseen tietokantamalliin ja ohjelmistoon yleensäkin, sekä riskit esimerkiksi tietojen sekoittumiseen ja tietoturvaan liittyen. Uusi rakenne tulisi ensin suunnitella, ja lisäksi mahdolliset järjestelmän ongelmatilanteet ja muutokset tulisivat aina koskemaan kaikkia asiakkaita. Vastaavasti usean tietokannan käyttämisessä etuja ovat pienet muutokset koodipohjaan ja mahdollisuus tehdä päivitykset ja kustomoinnit asiakaskohtaisesti. Ilmeinen ongelma usean tietokannan ratkaisussa on, että joudutaan samanaikaisesti ylläpitämään kaikkien asiakkaiden tietokantoja erikseen, ja suurella asiakkaiden määrällä päivitysten tekeminen voi myös aiheuttaa paljon työtä.

Toteutuksen tulisi olla hyvin eteenpäin kehitettävä vaatimusten ja teknologioiden muuttuessa tulevaisuudessa. Tästäkin näkökulmasta järkevässä mittakaavassa pienin muutoksin tehty versio vaikuttaisi hyvältä ratkaisulta, sillä se säilyttäisi paremmin vaihtoehtoja erilaisiin suuntiin kehittymisessä tulevaisuudessa. Jos toteutus tehdään suuremmalla työllä tiettyyn uuteen malliin, on todennäköisesti vaikeampaa muuttaa kehityksen suuntaa toiseksi myöhemmin.

4.2 Case-järjestelmän nykytila

Työssä ei kuvailla case-järjestelmän rakennetta tai koodin muutoksia yksityiskohtaisella tasolla, koska ne ovat Observis Oy:n omistuksessa olevaa tietoa. Järjestelmän perusidea on hallita useista mittalaitteista koostuvaa järjestelmää. Laitteet voivat mitata esimerkiksi haitallisia kaasuja, GPS-koordinaatteja (Global Positioning System) tai radioaktiivista säteilyä. Olennaisimpiin ominaisuuksiin kuuluu järjestelmään yhdistettyjen laitteiden havaintojen eli mittausarvojen tarkasteleminen, mahdollisten hälytysten hallinnoiminen ja interaktiivinen kartta. Mittalaitteiden hälytykset on integroitu järjestelmään, ja niihin liittyviä kriteerejä voi muokata. Mukana on myös useita toimintaa tukevia ominaisuuksia, joista tärkeimpiä ovat tehtävien (engl. mission) ja osatehtävien (engl. task) lisääminen, NATOn (North Atlantic Treaty Organization) ATP-45-standardin mukaisten CBRN-raporttien luominen, tärkeiden sijaintien merkintä kartalle ja laitteiden huoltotöiden hallinta. Tehtävillä tarkoitetaan käyttäjien luomia operaatioita, joissa esimerkiksi tietyssä sijainnissa tulee käydä suorittamassa jotain toimenpiteitä, ja osatehtävät ovat niihin mahdollisesti

liittyviä osakokonaisuuksia. Samalla järjestelmällä voi olla useita käyttäjiä, joilla voi olla erilaisia sisäisiä käyttöön liittyviä oikeuksia. Järjestelmässä käytetty termistö on englanniksi, ja tässä työssä termit on suomennettu pyrkien ilmaisemaan niiden merkitys mahdollisimman selkeästi.

Tämänhetkisessä järjestelmän rakenteessa havaintoja käsitellään pitkälti niiden tuottaneen laitteen kautta. Rakenne siirtyisi melko pienellä työllä uuteen versioon, jossa asiakasorganisaatiot on erotettu toisistaan. Koska eri laitteet itsessään kuuluvat jo valmiiksi aina tietylle organisaatiolle, havaintojen saaminen mukaan olisi yksinkertaista. Jokainen laite itsessään liittyy myös yhteen laitehallintayksikköön (engl. device manager), joita voi olla järjestelmässä yhteensä yksi tai useampi. Lisäksi järjestelmään on tallennettu kokoelma mahdollisia ilmiöitä, joita laitteet voivat mitata. Jokaiseen laitteeseen liittyy ainakin yksi ilmiö, joka voisi olla esimerkiksi kaasun konsentraatio.

Järjestelmässä käytetään tietojen tallennuksen PostgreSQL-relaatiotietokantaa. Nykyisen tietokannan lisäksi myös NoSQL-tyyppisen tietokannan käyttö voisi olla yksi mahdollisuus tulevaisuudessa pitkäaikaista suurten datamäärien tarkempaa analysointia ja tallennusta varten. Viestintään käytetään suurimmaksi osaksi MQTT- ja REST-palveluita. MQTT (Message Queuing Telemetry Transport) on IoT-teknologian käyttöön tarkoitettu viestintäprotokolla, joka noudattaa julkaisu/tilaus -mallia (engl. publish/subscribe) (MQTT.org, 2020). Viestintä toimii aiheiden avulla (engl. topic), joihin voi julkaista viestejä, jolloin kaikki aiheen tilanneet asiakasohjelmat saavat viestin. Aiheet voidaan luoda ja järjestää hierarkkisesti. REST (Representational State Transfer) on verkkopohjaisten järjestelmien arkkitehtuurityyli, joka luo rajapinnan järjestelmän kommunikaatiolle asettamalla tiettyjä rajoitteita (Fielding, 2000). Järjestelmän REST-kommunikoinnissa käytetään HTTP-viestintäprotokollaa (Hypertext Transfer Protocol). Osa komponenteista käyttää vielä toista viestijärjestelmää, josta mahdollisesti siirrytään tulevan kehitystyön myötä pois.

Järjestelmä koostuu Docker-säiliöistä, joista jokainen hoitaa omaa tehtäväänsä. Yksinkertaisin tapa skaalata järjestelmää pilvessä olisi monistaa säiliöt eri asiakkaille. Pienillä muutoksilla tehty toteutus voi aiheuttaa enemmän työtä ylläpitämisessä, mutta koodipohjan pitäminen mahdollisimman samanlaisena eri versioissa on myös hyvin tärkeää

kokonaisuuden ylläpidettävyyden kannalta. Lisäksi on mietittävä, kuinka suurella asiakasmäärällä on ylipäätään kannattavaa tehdä suuria muutoksia järjestelmän tehostamiseksi ja ylläpidon automatisoimiseksi pilvialustalla. Jos asiakkaiden määrä on tarpeeksi pieni, automatisoiminen aiheuttaa enemmän työtä kuin järjestelmän manuaalisempi ylläpitäminen.

4.3 Toteutuksen tavoitteet ja vaatimukset

Vaatimuksissa päädyttiin siihen tulokseen, että vaihtoehdoiksi halutaan kaksi erilaista pilvitoteutusta: geneerinen versio, johon kuka vain voi rekisteröityä Internetissä (tässä työssä käytetään nimitystä “yleisversio”), ja asiakaskohtaisempi toteutus, jossa on paremmat mahdollisuudet kustomointiin ja tietoturvaan (tässä työssä käytetään nimitystä “asiakaskohtainen versio”). Yleisversio on halvempi, toiminnoiltaan yksinkertaisempi, eikä sen tietoturva ole yhtä korkealla tasolla. Tämä vaihtoehto on tarkoitettu erityisesti sellaisille asiakkaille, joilla ei ole tietoturvaltaan kovin kriittistä tietoa. Kalliimpi asiakaskohtainen versio sopii parempaa tietoturvaa ja kustomointia haluaville.

Liikkeelle lähdettiin siitä, että yleisversiossa käytetään yhtä tietokantaa kaikille asiakkaille, ja asiakaskohtaisessa versiossa kunkin asiakkaan järjestelmä ja tietokanta on heille yksityinen. Yleisversiolle mahdollisuutena tarkasteltiin lyhyesti myös erillisten PostgreSQL:n skeemojen (engl. schema) käyttöä eri asiakkaille, mutta tässä vaihtoehdossa on huolena huono suorituskyky käyttäjämäärien kasvaessa, jos tästä vaihtoehdosta halutaan suuremmillekin asiakasmäärille skaalautuva. Vaikka yhden tietokannan käyttäminen vaatiikin jonkin verran muutoksia nykyisen kannan rakenteeseen, yhtä tietokantaa olisi helpompi hallita ja ylläpitää. Asiakaskohtaisenkin version ostajilla voi olla aliorganisaatioita tai kumppaneita, joiden tietoja olisi järkevää pitää lähempänä toisiaan, ja tietokannan sisäinen organisaatiorakenne voisi olla sopiva ratkaisu tähänkin. Tämän takia tietokannan malliin tulisi alustavassa toteutuksessa kuitenkin tehdä rakenne organisaatioille. Tämä lisää järjestelmään paljon joustavuutta, ja mahdollistaa yhdenkin tietokannan hyödyntämisessä enemmän vaihtoehtoja. Hyvin tärkeää on pitää eri versioiden koodin eroavaisuudet mahdollisimman pienenä ylläpidettävyyden takia. Mahdollinen tietokannan sisäisen

organisaatiorakenteen vaativa versio tulisi siis pyrkiä toteuttamaan siten, että se voidaan sisällyttää muihinkin versioihin ilman merkittävää vaikutusta niiden tehokkuuteen.

Usean asiakkaan käytössä olevaan järjestelmään siirryttäessä luonnollisesti tietokannan lisäksi myös muut osat vaativat muutoksia. Skaalautuvuus voidaan toteuttaa eri tavoin eri osissa. Toisia voidaan monistaa esimerkiksi säiliöinnin tasolla, mutta periaatteessa vaikkapa sisäisten viestien välitykseen voisi käyttää vain yhtä palvelua. Yksinkertainenkin järjestelmän ja tietokannan toteutus usealle asiakkaalle tai aliorganisaatiolle vaatii muutoksia esimerkiksi MQTT-viestien aiherakenteeseen. Omistavan yrityksen tai asiakkaan voi laittaa viestirakenteessa korkeimmalle tasolle, jolloin sen alla olevat tiedot voidaan ohjata oikeaan paikkaan. Tietojen salaus on mahdollista tehdä aiheen tasolla, jolloin muut eivät pääse käsiksi viestien hierarkiassa alemman tason aiheissa oleviin asiakkaan tietoihin.

Suunnittelutyössä saatavan käytännön lopputuloksen tarkoitus ja tavoite on olla proof of concept -tyyppinen toteutus järjestelmästä, joka voi toimia lähtökohtana jatkokehitykselle. Työssä pohditaan myös vaihtoehtoja tulevaisuudessa tehtäville muutoksille. Vaatimusten ja pohdintojen pohjalta järjestelmän pilvitoteutuksella ja samalla sen tietokannalla tulisi olla ainakin kaksi erilaista tyyppiä; halvempi ja yksinkertaisempi yleistoteutus ja täydemmillä ominaisuuksilla varustettu asiakaskohtaisempi toteutus. Lisäksi asiakaskohtaisemmissa toteutuksissa tulee kiinnittää huomiota erityistapauksena asiakkaisiin, joiden kumppani- tai aliorganisaatiot käyttävät myös järjestelmää. Tätä työtä varten tehtävässä toteutuksessa keskitytään vain yleisversioon. Huomioitavaa on myös, että tehtäviä muutoksia ei välttämättä toteuteta varsinaisessa tuotteessa sellaisenaan, eikä kaikkien muutosten ei odoteta olevan lopullisia. Kokonaisuuteen liittyy useita asioita, joita on hyvin haastavaa selvittää varmaksi ennen niiden toiminnan tarkastelua käytännössä. Yleisestikin case-järjestelmää kehitetään iteratiivisesti, joten viimeiseen versioon ei olekaan tarkoitus päätyä välittömästi.

Suoraviivaisin tapa toteuttaa usean asiakkaan järjestelmä on yksinkertaisesti monistaa järjestelmän muodostavia Docker-säiliöitä. Osalle komponentteja voi kuitenkin olla mahdollista tehdä myös hienostuneempaa hajautusta. On tietenkin mahdollista tehdä aluksi yksinkertaisempi toteutus, jossa muutokset on pidetty mahdollisimman pienenä ja jota voi optimoida lisää tulevaisuudessa.

Työssä toteutetaan prototyyppi yleisversiosta, jossa usean asiakkaan rakenne on integroitu tietokannan malliin. Tarvittaviin tietokannan tauluihin siis lisätään asiakkaan tai organisaation tunniste, ja tulee suunnitella, mihin kaikkiin se lisätään. Punnittavana on tietokannasta tietoa hakevien kyselyjen tehokkuus ja toisaalta tallennustila ja tietojen lisäyksen tehokkuus, joihin vaikuttaa, kuinka moneen tauluun tunniste lisätään. Muutettavia rakenteita koskevia kyselyitä täytyy myös päivittää. Vaikka tietokanta on tarkastelun keskipisteenä, työ ja varsinkin sen demonstroiminen vaatii muutoksia muihinkin komponentteihin, jotka tulee myös toteuttaa. Tarvittavat muutokset tehdään komponentteihin, jotka käsittelevät rakenteellisesti muutettavia tietokannan tietoja.

4.4 Käytännön toteutus

Työtä varten toteutettiin proof of concept -järjestelmä, jossa tarvittaviin tietokannan tauluihin lisättiin organisaation tunniste, ja taulujen tietoja hyödyntäviä komponentteja kehitettiin muutosten vaatimalla tavalla. Muutosten hyödyntämiseksi ja demonstroimiseksi usean muunkin järjestelmän komponentin rakennetta piti muuttaa tarvittavilta osin. Järjestelmässä tehdään vielä raskasta kehitystyötä, ja muutama osa jätettiin pois uudistetusta rakenteesta, koska niiden rakennetta on tarkoitus vielä muuttaa. Tätä työtä varten tehtävä kehitys menisi siis suurelta osin hukkaan. Erityisesti front-endin puolella toteutus jätettiin myös varsin alustavalle tasolle, eikä se liitykään suoraan tietokannan rakenteeseen.

Tietokannan tiedoista selvästi kullekin asiakkaalle yksityisiä ovat laitteet ja niiden havainnot, käyttäjät, tehtävät, hälytykset ja niihin liittyvät konfiguraatiot, ATP-45-raportit sekä tallennetut sijainnit. Lisäksi huoltotöiden hallintaan liittyy yksityisiä osia. Edellä mainittujen kokonaisuuksien yksityiskohtaisempi tietokantarakenne on Observis Oy:n yksityistä tietoa, mutta näiden kokonaisuuksien tärkeimpiin tauluihin lisättiin organisaation tunniste paria poikkeusta lukuun ottamatta. Koska käyttäjät ja tallennetut sijainnit käyttävät vielä vanhempaa viestintäjärjestelmää, josta todennäköisesti siirrytään tulevaisuudessa, niiden toteutus jätettiin pois. Uusien viestintärakenteiden tekeminen menisi selvästi työn laajuudesta yli. Lisäksi huolto-ominaisuuksien rakenne tullaan todennäköisesti refaktoroimaan lähitulevaisuudessa, joten senkin toteutus jätettiin työstä. Toteutuksesta pois

jätetyt osat siis tarvitsisivat vielä tulevaisuudessa muutoksia, jotta usean organisaation rakenne toimisi. Huomioitavaa on myös, että organisaation tunniste on mahdollista lisätä useaan muuhunkin tietokannan tauluun kuin nyt tehdyssä toteutuksessa. Tällöin tietokantaan tulisi enemmän tiedon toistoa, mutta kyselyjen tekeminen olisi yksinkertaisempaa ja ne olisivat tehokkaampia.

Osa rakenteista jätetään ilman muutoksia, koska niiden tiedot voivat hyvin olla asiakasorganisaatioiden kesken jaettuja. Näitä ovat tiedot järjestelmän tukemista käyttäjien rooleista, tiedot eri laitetyypeistä sekä tuetut kielet. Yleisversioissa, jota varten kehitystyö tehtiin, on tarkoitus olla ennalta valitut vaihtoehdot ilmiöille, joita laitteet voivat mitata. Tällöin ne voidaan myös pitää asiakkaiden kesken jaettuna tietona. Asiakaskohtaisessa versiossa käyttäjät pääsevät mahdollisesti itse konfiguroimaan ja lisäämään ilmiöitä, mutta tämä jätetään joka tapauksessa käytännön työstä.

Jotta organisaatio saadaan kunkin järjestelmässä olevan laitteen tietoihin, laitehallintayksikön konfigurointiin lisättiin organisaation tunniste yhdeksi kentäksi, josta tieto saadaan sen alaisille laitteille. Yksikkö ja siinä olevat laitteet kuuluvat aina kokonaisuudessaan vain yhdelle organisaatiolle.

Järjestelmän front-endiin lisättiin tieto järjestelmää käyttävän organisaation tunnisteesta sovelluksen tilan tietoihin. Toteutus tehtiin siten, että kirjautuessa tunnisteen voi yksinkertaisesti syöttää järjestelmään. Tulevaisuudessa tunniste tulisi integroida suoraan käyttäjätietoihin. Tätä ei voitu ottaa mukaan nyt tehtävään toteutukseen, koska käyttäjän tietokantarakenteen uudistaminen tehdään myöhemmin. Suurin motivaatio tunnisteen lisäämisessä front-endiin ylipäätään oli mahdollisuus hyödyntää ja demonstroida käytännössä järjestelmän rakenteeseen tehtyjä muutoksia. Myös käyttäjälle front-endissä näkyvät laitteet ja hälytyksille asetetut konfiguraatiot muutettiin huomioimaan vain tilatiedoissa olevalle organisaatiolle julkaistut tiedot.

ATP-45-raportteja tallennettaessa tilatiedoista saatu organisaation tunniste välitetään tietoihin mukaan, ja haettaessa aiempia raportteja haku tehdään vain käyttäjän organisaation raporteista. Samoin tehtäviä luotaessa organisaation tunniste välitetään mukaan ja tallennetaan, ja aiemmin tallennettuja tehtäviä haettaessa tuloksia etsitään vain tällä

tunnisteella. Asetettaessa hälytysrajoille uusia konfiguraatioita ne tallennetaan organisaatiotunnisteen kanssa. Hälytysrajoja havaintoihin vertaavaa ohjelmiston komponenttia, joka tunnistaa ja laukaisee hälytykset, ei kuitenkaan muokattu ottamaan huomioon organisaatioita. Tarvittavat muutokset tämän osan rakenteeseen olisivat olleet merkittäviä, eikä tällä osalla ole suoraa yhteyttä tietokannan rakenteen kanssa, eli se ei varsinaisesti kuulukaan työn laajuuteen.

Nyt tehdyssä toteutuksessa tunniste on pelkkä syötettävä nimi. Varsinaisessa järjestelmässä tulisi käyttää hienostuneempaa uniikkia tunnistetta, joka luodaan jokaiselle käyttäjäorganisaatiolle. Lisäksi aliorganisaatioiden välistä suhdetta ei vielä lisätty omaan taulurakenteeseensa, jossa ylemmän tason roolilla olisi mahdollisuus toimia alempien oikeuksilla. Potentiaalinen rakenne varsinaiselle lopulliselle järjestelmälle olisi erottaa käyttäjien ja organisaatioiden metadatat kokonaan omiin tietokantoihinsa, ja pitää ne erillään tilannekuvaan keskittyvistä rakenteista. Tällöin muut järjestelmän osat toimivat niille välitettyjen mahdollisesti väliaikaisten tunnuksien avulla käyttäen niitä eri toimintojen todentamiseen.

Jotta uudistettu rakenne voisi toimia, MQTT-viestinnän toimintaa muutettiin siten, että kullekin organisaatiolle luodaan aina oma aihe, ja suurin osa aiemmista viestintään käytetyistä aiheista siirrettiin näiden organisaation tunnisteella erotettujen aiheiden alaisiksi. Osa järjestelmän tietokantaan liittymättömään kommunikointiin kuuluvaa viestintärakennetta jätettiin kuitenkin vielä ennalleen yleiselle tasolle, kuten esimerkiksi jo aiemmin mainittu hälytyksiin liittyvä viestintä.

Erittäin tärkeä alustavassa käytännön toteutuksessa suureksi osaksi huomiotta jätetty osa on tietoturvan varmistaminen. Tämä puoli päätettiin jättää proof of concept -toteutuksesta pois, koska se on jo itsessään laaja aihealue, ja työ keskittyi enemmän varsinaiseen toiminnallisuuteen. Lisäksi yksityiskohdat olisivat monelta osalta vaatineet järjestelmän muun arkkitehtuurin huomioimista ja eri osien varsinaista toteutusta. Koska ohjelmiston toteutuksen yksityiskohdat ovat yksityistä tietoa ja varsinkin tietoturvaan liittyvät osat ovat varsin kriittistä tietoa, työtä olisi joka tapauksessa ollut hyvin haastavaa raportoida. Esimerkiksi nyt kehitetyissä REST-rajapinnan kutsuissa käytetään organisaation tunnistetta suoraan, ja varsinaisessa toteutuksessa palveluihin tulisi liittää salaus, jotta kuka tahansa

tunnisteen tietävä ei voi käyttää kutsuja tietojen hankkimiseen. Lisäksi todennukset tulisi liittää uudistettuun käyttäjän rakenteeseen. MQTT-viestinnässä salaus on mahdollista tehdä aiheen tasolla, joten olisi melko yksinkertaista salata kunkin organisaation viestintä erikseen.

4.5 Tulosten yleistäminen

Työssä käsitelty case-järjestelmä on yksittäistapaus, mutta saatuun tietoon pohjaten voidaan esittää ratkaisusuosituksia myös yleisemmällä tasolla tilannekuvajärjestelmän tietokannan toteuttamiseen pilviympäristössä. Eri järjestelmillä voi olla paljon niitä erottavia ominaisuuksia, jotka vaikuttavat siihen, millainen ratkaisu soveltuisi kuhunkin tapaukseen parhaiten. Kaksi tärkeää näkökulmaa toteutusta suunnitellessa ovat pilvipalvelun tyyppi ja tietokannan toteutus siinä, erityisesti tulisiko tietokantoja olla useita vai vain yksi. Pilvipalvelun tyypin valitsemisen suhteen olennaisia päätöksiin vaikuttavia kysymyksiä ovat:

- Onko koko järjestelmä pilvialustalla vai ainoastaan tietokanta?
- Halutaanko itse hallittava ympäristö, jolle kehittää ohjelmisto, vai maksaa palvelusta, jossa valmiille alustalle voi rakentaa ohjelmiston tarjotuilla työkaluilla?
- Ollaanko siirtämässä olemassa olevaa järjestelmää pilvialustalle vai luomassa uutta järjestelmää?
- Jos ollaan harkitsemassa PaaS-palvelua, onko hyväksyttävää tehdä lisäksi kehitystyö, jossa järjestelmä muokataan tarjotuilla työkaluilla toteutetuksi?

Järjestelmän toteutuksessa pilviteknologiaa on mahdollista hyödyntää vain tietokannassa pitäen muun järjestelmän paikallisena. Kuten Kulkarni et al (2012) kertovat, erilaisia pilvialustalla tarjottavia varastointipalveluja on runsaasti, ja niiden suurimpana etuna perinteisiin järjestelmiin nähden ovat säästöt henkilöstössä ja laitteistossa. Jos pilvipalveluita halutaan siis hyödyntää vain tietokannan kanssa tavoitteena säästää pääomaa, on saatavilla paljon SaaS-palveluja tarvitsematta pitää koko järjestelmää pilvialustalla.

Jos pilviteknologiaa halutaan hyödyntää koko järjestelmässä, ovat vaihtoehtoina PaaS ja IaaS-palvelut. Näiden erona PaaS-mallissa asiakkaalle tarjotaan kokoelma ohjelmointikieliä,

kirjastoja, palveluita ja työkaluja, joiden avulla voi ottaa käyttöön ohjelmistoja pilvialustalla, mutta IaaS-mallissa asiakkaalle tarjotaan perustavanlaatuisia tietojenkäsittelyresursseja, joita käyttäen voi ajaa mielivaltaisia ohjelmistoja tai käyttöjärjestelmiäkin (Mell & Grance, 2011). Valintaan näiden välillä vaikuttaa siis, kuinka paljon halutaan vapautta kehityksessä käytettävien työkalujen valinnassa ja kuinka paljon ollaan valmiita tekemään työtä alemman tason ympäristön hallinnoimisessa. Huomattavaa on kuitenkin, että pilvialustalle voidaan olla siirtämässä olemassa olevaa järjestelmää uuden kehityksen sijasta, kuten työn case-järjestelmän tapauksessa. Jos ollaan luomassa kokonaan uutta ohjelmistoa, PaaS-palvelu voidaan ottaa käyttöön ongelmitta, mutta siirrettäessä jo tehtyjä ohjelmistokomponentteja ei ole takeita siitä, että PaaS-alustalla käytössä olevat työkalut sisältävät aiemmassa toteutuksessa käytetyt. Vaikka PaaS olisi muuten toivottava vaihtoehto, ohjelmiston osien muokkaamisesta aiheutuva lisätyö voi olla niin merkittävää, että onkin parempi siirtää valmiiksi kehitetyt osat itse hallittavaan IaaS-ympäristöön.

Tietokannan toteutuksen suhteen olennaisia kysymyksiä ovat:

- Koostuuko koko järjestelmä monista eri omistajien tai eri paikoissa sijaitsevista järjestelmistä?
- Jos järjestelmä sisältää eri kohteita, onko niiden data muiden kesken jaettua vai yksityistä tietoa?
- Onko datan tietoturvan kriittisyys suuri?

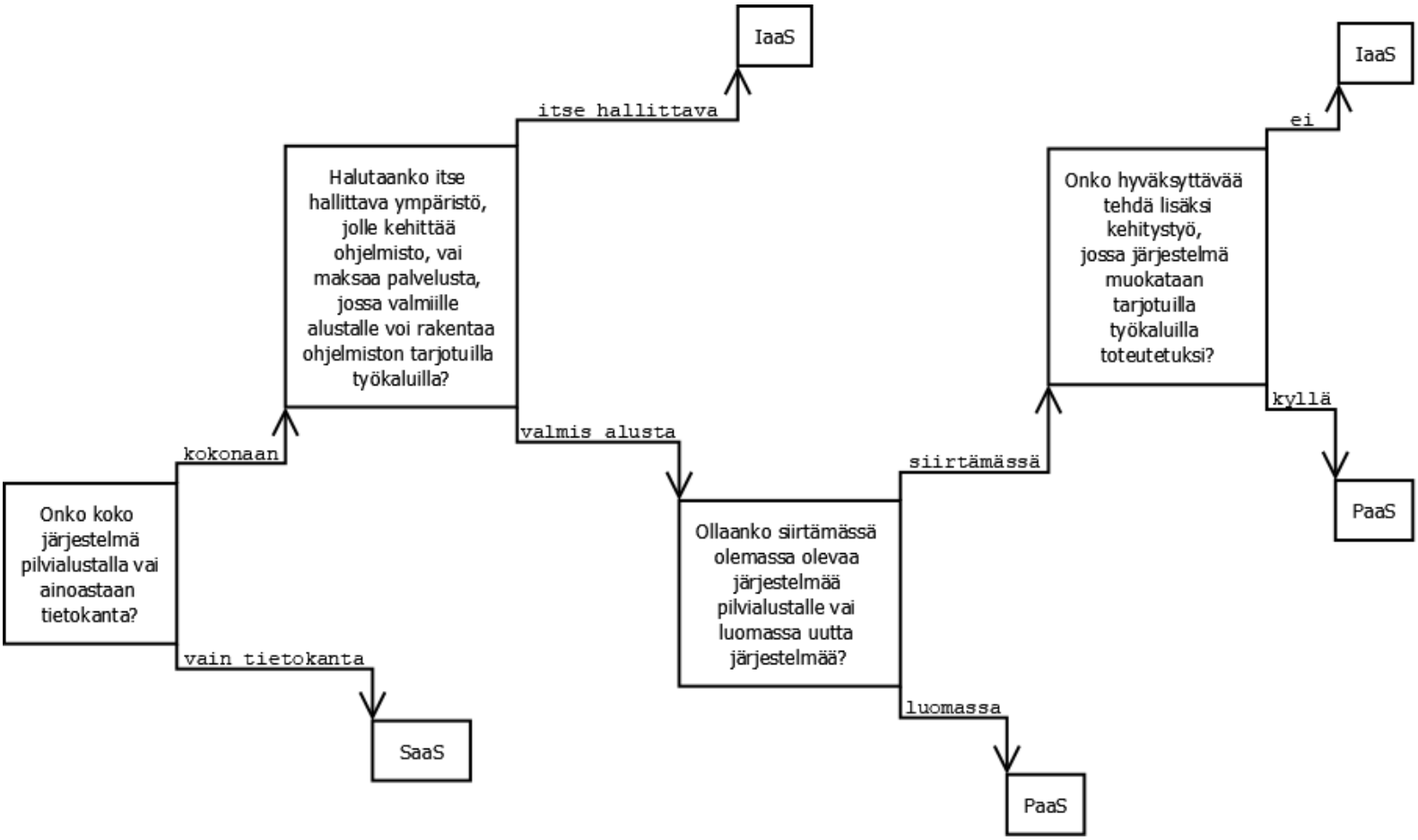
Järjestelmä voi olla joko ainoastaan yhden asiakkaan tai organisaation käytössä, kuten case-järjestelmän paikalliset toteutukset, tai siihen voi liittyä useissa eri sijainneissa useille eri organisaatioille kuuluvia järjestelmiä, kuten case-järjestelmälle kehitteillä olevassa pilvitoteutuksen yleisversiossa. Jos järjestelmä kuuluu kokonaan yhdelle organisaatiolle, ei yleisesti ottaen ole tarvetta kuin yhdelle tietokannalle. Vaikka järjestelmään kuuluisikin eri sijainneissa ja eri toimijoillakin olevia osia, tiedot voivat olla järjestelmän sisäisesti jaetuksi tarkoitettuja esimerkiksi yhteistyön takia, jolloin kaikki tiedot voidaan yhä pitää yhdessä tietokannassa. Tähän antaa viitteitä tapaus, jonka Alamdar et al (2016) esittelivät artikkelissaan. Artikkelin tapaus poikkeaa tässä työssä esitetystä siten, että siinä on kyse useiden organisaatioiden kokonaan omista systeemeistä, joissa sensorit keräävät tietoa organisaatioiden omiin tietokantoihin. Sensoridatat kuitenkin viedään lopuksi jaetuksi

tiedoksi yhteiseen yleistietokantaan. Koska tässä tapauksessa on kyse useiden järjestelmien tuottaman tiedon myöhemmästä yhdistämisestä, useiden omien tietokantojen käyttö on ymmärrettävää. Jaettu yleistietokanta kuitenkin viittaa siihen, että jo alun perin yhdeksi kokonaisuudeksi kehitetyssä järjestelmässä data voidaan säilöä yhteen tietokantaan.

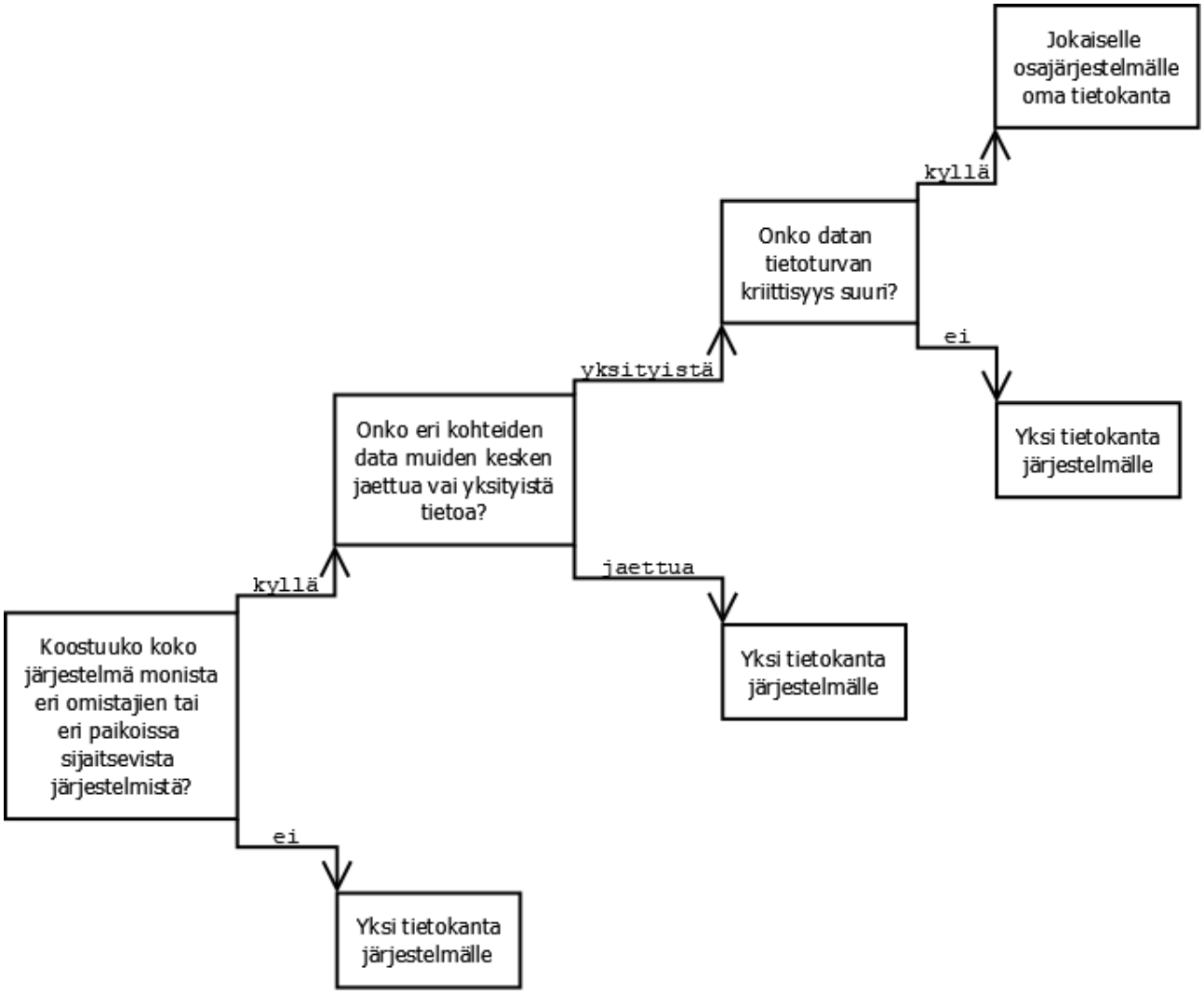
Tilannekuvajärjestelmää voivat käyttää myös useat erillisesti toimivat asiakasorganisaatiot, jotka haluavat pitää oman datansa kokonaan yksityisenä. Tällöin on mietittävänä, pitäisikö tietojen olla kokonaan omissa tietokannoissaan. Esimerkiksi Adewojo et al (2015) esittävät pilvialustalle siirtämänsä usean käyttäjän ohjelmiston, jossa käytetään yhtä asiakkaiden kesken jaettua tietokantaa. Tämä mahdollistaa resurssien tehokkaan käytön, mutta datan tietoturva ei kuitenkaan ole täysin taattu (Adewojo et al, 2015). Myös case-järjestelmää varten tehdyssä työssä esitettiin erityisesti tietoturvaltaan vähemmän kriittisiin tapauksiin tarkoitettua yleisversiota, jossa käyttäjät jakaisivat yhden tietokannan. Toisaalta esimerkiksi Slominski et al (2015) esittävät pilvialustalle siirretyssä järjestelmässään erillistä tietokantaa jokaiselle käyttäjälle. Tällä perusteella ratkaisusuosituksessa esitetään, että tietoturvan kriittisyyden ollessa suuri tulisi joka asiakkaalla olla oma tietokanta, ja muissa tapauksissa yksi tietokanta järjestelmälle olisi riittävä. Vaikka annetuissa esimerkeissä käsitellyt järjestelmät eivät ole tilannekuvajärjestelmiä, voidaan ratkaisut yleistää myös tilannekuvajärjestelmiin sopiviksi.

Huomiona esitettyihin tietokantaratkaisuihin voidaan sanoa, että järjestelmään liittyessä useaa hyvin erityyppistä dataa nämä voidaan myös sijoittaa omiin tietokantoihinsa. Tapauksessa, jonka Alamdar et al (2016) esittelivät, eri organisaatioiden esitysmuodoltaan hyvin erilaiset datakokoelmat ovat omissa kannoissaan, ja tiedot siirretään yhteiseen kantaan vasta niiden harmonisoinnin jälkeen. Ratkaisuehdotuksissa esitettyjen ominaisuuksien lisäksi tärkeä tietokantaan liittyvä kysymys on, että käytetäänkö järjestelmässä relaatiotietokantaa vai NoSQL-tietokantaa. Koska tämän työn case-järjestelmässä toteutuksessa käsiteltiin vain relaatiotietokantoja, kysymys on suurelta osin rajattu työn ulkopuolelle. Oikea ratkaisu malliksi riippuu datan ominaisuuksista, määrästä sekä esimerkiksi tietokannan operaatioiden käytöstä. Käytännön järjestelmätoteutuksiin ja pilvipalveluiden valitsemiseen vaikuttavat luonnollisesti myös esimerkiksi eri palveluiden hinnat.

Mahdollisten eri skenaarioiden ja niiden yhdistelmien määrä on huomattava. Tässä mukaan on otettu ominaisuuksia, joiden voidaan nähdä olevan valintojen kannalta olennaisia, ja ehdotetaan päätöksenteon tueksi ratkaisuja erilaisiin tapauksiin. Kuvissa 4.5.1 ja 4.5.2 ehdotukset on tiivistetty päätöksentekopuiden avulla esitetyksi. On otettava huomioon, että käytännössä tapauksiin voi liittyä muitakin tekijöitä, jotka vaikuttavat eri ratkaisujen kannattavuuteen. Ehdotukset on tarkoitettu yksinkertaistetuiksi malleiksi päätöksenteon avuksi, eivätkä siis ole absoluuttisia.



Kuva 4.5.1. Päätöksentekopuu mahdollisista pilvipalvelumalleista.



Kuva 4.5.2. Päätöksentekopuu mahdollisista tietokantarakenteista.

5 HUOMIOT TULOKSISTA JA NIIDEN MERKITYKSESTÄ

Työssä vastattiin tutkimuskysymykseen kehittämällä malli päätöksille tilannekuvajärjestelmän tietokannan toteuttamiseen pilviympäristössä, ja tekemällä myös toimiva case-järjestelmän alustava toteutus. Käytännön työn perusteella on selvää, että pilvialustalle tehdyn toteutuksen yksityiskohdat riippuvat paljon kulloinkin kyseessä olevan järjestelmän ominaisuuksista. Case-järjestelmää varten tehdyn alustavan toteutuksen piirteet olivat suurelta osin seurausta järjestelmän rakenteesta ja vaatimuksista.

Yleisten toteutusten suunnittelun avuksi tehty päätöksentekopuilla esitetty malli pyrkii tiivistämään järjestelmien olennaisten ominaisuuksien vaikutuksen siihen, mitkä toteutustavat toimivat milloinkin parhaiten. Työtä varten tutustuessa aiheeseen liittyvään aiempaan tutkimukseen selvisi, että tämän työn tarkasta aiheesta ei ole aiempaa tutkimusta, tai sitä ei ainakaan onnistuttu löytämään. Tämän takia mallia ei voitu johtaa suoraan aiempien tutkimusten tuloksista tai järjestelmätoteutuksista, vaan eri lähteiden tietoja piti soveltaa sen tuottamisessa. Mallissa esitetyt toteutukseen olennaisimmin vaikuttavat kysymykset johdettiin case-järjestelmässä suunnitteluun kriittisimmin vaikuttaneista sekä lähimmin aihetta vastaavissa tieteellisissä julkaisuissa olennaisimpina käsitellyistä ominaisuuksista. Vastaukset kysymyksiin johdettiin aiemmissa järjestelmätoteutuksissa käytetyistä ratkaisuista sekä tieteellisessä kirjallisuudessa esitetystä tiedosta.

Koska tietokannan toteuttamisen mallin luomiseksi on jouduttu käyttämään paljon esimerkkitapauksia, jotka eivät ole varsinaisia tilannekuvajärjestelmiä, on olemassa riski epätarkkuudesta. Järjestelmiä voidaan kuitenkin pitää tarpeeksi lähellä toisiaan olevina, etteivät eroavaisuudet vaikuta tässä työssä esitettyihin toteutustapoihin. Malli päätösten tekemiseksi ei myöskään ole absoluuttinen, koska käytännön tapauksissa järjestelmiin voi vaikuttaa asioita, joita ei ole huomioitu yksinkertaistuksessa.

Työssä on siis huomioitu, että tulosten yleistämiseen liittyy rajoitteita. Vaikka yleispäteviä vastauksia ei saadakaan kaikkiin kysymyksiin, suunnittelun päätösten malli ja alustava toteutus antavat silti selvästi suuntaa toteutukselta vaadituille ominaisuuksille ja erityispiirteille. Myös case-järjestelmään kehittämisessä saadut tulokset antavat mahdollisen toteutustavan tietokannan rakenteelle ja avaavat myös muita järjestelmän kokonaisuuden

ominaisuuksia. Kehitystyötä on mahdollista soveltaa samantyyppisiin järjestelmiin, vaikka yksityiskohdissa olisikin eroavaisuuksia. Työssä käsitellyt asiat antavat viitteitä erilaisista aiheeseen liittyvistä ongelmista ja niiden mahdollisista ratkaisuista. Joihinkin yksityiskohtiin liittyy yhä kysymyksiä siitä, mitkä ratkaisut olisivat optimaalisimpia erilaisissa tapauksissa, mutta osaan kysymyksistä ei voidakaan saada varmoja vastauksia ennen käytännön toimintaa aidossa käyttäjäympäristössä.

Case-järjestelmän toteutusta ei voida suoraan verrata muihin tapauksiin vastaavaa järjestelmää käsittelevien tieteellisten tekstien puutteessa. Monista läheisesti siihen liittyvistä aiheista löytyy kuitenkin varsin runsaastikin erilaisia tutkimusjulkaisuja. Useimmissa niissä on tosin vain niukasti tietoa tietokantojen sisäisestä rakenteesta, eli suora vertailu on haastavaa. Useissa julkaisuissa on silti hyödynnetty case-järjestelmän kanssa samoja teknologioita. Esimerkiksi Grothe et al (2016) käyttivät sekä PostgreSQL-tietokantaa sekä Docker-säiliöintiä sensoridatan tallentamiseen pilvialustalle. Myös Alamdar et al (2016) ja Dias et al (2018) hyödynsivät PostgreSQL-tietokantaa senoridatan tallentamiseen pilveen. Slominski et al (2015) käyttivät relaatiotietokantaa ja REST-rajapintoja vanhemman järjestelmän siirtämisestä usean asiakkaan pilvipalveluksi. Ratkaisussa käytettiin jokaiselle asiakkaalle omaa tietokantaa, kuten tässä työssä konseptitasolla esitellyssä case-järjestelmän asiakaskohtaisessa versiossa. Case-järjestelmään liittyvissä työn tuloksissa on siis monia yhtäläisyyksiä aiempiin tutkimuksiin ja järjestelmätoteutuksiin.

Työhön liittyvän aiemman tutkimuksen niukkuus asetti haasteita muiden julkaisujen tuloksiin pohjaamisessa sekä saatujen tulosten yleistämisessä. Työhön läheisesti liittyvistä aiheista löytyy kuitenkin paljon tieteellisiä julkaisuja, joita oli mahdollista hyödyntää toteutuksessa. Työssä saadut vastaukset jäävät enimmäkseen suuntaa-antaviksi, mutta tarkempiin yksityiskohtiin liittyviä kysymyksiä voi pohtia lisää jatkotutkimuksissa aiheesta.

6 YHTEENVETO

Tässä kandidaatintyössä tutkittiin, miten tilannekuvajärjestelmän tietokanta tulisi toteuttaa pilviympäristössä. Aihetta lähestyttiin case-järjestelmän avulla suunnitellen ja toteuttaen ratkaisu järjestelmän rakenteelle. Lisäksi työssä tehtiin yleinen päätöksentekopuilla esitetty malli tilannekuvajärjestelmien pilvitoteutusten suunnittelemista varten. Suunnittelun ja käytännön työn pohjana oli aiempi aihetta käsittelevä tutkimus sekä case-järjestelmän tapauksessa sen paikallisia asennuksia varten kehitetty versio.

Aihetta tutkittiin uuden tiedon luomiseksi tilannekuvajärjestelmän ja erityisesti sen tietokannan siirtämisestä pilvialustalle. Lähtökohtana työlle oli alan yrityksen käytännön tarve tutkimukselle ja kehitykselle. Työssä tuotiin vastauksia case-järjestelmän näkökulmasta ohjelmistokehitykseen sekä yleisemmällä tasolla vastaavien järjestelmien rakennetta koskeviin kysymyksiin.

Työn tuloksina suunniteltiin malli tilannekuvajärjestelmän tietokannan toteuttamiselle pilviympäristössä sekä alustava käytännön toteutus usean asiakasorganisaation käytössä olevassa case-järjestelmässä. Toteutusta varten suunniteltiin rakenne sekä vertailtiin erilaisia mahdollisia vaihtoehtoja. Lisäksi työtä varten case-järjestelmään tehtiin toiminnan demonstroinnin mahdollistavia muutoksia sen muihinkin rakenteisiin, kuten viestintäjärjestelmään. Alustava toteutus toimii pohjatyönä, joka mahdollistaa järjestelmän kehittämisen pilviympäristölle sopivaksi.

Tulokset voisivat toimia käytännössä päätösten tukena käytettävänä mallina pilviympäristössä toimivan tilannekuvajärjestelmän tietokannan kehitystyössä. Myös case-järjestelmässä tehty työ antaa tietoa esimerkistä käytännön toteutukselle. Tulokset vastaavat tutkimuskysymykseen tarjoamalla tietokannan toteutusta ohjaavan mallin sekä mahdollisen tietokannan rakenteen ja sitä hyödyntävän ohjelmiston. Työssä pohdittiin myös vaihtoehtoisia ratkaisuja. Käytännön työ antaa tietoa mahdollisuuksista toteutuksille ja luo pohjaa myös tulevaisuudessa tapahtuvalle kehitystyölle.

Työssä saatuja tuloksia voitaisiin tulevaisuudessa hyödyntää erilaisten tilannekuvajärjestelmien siirtämisessä pilviympäristöön tai kokonaan uusien järjestelmien

kehittämisessä pilvialustalle. Työn tulokset antavat suuntaa mahdollisille toteutustavoille ja kartoittavat erilaisia vaihtoehtoja arkkitehtuureille. Lisäksi työtä voitaisiin hyödyntää lisätutkimuksissa aiheesta ja käyttää apuna tarkemmin optimoitujen ratkaisujen etsinnässä. Työ antaa myös suoraan lähtökohtia case-järjestelmän jatkokehitykselle pilvialustalle tehtävää toteutusta varten.

LÄHTEET

Abawajy, J.H. & Hassan, M.M. (2017). "Federated Internet of Things and Cloud Computing Pervasive Patient Health Monitoring System," in IEEE Communications Magazine, vol. 55, no. 1, pp. 48-53, January 2017, doi: 10.1109/MCOM.2017.1600374CM.

Adewojo, A.A., Bass, J.M., Hui, K., Allison, I.K. (2015). Cloud deployment patterns: Migrating a database driven application to the cloud using design patterns. In Proceedings of the World Congress on Engineering and Computer Science (Vol. 1).

Alamdar, F., Kalantari, M., Rajabifard, A. (2016). Towards multi-agency sensor information integration for disaster management. Computers, Environment and Urban Systems Volume 56, March 2016, Pages 68-85. https://doi.org/10.1016/j.compenvurbsys.2015.11.005

Al-Dhuraibi, Y., Paraiso, F., Djarallah, N., Merle, P. (2018). Elasticity in Cloud Computing: State of the Art and Research Challenges, IEEE Transactions on Services Computing, vol. 11, no. 2, pp. 430-447, 1 March-April 2018, doi: 10.1109/TSC.2017.2711009.

Andrikopoulos, V., Binz, T., Leymann, F., Strauch, S. (2012). How to adapt applications for the Cloud environment. Computing 95, 493–535 (2013). https://doi.org/10.1007/s00607-012-0248-2

Balalaie, A., Heydarnoori, A., Jamshidi, P. (2015). Migrating to Cloud-Native Architectures Using Microservices: An Experience Report. In: Celesti A., Leitner P. (eds) Advances in Service-Oriented and Cloud Computing. ESOCC 2015. Communications in Computer and Information Science, vol 567. Springer, Cham. https://doi.org/10.1007/978-3-319-33313-7_15

Barros, V.A., Estrella, J.C., Prates, L.B., Bruschi, S.M. (2018). An IoT-DaaS Approach for the Interoperability of Heterogeneous Sensor Data Sources. In Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile

Systems (MSWIM ’18). Association for Computing Machinery, New York, NY, USA, 275–279. . DOI:https://doi.org/10.1145/3242102.3242141

Bogdanov, A.V. & Lwin, T.K. (2015). Storage database in cloud processing. Computer Research and Modeling, 2015, Volume 7, Issue 3, Pages 493–498 (Mi crm208). https://doi.org/10.20537/2076-7633-2015-7-3-493-498

Chaturvedi, K. & Kolbe, T.H. (2018). "InterSensor Service: Establishing Interoperability over Heterogeneous Sensor Observations and Platforms for Smart Cities," 2018 IEEE International Smart Cities Conference (ISC2), Kansas City, MO, USA, 2018, pp. 1-8, doi: 10.1109/ISC2.2018.8656984.

Chauhan, M.A. & Babar, M.A. (2011). "Migrating Service-Oriented System to Cloud Computing: An Experience Report," 2011 IEEE 4th International Conference on Cloud Computing, Washington, DC, 2011, pp. 404-411, doi: 10.1109/CLOUD.2011.46.

Chung, W.-Y., Yu, P.-S., Huang, C.-J. (2013). "Cloud computing system based on wireless sensor network," 2013 Federated Conference on Computer Science and Information Systems, Krakow, 2013, pp. 877-880.

Combe, T., Martin, A., Di Pietro, R. (2016). "To Docker or Not to Docker: A Security Perspective," in IEEE Cloud Computing, vol. 3, no. 5, pp. 54-62, Sept.-Oct. 2016, doi: 10.1109/MCC.2016.100.

Dash, S.K., Sahoo, J.P., Mohapatra, S., Pati, S.P. (2012). Sensor-Cloud: Assimilation of Wireless Sensor Network and the Cloud. In: Meghanathan N., Chaki N., Nagamalai D. (eds) Advances in Computer Science and Information Technology. Networks and Communications. CCSIT 2012. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 84. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-27299-8_48

Debauche, O., Mahmoudi, S., Andriamandroso, A.L.H., Manneback, P., Bindelle, J., Lebeau, F. (2018). Cloud services integration for farm animals' behavior studies based on smartphones as activity sensors. J Ambient Intell Human Comput 10, 4651–4662 (2019). https://doi.org/10.1007/s12652-018-0845-9

Dias, G.M., Margi, C.B., de Oliveira, F.C.P., Bellalta, B. (2018). "Cloud-Empowered, Self-Managing Wireless Sensor Networks: Interconnecting Management Operations at the Application Layer," in IEEE Consumer Electronics Magazine, vol. 8, no. 1, pp. 55-60, Jan. 2019, doi: 10.1109/MCE.2018.2868110.

Ellison, M., Calinescu, R., Paige, R.F. (2018). Evaluating cloud database migration options using workload models. J Cloud Comp 7, 6 (2018). https://doi.org/10.1186/s13677-018-0108-5

El-Gazzar, R.F. (2014). A Literature Review on Cloud Computing Adoption Issues in Enterprises. In: Bergvall-Kåreborn B., Nielsen P.A. (eds) Creating Value for All Through IT. TDIT 2014. IFIP Advances in Information and Communication Technology, vol 429. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-43459-8_14

Fehling C., Leymann, F., Mietzner, R., Schupeck, W. (2011). A collection of patterns for cloud types, cloud service models, and cloud-based application architectures. University of Stuttgart, Faculty of Computer Science, Electrical Engineering, and Information Technology, Germany, University of Stuttgart, Institute of Architecture of Application Systems, Technical Report Computer Science, 2011, 5: 19.

Ferretti, L., Pierazzi, F., Colajanni, M., Marchetti, M. (2014). "Scalable Architecture for Multi-User Encrypted SQL Operations on Cloud Database Services," in IEEE Transactions on Cloud Computing, vol. 2, no. 4, pp. 448-458, 1 Oct.-Dec. 2014, doi: 10.1109/TCC.2014.2378782.

Feuerlicht, G. & Pokorný, J. (2013). Can Relational DBMS Scale Up to the Cloud?. In: Pooley R., Coady J., Schneider C., Linger H., Barry C., Lang M. (eds) Information Systems Development. Springer, New York, NY. https://doi.org/10.1007/978-1-4614-4951-5_26

Fielding, R. (2000). Architectural Styles and the Design of Network-based Software Architectures. Dissertation. [verkkosivu] [viitattu 2020-09-24]. Saatavilla: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Fortino, G., Di Fatta, G., Pathan, M., Vasilakos, A.V. (2014). Cloud-assisted body area networks: state-of-the-art and future challenges. Wireless Netw 20, 1925–1938. https://doi.org/10.1007/s11276-014-0714-1

Freet, D., Agrawal, R., John, S., Walker, J.J. (2015). Cloud forensics challenges from a service model standpoint: IaaS, PaaS and SaaS. In Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems (MEDES '15). Association for Computing Machinery, New York, NY, USA, 148–155. DOI:https://doi.org/10.1145/2857218.2857253

Gibson, J., Rondeau, R., Eveleigh, D., Tan, Q. (2012). "Benefits and challenges of three cloud computing service models," 2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN), Sao Carlos, 2012, pp. 198-205, doi: 10.1109/CASoN.2012.6412402.

Gonidis, F., Simons, A.J.H., Paraskakis, I., Kourtesis, D. (2013). Cloud application portability: an initial view. In Proceedings of the 6th Balkan Conference in Informatics (BCI '13). Association for Computing Machinery, New York, NY, USA, 275–282. DOI:https://doi.org/10.1145/2490257.2490290

Grothe, M., van den Broecke, J., Carton, L.J. ; Volten, H., Kieboom, R. (2016). Smart Emission - Building a Spatial Data Infrastructure for an Environmental Citizen Sensor Network. Ceur Workshop Proceedings, (2016) Jirka, S.; Stasch, C.; Hitchcock, A. (ed.),

Proceedings of the Geospatial Sensor Webs Conference 2016 (GSW 2016), Münster, Germany, August 29 - 31, 2016., pp. 1-7.

Hammes, D., Medero, H., Mitchell, H. (2014). Comparison of NoSQL and SQL Databases in the Cloud. Proceedings of the Southern Association for Information Systems (SAIS), Macon, GA, 21-22.

Hassan, M.M., Song, B., Huh, E.-N. (2009). A framework of sensor-cloud integration opportunities and challenges. In Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication (ICUIMC '09). Association for Computing Machinery, New York, NY, USA, 618–626. DOI:https://doi.org/10.1145/1516241.1516350

Helmi, A.M., Farhan, M.S., Nasr, M.M. (2018). A framework for integrating geospatial information systems and hybrid cloud computing. Computers & Electrical Engineering, Volume 67, April 2018, Pages 145-158. https://doi.org/10.1016/j.compeleceng.2018.03.027

Hevner, A. (2007). A three-cycle view of design science research. Scandinavian Journal of Information Systems 19 (2), pp. 87–92.

Hevner, A., & Chatterjee, S. (2010). Introduction to design science research. Design Research in Information Systems (pp. 1-8). Springer, Boston, MA.

Hou, L., Zhao, S., Xiong, X., Zheng, K., Chatzimisios, P., Hossain, M.S., Xiang, W. (2016). "Internet of Things Cloud: Architecture and Implementation," in IEEE Communications Magazine, vol. 54, no. 12, pp. 32-39, December 2016, doi: 10.1109/MCOM.2016.1600398CM.

Hromic, H., Phuoc, D.L., Serrano, M., Antonić, A., Žarko, I.P., Hayes, C., Decker, S. (2015). "Real time analysis of sensor data for the Internet of Things by means of clustering and event processing," 2015 IEEE International Conference on Communications (ICC), London, 2015, pp. 685-691, doi: 10.1109/ICC.2015.7248401.

Hummen, R. Henze, M., Catrein, D., Wehrle, K. (2012). "A Cloud design for user-controlled storage and processing of sensor data," 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, Taipei, 2012, pp. 232-240, doi: 10.1109/CloudCom.2012.6427523.

IBM Cloud Education. (2019). DBaaS (Database-as-a-Service). [verkkosivu] [viitattu 2020-07-30]. Saatavilla:
https://www.ibm.com/cloud/learn/dbaas

Ji, Z., Ganchev, I., O'Droma, M., Zhao, L., Zhang, X. (2014). A Cloud-Based Car Parking Middleware for IoT-Based Smart Cities: Design and Implementation. Sensors 2014, 14(12), 22372-22393; https://doi.org/10.3390/s141222372

Kiefer, T. & Lehner, W. (2011). "Private Table Database Virtualization for DBaaS," 2011 Fourth IEEE International Conference on Utility and Cloud Computing, Victoria, NSW, 2011, pp. 328-329, doi: 10.1109/UCC.2011.52.

Kulkarni, G., Waghmare, R., Palwe, R., Waykule, V., Bankar, H., Koli, K. (2012). "Cloud storage architecture," 2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA), Bali, 2012, pp. 76-81, doi: 10.1109/TSSA.2012.6366026.

Maenhaut, P.-J., Moens, H., Verheye, M., Verhoeve, P., Walraven, S., Truyen, E., Joosen, W., Ongenae, V., De Turck, F. (2013). "Migrating medical communications software to a multi-tenant cloud environment," 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, 2013, pp. 900-903.

Maenhaut, P.-J., Moens, H., Ongenae, V., De Turck, F. (2015a). "Design and evaluation of a hierarchical multi-tenant data management framework for cloud applications," 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, 2015, pp. 1208-1213, doi: 10.1109/INM.2015.7140468.

Maenhaut, P.-J., Moens, H., Volckaert, B., Ongenae, V., De Turck, F. (2015b). "Design of a hierarchical software-defined storage system for data-intensive multi-tenant cloud applications," 2015 11th International Conference on Network and Service Management (CNSM), Barcelona, 2015, pp. 22-28, doi: 10.1109/CNSM.2015.7367334.

Majumdar, S., Asif, M., Melendez, J.O., Kanagasundaram, R., Lau, D.T., Nandy, B., Zaman, M., Srivastava, P., Goel, N. (2012). Middleware architecture for sensor-based bridge infrastructure management. In Proceedings of the 15th Communications and Networking Simulation Symposium (CNS '12). Society for Computer Simulation International, San Diego, CA, USA, Article 8, 1–10.

Mell, P., Grance, T. (2011). The NIST Definition of Cloud Computing, NIST Special Publication 800-145, September 2011.

Microsoft. What is a virtual machine? [verkkosivu] [viitattu 2020-07-16]. Saatavilla: https://azure.microsoft.com/en-us/overview/what-is-a-virtual-machine/

Modak, A., Chaudhary, S. D., Paygude, P.S., Ldate, S.R. (2018). "Techniques to Secure Data on Cloud: Docker Swarm or Kubernetes?," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, 2018, pp. 7-12, doi: 10.1109/ICICCT.2018.8473104.

MQTT.org. MQTT: The Standard for IoT Messaging. [verkkosivu] [viitattu 2020-09-24]. Saatavilla:
https://mqtt.org/

Nadgowda, S., Suneja, S., Bila, N., Isci, C. (2017). "Voyager: Complete Container State Migration," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, 2017, pp. 2137-2142, doi: 10.1109/ICDCS.2017.91.

Nayak, A., Poriya, A., Poojary, D. (2013). Type of NOSQL databases and its comparison with relational databases. International Journal of Applied Information Systems, 5(4), 16-19.

Oracle. A Relational Database Overview. [verkkosivu] [viitattu 2020-07-16]. Saatavilla: https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html

Peinl, R., Holzschuher, F., Pfitzer, F. (2016). Docker Cluster Management for the Cloud - Survey Results and Own Solution. J Grid Computing 14, 265–282 (2016). https://doi.org/10.1007/s10723-016-9366-y

Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M. (2008). Systematic mapping studies in software engineering. 12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12 (pp. 1-10).

Phan, T.A.M, Nurminen, J.K., Di Francesco, M. (2015). "Cloud Databases for Internet-of-Things Data," 2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom), Taipei, 2014, pp. 117-124, doi: 10.1109/iThings.2014.26.

Rajkumar, R. & Iyengar, N.C.S.N (2013). Dynamic Integration of Mobile JXTA with Cloud Computing for Emergency Rural Public Health Care. Osong Public Health and Research Perspectives. Volume 4, Issue 5, October 2013, Pages 255-264. https://doi.org/10.1016/j.phrp.2013.09.004

Ray, P.P. (2016). A survey of IoT cloud platforms. Future Computing and Informatics Journal. Volume 1, Issues 1–2, December 2016, Pages 35-46. https://doi.org/10.1016/j.fcij.2017.02.001

Sakr, S. (2013). Cloud-hosted databases: technologies, challenges and opportunities. Cluster Comput 17, 487–502 (2014). https://doi.org/10.1007/s10586-013-0290-7

Saunders, C.S., Liu, G., Yu, Y., Zhu, W. (2016). “Data-driven distributed analytics and control platform for smart grid situational awareness,” CSEE Journal of Power and Energy Systems, vol. 2, no. 3, pp. 51-58, Sept. 2016, doi: 10.17775/CSEEJPES.2016.00035.

Shah, J. & Dubaria, D. (2019). "Building Modern Clouds: Using Docker, Kubernetes & Google Cloud Platform," 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2019, pp. 0184-0189, doi: 10.1109/CCWC.2019.8666479.

Slominski, A., Muthusamy, V., Khalaf, R. (2015). "Building a Multi-tenant Cloud Service from Legacy Code with Docker Containers," 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, 2015, pp. 394-396, doi: 10.1109/IC2E.2015.66.

Stanton, N.A., Chambers, P.R.G., Piggott, J. (2001). Situational awareness and safety. Safety Science, Volume 39, Issue 3, December 2001, Pages 189-204. https://doi.org/10.1016/S0925-7535(01)00010-8

Strauch, S., Andrikopoulos, V., Bachmann, T., Karastoyanova, D., Passow, S., Vukojevic-Haupt, K. (2013a). "Decision Support for the Migration of the Application Database Layer to the Cloud," 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Bristol, 2013, pp. 639-646, doi: 10.1109/CloudCom.2013.90.

Strauch, S., Andrikopoulos, V., Bachmann, T., Leymann, F. (2013b). Migrating Application Data to the Cloud using Cloud Data Patterns. In Closer (pp. 36-46).

Strauch, S., Andrikopoulos, V., Karastoyanova, D., Leymann, F., Nachev, N., Stäbler, A. (2014). Migrating enterprise applications to the cloud: methodology and evaluation. International Journal of Big Data Intelligence, 1(3), pp. 127–140. https://doi.org/10.1504/IJBDI.2014.066319

Su, J., Huang Y., Lv, G., Liu, H., Jin, P. (2016). "A Framework Research of Power Grid Knowledge Recommendation and Situation Reasoning Based on Cloud Computing and

CEP," 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud), Beijing, 2016, pp. 79-83, doi: 10.1109/CSCloud.2016.14.

Suciu, G., Ularu, E.G., Craciunescu, R. (2012). "Public versus private cloud adoption — A case study based on open source cloud platforms," 2012 20th Telecommunications Forum (TELFOR), Belgrade, 2012, pp. 494-497, doi: 10.1109/TELFOR.2012.6419255.

Suciu, G., Vulpe, A., Halunga, S., Fratu, O., Todoran, G., Suciu, V. (2013). "Smart Cities Built on Resilient Cloud Computing and Secure Internet of Things," 2013 19th International Conference on Control Systems and Computer Science, Bucharest, 2013, pp. 513-518, doi: 10.1109/CSCS.2013.58.

Sun, J., Aida, K., Tanjo, T. (2017). Architecture-Independent Cloud Computation for Sensor Environment and Its Applications. In: Fortino G., Ali A., Pathan M., Guerrieri A., Di Fatta G. (eds) Internet and Distributed Computing Systems. IDCS 2017. Lecture Notes in Computer Science, vol 10794. Springer, Cham. https://doi.org/10.1007/978-3-319-97795-9_3

Tran, V., Keung, J., Liu, A., Fekete, A. (2011). Application migration to cloud: a taxonomy of critical factors. In Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing (SECLOUD '11). Association for Computing Machinery, New York, NY, USA, 22–28. DOI:https://doi.org/10.1145/1985500.1985505

Van der Veen, J.S., van der Waaij, B., Meijer, R.J. (2012). "Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual," 2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, 2012, pp. 431-438, doi: 10.1109/CLOUD.2012.18.

Varghese, B. & Buyya, R. (2018). Next generation cloud computing: New trends and research directions. Future Generation Computer Systems, Volume 79, Part 3, February 2018, Pages 849-861. https://doi.org/10.1016/j.future.2017.09.020

Vodomin, G. & Andročec, D. (2015). Problems during database migration to the cloud. In Central European Conference on Information and Intelligent Systems (p. 11). Faculty of Organization and Informatics Varazdin.

Wang, J., Chen, X., Li, J., Zhao, J., Shen, J. (2017). Towards achieving flexible and verifiable search for outsourced database in cloud computing. Future Generation Computer Systems, Volume 67, February 2017, Pages 266-275. https://doi.org/10.1016/j.future.2016.05.002

Wu, C., Buyya R., Ramamohanarao, K. (2019). Cloud Pricing Models: Taxonomy, Survey, and Interdisciplinary Challenges. ACM Comput. Surv. 52, 6, Article 108 (January 2020), 36 pages. DOI:https://doi.org/10.1145/3342103

Yu, T., Qiu, J., Reinwald, B., Zhi, L., Wang, Q., Wang, N. (2012). "Intelligent Database Placement in Cloud Environment," 2012 IEEE 19th International Conference on Web Services, Honolulu, HI, 2012, pp. 544-551, doi: 10.1109/ICWS.2012.74.

Zhao, L., Sakr, S., Fekete, A., Wada, H., Liu, A. (2012). "Application-Managed Database Replication on Virtualized Cloud Environments," 2012 IEEE 28th International Conference on Data Engineering Workshops, Arlington, VA, 2012, pp. 127-134, doi: 10.1109/ICDEW.2012.77.

Zou, C., Deng, H., Qiu, Q. (2013). "Design and Implementation of Hybrid Cloud Computing Architecture Based on Cloud Bus," 2013 IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks, Dalian, 2013, pp. 289-293, doi: 10.1109/MSN.2013.72.