Tampereen yliopisto

Paavo Natri

TEOLLISTEN JÄRJESTELMIEN JA PILVI-POHJAISTEN SOVELLUSTEN VÄLISET INTEGRAATIOMENETELMÄT

Tekniikan ja luonnontieteiden tiedekunta
Diplomityö
Toukokuu 2019

TIIVISTELMÄ

Paavo Natri: Teollisten järjestelmien ja pilvipohjaisten sovellusten väliset integraatiomenetelmät
Diplomityö
Tampereen yliopisto
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Toukokuu 2019

Pilvipohjainen järjestelmäarkkitehtuuri, jossa laskenta- ja tallennusresursseja tarjotaan internetin yli abstrakteina palveluina, on jo pitkälti valtavirtaa tyypillisissä IT-maailman sovelluksissa. Pilvipohjaisuudesta haetaan esimerkiksi kustannussäästöjä ja joustavuutta eri tyyppisten sovellusten toimintaan, sekä siirretään vastuuta IT-infrastruktuurin ylläpidosta kolmannelle osapuolelle.

Teollisen internetin eri mallit tähtäävät teollisen toiminnan tehostamiseen digitalisaation menetelmin. Tehdasjärjestelmien tuottaman raakadatan jalostaminen hyödylliseksi informaatioksi on teollisen internetin ydinajattelua. Pilvimallisten palveluiden rooli teollisen internetin järjestelmissä on tarjota teollisen mittakaavan vaatimia suuria laskenta- ja talletusresursseja, toimia esimerkiksi koneoppimiseen ja raportointiin keskittyvien sovellusten julkaisualustana, sekä tuoda läpinäkyvyyttä ja vertailtavuutta yrityksen eri tuotantolaitosten välille.

Tässä työssä pyritään löytämään hyviä suunnitteluperiaatteita teollisten järjestelmien ja pilvipohjaisten sovellusten integraatioihin. Aihetta tarkastellaan järjestelmäarkkitehtuurin ja integraatioihin kohdistuvien vaatimusten tasolla, mutta taustalle rakennetaan myös käsitys olennaisimmista menetelmistä ja tekniikoista, jotka toimivat teollisen internetin integraatioiden rakennuspalikoina.

Työn ensimmäinen vaihe on teoriaosuus, jossa selvitetään mistä osista ja teknologioista teollisen internetin järjestelmä koostuu, ja miten teolliset järjestelmät eroavat IT-ympäristöistä. Teoriaosuudessa tehdään myös katsaus teollisen internetin periaatteellisiin arkkitehtuurimalleihin. Työn toisen osan muodostavat kohdeyrityksen toteuttamista integraatioista poimitut case-arkkitehtuurit, joista jokaisessa esitetään teollisen järjestelmän ja pilvipohjaisen sovelluksen välinen integraatio toteutettuna eri lähestymistavoilla. Työn viimeisessä vaiheessa luokitellaan eri näkökulmia, joiden pohjalta integraatiototeutuksia voidaan arvioida, arvioidaan case-arkkitehtuurit niiden avulla ja listataan työn aikana havaitut integraatioihin kohdistuvat vaatimukset. Vaatimusten pohjalta hahmotellaan arkkitehtuurimalli, jonka katsotaan kokoavan useita työn aikana hyväksi todettuja suunnitteluperiaatteita yhteen. Malli vastaa erityisesti yleiskäyttöisyyden ja siiloutumisen sekä ylläpidettävyyden ja tietoturvan asettamiin haasteisiin.

Avainsanat: Teollinen internet, pilvi, integraatiot, järjestelmäarkkitehtuuri

ABSTRACT

Paavo Natri: Integration methods between industrial systems and cloud-based applications.
Master of Science Thesis
Tampere University
Master's Degree Programme in Automation Engineering
May 2019

Cloud-based system architecture, where storing and processing of data is delivered via internet as abstract services, has already reached mainstream status in typical IT-applications. Motivation for using cloud as application publishing model is for example saving in operational costs, gaining flexibility for different application deployments and to externalize the administration responsibilities to a 3rd party.

The various models of industrial internet aim to streamline industrial production and environments by means of digitalization. Processing raw measurement data gathered from factory systems into meaningful and usable information is part of the core of industrial internet. The role of cloud services in industrial internet solutions is to offer data processing and storage in industrial scale, act as a publishing platform for applications focusing for example on machine learning and reporting, and to bring visibility and comparability throughout the company.

The aim of this thesis is to find good design principles for building data integrations between industrial systems and cloud-based applications. The topic is reviewed in the level of system architecture and requirements, but also the essential technologies acting as building blocks of integrations are studied.

The first section of study is the theory part, where the aim is to find out of which components and technologies the industrial internet systems are formed of, and how they differ from common IT-environments. Also, in theory part a review of key architectural models of industrial internet is performed. The second section of study consists of case-architectures, which are selected from actual integrations implemented by the target company of this thesis. Each case-architecture describes a differently implemented integration between an industrial system and a cloud-based application. In the last section different perspectives for reviewing integration architectures are categorized, and the system requirements concerning the integrations based on the outcomes of previous study sections are listed. Based on detected requirements a general-purpose architectural model for building integrations is also drafted in the last section. The model combines the good design principles discovered during the study and acts as the result of it. The main challenges to which the model takes a stance on are general usability across different systems, avoiding overhead in horizontal integrations, maintainability and security.

Keywords: Industrial internet, cloud computing, integrations, system architecture

ALKUSANAT

Tämä diplomityö syntyi vaiheittain alkuvuoden 2018 ja loppukevään 2019 välillä. Alkuperäinen suunniteltu aikataulu joutui ehkä hieman venymään, ja kerkesipä yliopiston nimikin vaihtua työn kirjoittamisen aikana. Prosessi oli kuitenkin palkitseva, eivätkä pienet mietintätauot uskoakseni ainakaan huonontaneet lopputulosta.

Haluan tässä esittää kiitokseni Novotek Oy:n työporukalle, sekä erityisesti tätä diplomityötä ohjanneelle Jukka Piriselle. Novotekilla minulla on ollut mahdollisuus päästä suunnittelemaan ja toteuttamaan tässäkin työssä esiin tulevia järjestelmiä kokonaisvaltaisesti, saaden näin hyvän kuvan alan nykyisestä tilasta.

Toiseksi haluan kiittää assistant professor David Hästbackaa sekä professori Hannu Koivistoa asiantuntevista kommenteista tutkimuksen rakennetta ja sisältöä koskien, teistä oli paljon apua ajatusteni kytkemisessä alaan yleisemmällä tasolla ja huomioimaan myös tulevaisuuden kehityssuuntia.

Kolmantena kiitän vielä läheisiäni ja ystäviäni tuesta sekä kannustuksesta työn aikana. Toisinaan pieni eteenpäin potkiminenkin oli varmasti paikallaan.

Espoossa, 17.5.2019

Paavo Natri

SISÄLLYSLUETTELO

KUVALUETTELO

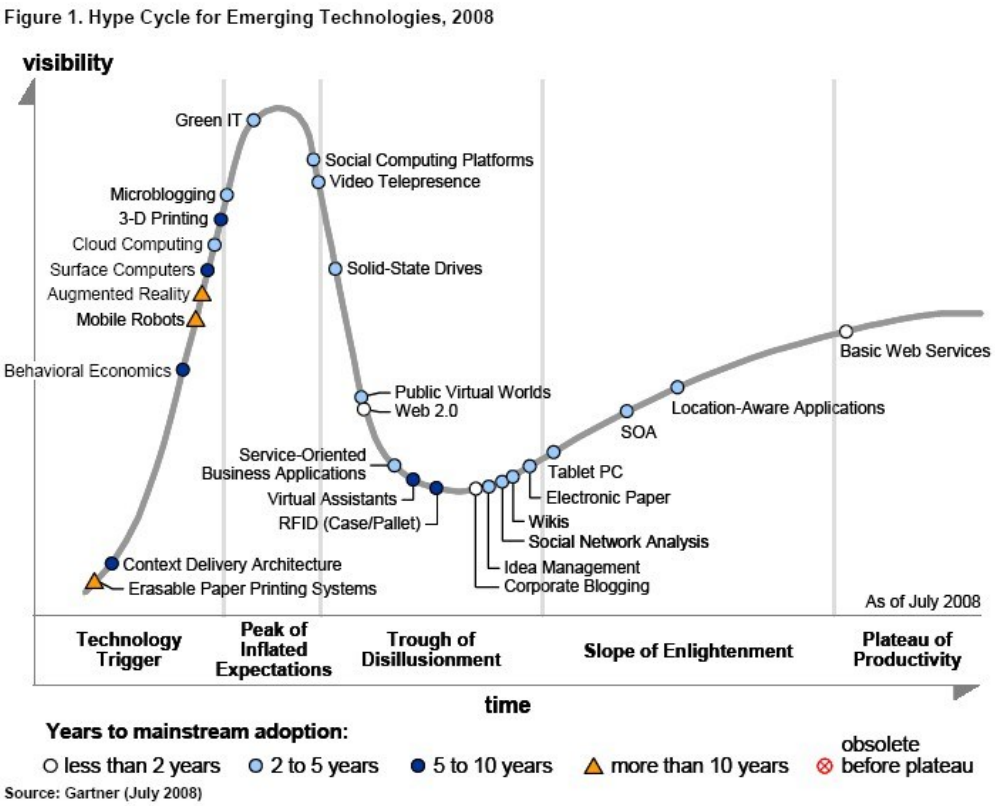
LYHENTEET JA MERKINNÄT

AD	Active Directory, Microsoft-pohjainen hakemistopalvelu.
API	Application programming interface, ohjelmointirajapinta, jonka kautta kohdeohjelmistoa voidaan käyttää tekstipohjaisesti.
Big Data	Kattotermi teknologioille, joilla pyritään ratkaisemaan suuriin, jäsentelemättömiin data-aineistoihin liittyviä ongelmia.
BI	Business Intelligence, liiketoiminnan tuottaman tiedon analysointi ja tulosten hyödyntäminen päätöksenteossa.
DCS	Distributed Control System, hajautettu automaatio- ja ohjausjärjestelmä.
DMZ	Demilitarized Zone, eteisverkko, joka eristää yrityksen sisäiset verkot internetistä.
ERP	Enterprise Resource Planning, toiminnanohjausjärjestelmä.
Ethernet	OSI-mallin kerrokset 1 ja 2 toteuttava verkkotekniikka, de-facto standardi langallisten lähiverkkojen toteuttamiseen.
ESB	Enterprice Service Bus, yrityksen palvelu- ja integraatioväylä.
HTTP	Hypertext Transfer Protocol, webin pohjana toimiva tekstipohjainen tiedonsiirtoprotokolla.
Middleware	Käyttöjärjestelmän ja lopullisen sovelluksen välissä toimiva ohjelmistokerros, joka tarjoaa rajapinnan ja palveluita näiden tasojen välille.
M2M	Machine-to-Machine. Laitteiden välinen ohjelmallinen kommunikointi.
MES	Manufacturing Execution System, tuotannonohjausjärjestelmä.
NoSQL	Not Only SQL, Non-SQL, suurten jäsentelemättömien data-aineistojen varastointiin soveltuva tietokanta, joka ei noudata relaatiomallia.
OSI-malli	Open Systems Interconnection Reference Model, referenssimalli sovelluksen kommunikaatiopinon esittämiseen fyysiseltä tasolta sovellustasolle.
PID	Proportional Integral Derivative, säätötekninen perusalgoritmi.
PLC	Programmable Logic Controller, ohjelmoitava logiikka.
REST	Representational State Transfer, tilaton malli ohjelmointirajapinnan rakentamiseen.
SCADA	Supervisory Control And Data Acquisition, prosessia valvovat ja ohjaavat sovellukset.
TLS	Transport Layer Security, yleisesti käytetty TCP-yhteyksien salausmenetelmä.
URL	Uniform Resource Locator, viittaus verkossa sijaitsevaan resurssiin.
VPN	Virtual Private Network, tyypillisesti julkista internetiä siirtotienään käyttävä virtuaalinen yksityinen verkkoyhteys, jolla voidaan liittää verkkoja toisiinsa riippumatta niiden maantieteellisestä sijainnista.
WLAN	Wireless Local Area Network, langaton lähiverkkotekniikka.

1. JOHDANTO

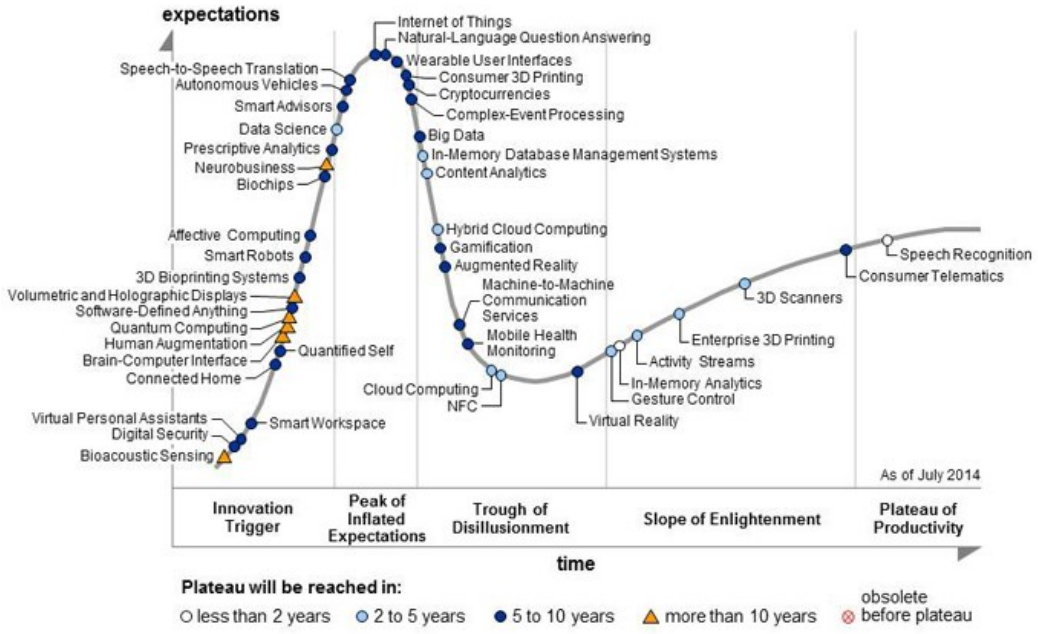
Pilvimalli, eli laskenta- ja tallennusresurssien tarjoaminen abstrakteina palveluina internetin välityksellä, on viimeisen vuosikymmenen aikana vakiinnuttanut asemansa verkottuneiden IT-sovellusten ja -palveluiden tärkeänä julkaisu- ja kehitysalustana. Muutoksen myötä yhä useampi yritys on kustannussäästöjen, ja pilvimallin joustavuuden tuomien etujen toivossa luopunut omista datakeskuksistaan ja keskittänyt IT-resurssiensa hallinnan ulkoiselle pilvitarjoajalle. Alan siirtyminen pilvimalliin on mahdollistanut, ja tulee myös jatkossa mahdollistamaan täysin uudentyyppisten teknologioiden ja liiketoimintamallien kehittymisen pilviteknologian päälle.

Tutkimuslaitos Gartnerin uusien teknologioiden kypsyyttä kuvaavalla hypekäyrällä vuodelta 2008 pilviteknologiat on merkitty jyrkän nousun vaiheeseen. Jyrkässä nousussa oleva tuore teknologia saa paljon medianäkyvyyttä ja siihen kohdistetaan suuria odotuksia, mutta käytännön ratkaisumalleja tai valmiita tuotteita ei juurikaan ole tarjolla. Hypevaiheen teknologiasta ei myöskään tyypillisesti ole yleistä käsitystä mitkä uuden teknologian lupauksista ovat todellisuudessa lunastettavissa ja siten realistisia.



Kuva 1: Gartner-hypekäyrä heinäkuu 2008

Vastaava kaavio kuusi vuotta myöhemmin kesäkuussa 2014 näyttää pilvimallin jo ohittaneen ylittyneiden odotusten huipun, ja etenevän kohti tuottavuuden tasankoa, joka viittaa tekniikan kypsymiseen ja tuotantovalmiiden ratkaisujen saapumiseen markkinoille.



Kuva 2: Gartner-hypekäyrä kesäkuu 2014

Kehitys ei myöskään pysähtynyt vuoteen 2014, vaan Gartnerin yhteenveto vuodelta 2018 [1] toteaa pilvimallin siirtyneen ”Slope of Enlightenment”-vaiheeseen, joka viittaa teknologian tarjoamien käytännön hyötyjen konkretisoitumiseen ja kypsien toisen tai kolmannen sukupolven tuotteiden saapumiseen markkinoille. Pilvimallin kypsyminen mahdollistaa myös uusien, pilvimallin itsensä päälle rakentuvien teknologioiden kehittymisen. Eräs kypsän pilvimallin varaan nojautuva uusi teknologiakokonaisuus on teollinen internet.

Teollinen internet on yläkäsite useille eri lähestymistavoille digitalisaation ja tiedon hyödyntämiseen teollisuusympäristöissä tuotannon tehostamiseksi. Käytännössä teollinen internetin sovellus sisältää tyypillisesti tuotanto- ja automaatiojärjestelmien tuottaman datan keruuta ja talletusta, sekä sen käsittelyä laskennallisesti tekoälyalgoritmeja hyödyntäen. Teolliseen internetiin liittyviä käsitteitä ja konsepteja ovat esimerkiksi teollinen esineiden internet (IIoT, *industrial internet of things*) ja Industrie 4.0.

Teolliset järjestelmät tuottavat suuria määriä dataa, ja siten myös sen käsittelyyn tarvitaan paljon resursseja. Perinteisillä paikallisilla konesalipalveluilla tarvittavan kokoluokan järjestelmän rakentaminen vaatisi merkittäviä taloudellisia panostuksia, ja tarpeeksi joustavan järjestelmän rakentaminen olisi haastavaa. Pilvimallin avulla resursseja voidaan hankkia järjestelmän käyttöön tarpeen mukaan, ja se voidaan rakentaa yhdistelmänä eri tasoisia palveluita. Lisäksi järjestelmä voidaan pystyttää nopeasti ja ilman merkittäviä

kertainvestointeja. Pilvimallin rooli teollisen internetin järjestelmissä on siis tarjota talletus- ja laskentaresursseja, sekä valmiita datankäsittelypalveluita.

Automaatio- ja tuotantojärjestelmien kehitys on perinteisesti ollut konservatiivisempaa kuin IT-teknologioiden, ja eri järjestelmien ja tekniikoiden elinkaaret ovat tuotantoympäristöissä pitkiä. Teollisuusympäristöissä esiintyy myös paljon valmistajakohtaisia suljettuja protokollia, eikä rajapintojen avoimuus ole itseisarvoista. Sulkeutuneisuus ja eri ympäristöjen siiloutuminen aiheuttavat haasteita järjestelmien integroinnissa pilvimallia hyödyntäviin teollisen internetin sovelluksiin. Teollisen internetin sovellusten on myös kyettävä vakuuttamaan teolliset toimijat siitä, etteivät sovellusten vaatimat rajapinnat aiheuta turvallisuusriskejä tai toimintahäiriöitä kriittisissä tuotantojärjestelmissä, ja että sovelluksista saatavat hyödyt ovat tarpeeksi merkittäviä, jotta ympäristöjä kannattaa lähteä avaamaan ulkomaailmaan.

1.1 Ongelman asettelu ja tutkimuskysymykset

Työssä pyritään saamaan käsitys, miten jo olemassa olevien kappaletavara- ja prosessiautomaatiojärjestelmien tuottaman aikasarjatiedon keruuta ja siirtoa voidaan kehittää pilviympäristöön nojautuvien teollisen internetin sovellusten kannalta. Työssä nostetaan esiin joitakin kohdeyrityksen toteuttamia tiedonkeruusovelluksia, joiden avulla pyritään saamaan kuvaa mitkä ovat reunaehtoja teollisen internetin järjestelmän tiedonsiirron toteutukselle.

Pilviympäristöllä tarkoitetaan työssä pääasiassa julkisia pilvipalveluita (*public cloud*). Koska tuotantojärjestelmät sisältävät usein useilta eri aikakausilta periytyviä teknologioita ja niiden uusiutumissykli on hidas, ovat toteutetut käytännön arkkitehtuurit tyypillisesti enemmän tai vähemmän hybridimallisia (*hybrid cloud*), joissa osa toteutuksesta on fyysisesti tuotantolaitoksella ja osa julkisessa pilvessä.

Tutkimuksen pohjalle asetetaan seuraava tutkimuskysymys, johon työn aikana pyritään löytämään vastaus:

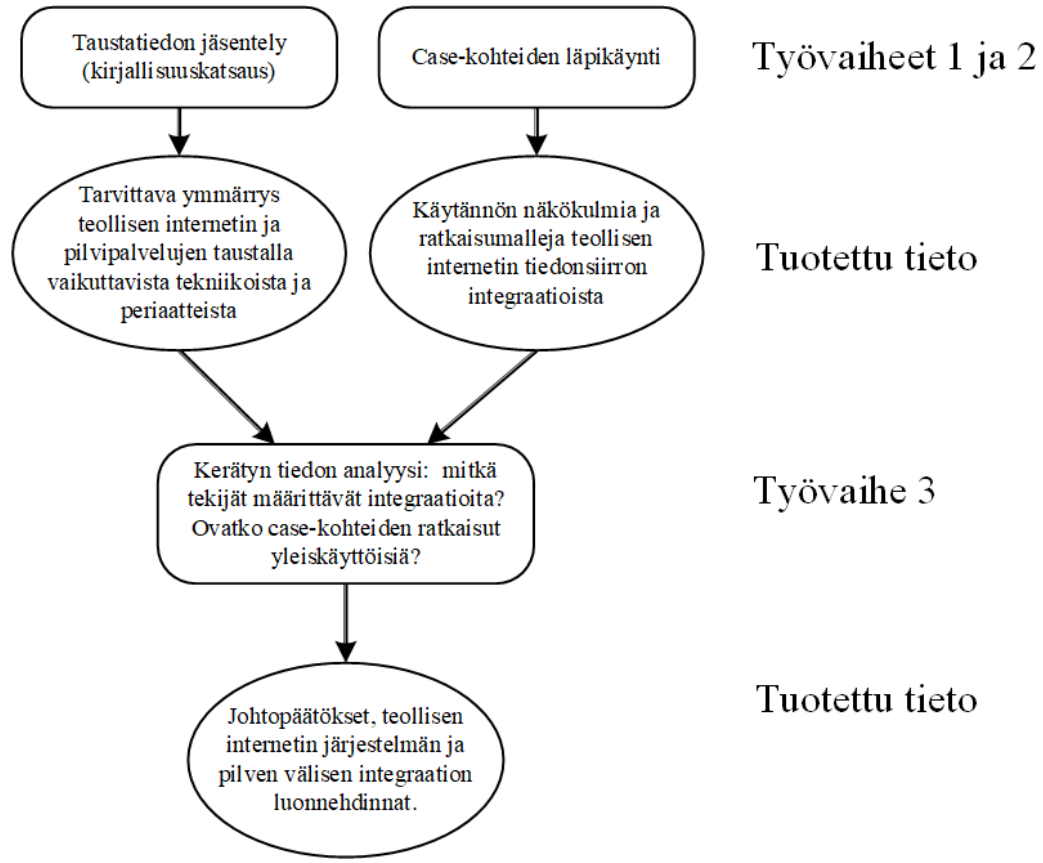
1. Mitkä ovat hyviä suunnitteluperiaatteita jo olemassa olevien tuotantojärjestelmien tiedonkeruun integroinnissa osaksi pilvipohjaisen teollisen internetin sovelluksia?

Päätutkimuskysymystä täsmennetään ja työn sisältöä tarkennetaan seuraavilla apukysymyksillä:

2. Mitkä ovat automaatio- ja tehdasjärjestelmien, sekä tavallisen IT-tekniikan toisistaan erottavia tekijöitä?
3. Millaisia teollisen internetin malleja on kehitetty, ja miten ne sopivat yhteen ns. legacy-järjestelmien kanssa?
4. Millaisia käytännön haasteita ja teknisiä ratkaisuja paikallisten järjestelmien ja pilvessä sijaitsevien sovellusten väliseen kommunikaatioon liittyy?

1.2 Tutkimussuunnitelma ja työn rakenne

Työn pohjalle esitettyihin tutkimusongelmiin pyritään etsimään vastauksia perustuen sekä kirjallisuustietoon, että käytännön toteutetuista järjestelmäarkkitehtuurimalleista muodostettavien johtopäätösten pohjalta. Tutkimuksen karkea rakenne esitetään kuvassa 3:



Kuva 3: Tutkimuksen rakenne

Kuva 3 esittämä työvaihe 1, eli taustatiedon jäsentely kirjallisuuslähteiden pohjalta sisällytetään kappaleisiin 2 ja 3. Näissä kappaleissa rakennetaan työn teoreettinen runko, jonka avulla pyritään saamaan käsitys teollisen internetin ja pilvipalveluiden taustalla vaikuttavista tekniikoista ja periaatteista. Samalla saadaan käsitys mitkä ovat näiden ympäristöjen erot, ja mitä haasteita niiden välisiin integraatioihin liittyy.

Työvaihe 2 eli katsaus case-kohteisiin suoritetaan kappaleessa 4, jossa asiakasprojekteina toteutetut, case-kohteiksi työhön valitut arkkitehtuurit käydään läpi tarvittavalla tarkkuudella, jotta saadaan kunkin kohteen tyypilliset piirteet kuvattua ja muodostettua käsitys mitkä ovat kyseisen kohteen ja sen sovellusarkkitehtuurin edut ja haasteet.

Työn kolmannessa vaiheessa kappaleessa 5 pyritään muodostamaan käsitys siitä mistä näkökulmista integraatioratkaisua voidaan arvioida, ja muodostetaan suunnittelumalli in-

tegraatioiden toteuttamiselle hyödyntäen aiempien kappaleiden lopputulemia ja tunnistettuja arviointinäkökulmia. Suunnittelumallin avulla pyritään myös vastaamaan työn tutkimusongelmaan.

2. TEOLLISEN INTERNETIN RAKENNE

Teollinen internet (*Industrial internet*) on yläkäsite, jonka voidaan katsoa kattavan useita eri konsepteja ja lähestymistapoja tuotannon digitalisaatioon ja tehostamiseen. Esimerkiksi termit IIoT (*Industrial Internet of Things*), Digital factory, Smart manufacturing ja Industry 4.0 käsittelevät keskenään hyvin samankaltaisia aihealueita, mutta toteutustavat tai tekniikat vaihtelevat esimerkiksi kaupallisten näkökulmien mukaan. Ensimmäisten joukossa määritelmän teolliselle internetille tarjosi General Electric vuonna 2012. GE:n määritelmässä teollinen internet jaetaan kolmeen avainelementtiin: älykkäisiin koneisiin, kehittyneeseen analytiikkaan ja järjestelmien parissa työskenteleviin ihmisiin [2]. Teolliseen internetiin liittyy myös IoT:n (*Internet of Things*) käsite, joka esittää fyysisten laitteiden kytkemistä osaksi laitteiden, IT-infrastruktuurin ja ihmisten muodostamaa verkostoa. Teollisen internetin näkökulma on kuitenkin erityisesti teollisuuden alojen, kuten energiantuotannon, valmistuksen, logistiikan ja vastaavien teollisuuden järjestelmien tarpeissa [3], mikä aiheuttaa tiukempia vaatimuksia verkoston toiminnalle kuin mihin pitkälti kuluttaja- tai IT-sovelluksiin suunnatut IoT-ratkaisut välttämättä ottavat kantaa. Kuvassa 4 esitetään kuluttaja-IoT:n ja teollisen internetin ja IIoT:n eroja:

	Consumer IoT	Industrial IoT
Impact	Revolution	Evolution
Service Model	Human-centered	Machine-oriented
Current Status	New devices and standards	Existing devices and standards
Connectivity	Ad-Hoc (infrastructure is not tolerated; nodes can be mobile)	Structured (nodes are fixed; centralized network management)
Criticality	Not stringent (excluding medical applications)	Mission critical (timing, reliability, security, privacy)
Data Volume	Medium to High	High to Very High

Kuva 4: IoT:n ja IIoT:n eroja. [4]

Sissini et. al. mukaan IoT:n tavoitteet painottuvat uusien, edullisten laitteiden kytkemiseen internetiin joustavasti ja käyttäjäystävällisesti, kun taas teollisen IIoT:n tavoitteena on jo olemassa olevien, aiemmin internetistä eristettyjen tehtaiden ja koneiden kytkemisessä internetiin. Näin ollen myös vaatimukset IoT:n ja IIoT:n välillä vaihtelevat esimerkiksi palvelulta vaadittavan luotettavuuden ja turvallisuuden mukaan. IIoT voidaan myös esittää aiempien M2M (*Machine-to-Machine*) -sovellusten kehityksenä, eikä täydellisenä vallankumouksena kuten kuluttajamaailman IoT. [4]

Ongelmallisesti useat keskenään hyvin samankaltaista toiminnallisuutta korkealla tasolla kuvaavat teollisen internetin konseptit ja termit jättävät käytännön toteutukset ja sovellusmahdollisuudet usein hyvin avoimiksi. Esimerkiksi tammikuussa 2015 Yhdysvalloissa tehdyssä tutkimuksessa 87% teollisuuden johtajista myönsi, ettei heillä ollut selvää käsitystä teollisen internetin liiketoimintamalleista tai tekniikoista [3]. Suomessa vuonna 2014 tehdyssä vastaavassa tutkimuksessa jopa kolmannes suomalaisista yritysjohtajista tunnusti teollisen internetin aivan tuntemattomaksi asiaksi ja toinen kolmannes seuraavansa kehitystä vain passiivisesti. [5] Gilchristin mukaan tämä ei kuitenkaan ole yllättävää, sillä teollista internetiä kuvataan usein niin korkealla tasolla, että sitä tukevat teknologiat ja vaadittavat käytännön ratkaisut jäävät kokonaisuuden varjoon. Esimerkiksi mitta- ja toimilaitteiden tuottaman tiedon kerääminen ja sen käyttäminen päätöksenteossa kuten säädössä, on jo pitkään ollut osa tuotannon automaatiojärjestelmien toimintaa. Vastaavasti myöskään M2M-tyyppisessä (*machine-to-machine*) laitteiden välisessä automaattisessa kommunikoinnissa ei itsessään ole teollisuusympäristössä varsinaisesti mitään uutta. Koska juuri tiedon kerääminen ja sen välittäminen automaattisesti analysoitavaksi on teollisen internetin ydintä, on tämä ristiriita omiaan herättämään hämmennystä siitä, mikä on teollisen internetin uutuusarvo ja hyöty perinteisiin toteutuksiin nähden [3].

Teollisen internetin eri konsepteille ja malleille ovat yhteisiä esimerkiksi seuraavat piirteet:

- Tuotantoprosessien tuottaman raakadatan kerääminen ja jatkokäyttö.
- Integraatiot järjestelmien välillä. Pystysuunnassa esimerkiksi automaatiopyramidin eri tasojen pilven välillä ja vaakasuunnassa esimerkiksi eri tuotantolaitosten välillä.
- Data-analytiikan, kuten koneoppimisen hyödyntäminen ja big-data tekniikoiden soveltaminen kerättyyn data-aineistoon. Analyysin tulosten kytkeminen takaisin tuotannon järjestelmiin esimerkiksi ennakoivan kunnossapidon kautta. Älykkyyden tuominen myös järjestelmän alimmille tasoille kuten yksittäisiin antureihin saakka.
- Prosessin tilan reaaliaikainen visualisointi sekä hälytysten ja poikkeustapausten käsittely.
- Pilvimallin hyödyntäminen integraatioiden mahdollistajana sekä laskentaresurssien tarjoajana.

Teollinen internet on siis malli, joka pyrkii integroimaan älyä ja analytiikkaa teollisuuden järjestelmiin. Tavoitteita teollisen internetin hyödyntämiselle voivat olla esimerkiksi:

- Nykyisten manuaalisten tehtävien automatisointi ja optimointi. [6]
- Uusien palveluiden ja liiketoimintamallien kehittyminen ja kehittymisen mahdollistuminen. [6] Järjestelmien ja datan avautuminen mahdollistaa uusien toimijoiden syntymisen kokonaisuuden ympärille.

- Tuotannon kapasiteetin, tehokkuuden ja joustavuuden lisääminen. [6] Tuotteita voidaan yksilöidä kappaletasolla sarjatason sijaan, ja nopeastikin muuttuviin asiakastoiveisiin voidaan vastata pienemmällä viiveellä.
- Kunnossapidon ennakointi ja tarpeettomien seisokkien vähentäminen koneoppimiseen ja datan analyysiin perustuen. [3]
- Läpinäkyvyyden ja ymmärryksen lisääminen yrityksen omiin toimintoihin ja järjestelmiin. [3]

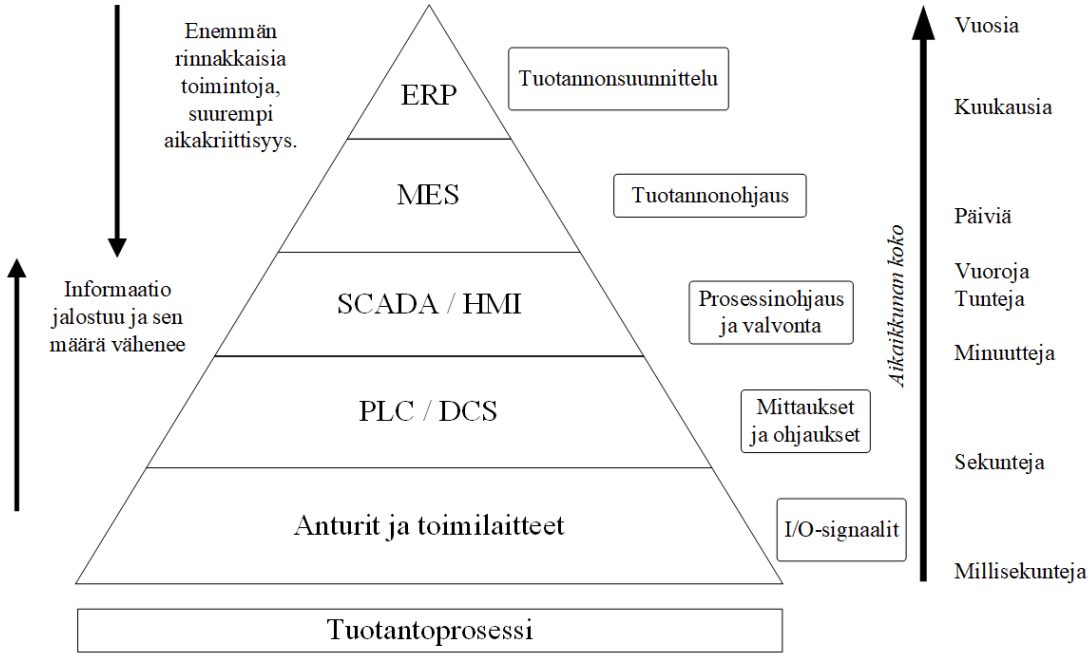
Kustannusmielessä teolliseen internetiin voidaan liittää myös käsite 1%:n merkittävyydestä (*The Power of 1%*). Vaikka sovelluksen avulla saavutettu välitön taloudellinen hyöty olisi vain 1%:n luokkaa, voi se teollisessa mittakaavassa tarkoittaa merkittävää kustannusvaikutusta. Esimerkkinä Gilchrist mainitsee ilmailun, jossa vuosittainen 1% säästö polttoainekustannuksissa voi tarkoittaa 30 miljardin dollarin säästöjä [3]. Vastaavasti myös muilla teollisuudenaloilla voidaan katsoa olevan potentiaalisia tehostamiskohteita, joihin voidaan tarttua teollisen internetin menetelmillä säästöjen saavuttamiseksi.

Teollisen internetin pohjalla vaikuttavien teknologioiden ja ratkaisujen voidaan katsoa olleen olemassa ja käytössä jo pitkään, mutta teollisen internetin malleissa uutuusarvo ja liiketoimintahyöty tulee esimerkiksi uusista toimintatavoista ja mahdollisuuksista, joita rajapintojen ja datan avautuminen ja standardoituminen mahdollistaa. Gilchrist esittää, että teollisten järjestelmien monimutkaisuus on jo ylittänyt ihmisoperaattoreiden ja prosessinohjaajien kyvyn tunnistaa järjestelmien epäkohtia, mikä tekee niiden havaitsemisesta ja toiminnan tehostamisesta perinteisin keinoin vaikeaa [3].

Tässä kappaleessa käydään läpi perinteisen automaatiojärjestelmän rakennetta ja sen käsittelemän informaation luokittelua sekä sitä miten teollinen internet muuttaa ja rikkoo tätä mallia.

2.1 Automaatiopyramidi

Automaatiojärjestelmän perinteinen hierarkkinen rakenne kuvataan automaatiopyramidilla, joka kuvaa korkealla tasolla automaatiojärjestelmän eri komponenttien keskinäistä loogista jaottelua niiden käsittelemän informaation mukaisesti. Itse fyysisen prosessin voidaan ajatella sijaitsevan itse automaatiopyramidin alapuolella ja pyramidin olevan keino hankkia siitä tietoa ja vaikuttaa siihen. Automaatiopyramidi esitetään kuvassa 5:



Kuva 5: Automaatiojärjestelmän informaatiopyramidi. Mukaillen [7] [8]

Yleisesti pyramidin huipulla olevat komponentit käsittelevät pitkän aikavälin muuttujia esimerkiksi päivien tai vuosien tasolla. Vastaavasti lähestyttäessä fyysistä prosessia ja pyramidin pohjaa siirrytään käsittelemään sekuntien tai sekunnin tuhannesosien aikajänteitä. Ylemmillä tasoilla myös informaation absoluuttinen määrä vähenee, mutta sen merkittävyys kasvaa esimerkiksi analyysin tuloksena. Alemmilla tasoilla käsitellään rinnakkaisesti useita eriäviä toimintoja ja transaktioita, ylempien tasojen keskittyessä harvempiin toimintoihin. Myös reaaliaika- ja aikakriittisyysvaatimukset ovat tyypillisesti korkeammat pyramidin alemmilla tasoilla.

Pyramidimallin mukaan tuotantoprosessin päällä olevalla ensimmäisellä kerroksella on kenttätaso, jolla sijaitsevat prosessiin suoraan vaikuttavat toimilaitteet, kuten pumput tai venttiilit sekä prosessista tietoa tuottavat anturit. Anturi- ja toimilaitetason yläpuolella sijaitsee ohjaustaso, joka sisältää antureiden ja toimilaitteiden I/O-signaaleihin kytketyt ohjelmoitavat logiikat (*Programmable Logic Controller, PLC*), sekä logiikoiden ja prosessiasemien muodostamat hajautetut ohjausjärjestelmät (*Distributed Control System, DCS*). Ohjaustason tehtävä on välittää ja laskea ohjaussignaaleja toimilaitteille ja kerätä mittauksia antureilta.

Kenttätason yläpuolella sijaitsee prosessinohjaus- ja valvontakerros, joka käsittää esimerkiksi prosessin monitoroinnin ja kunnonvalvonnan järjestelmiä. SCADA- (*Supervisory Control And Data Acquisition*) ja HMI- (*Human-machine-interface*) termeillä viitataan joukkoon sovelluksia ja laitteita, jotka ohjaavat ja valvovat prosessia [8]. Esimerkiksi prosessin operaattorille reaaliaikaisesti prosessiarvoja piirtävä infonäyttö on tyypillinen SCADA/HMI-sovellus.

Pyramidin huipun ylimmät tasot muodostavat tuotannonohjaus- ja suunnittelutasot. Yritys- ja konsernitasolla siirrettävä tieto ei enää pääasiassa ole reaaliaikaista raakadataa, vaan alempien järjestelmien datasta jalostettua informaatiota. Alempien tasojen keräämän datan päällä toimivat tyypillisesti tuotannonohjauksen ohjelmistot, eli MES-järjestelmät (*Manufacturing Execution System*). MES-kerros koostuu useista eri toiminnoista ja se voi olla yksittäinen sovellus, joka toteuttaa useita toimintoja, tai kokoelma pienempiä ohjelmistoja, jotka yhdessä toteuttavat tarvittavat MES-toiminnot. MES-toiminnallisuuksia ovat Klettin mukaan esimerkiksi tuotannonohjaus ja laatutoiminnot [9]. Pyramidin huipulla sijaitsevat ERP-järjestelmät (*Enterprise Resource Planning)* käsittelevät yleisesti yritystason dataa, kuten varastoinventaarioita, tuotannon karkeasuunnittelua, projekteja sekä taloushallintoa.

2.2 Pilvimalli

Pilvimalli, tai pilvilaskenta (*Cloud computing*), eli laskennan ja tietokoneresurssien tarjoaminen tietoverkon yli saatavana palveluna on viimeisen vuosikymmenen aikana tullut pitkälti osaksi valtavirtaa [6] tyypillisissä IT- sovelluksissa, kuten websovellusten taustapalveluna (*Back-end*) tai videon suoratoistopalvelun vaatiman suuren tiedonsiirtokapasiteetin mahdollistajana. Pilvimalli nähdään tyypillisesti osana erityyppisiä teollisen internetin sovelluksia, joissa se toimii laskenta- ja talletusresurssien tarjoajana sekä esimerkiksi maantieteellisesti hajautettujen laitosten ja yksiköiden linkittäjänä. Yhdysvaltalaisen NIST:n (*National Institute of Standards and Technology*) pilvimallin määritelmä luonnehtii sitä ”mallina, joka mahdollistaa verkkoyhteyden yli saatavissa olevien jaettujen IT-resurssien, kuten virtualisoitujen verkkojen, palvelimien, tallennustilan, sovellusten ja palveluiden käytön” [10]. NIST:n määritelmän mukaan pilvimalli koostuu viidestä pääpiirteestä (*characteristics*), jotka ovat [10]:

- *Itsepalveluperiaate*. Palvelun kuluttaja voi itsenäisesti, ilman vuorovaikutusta palveluntarjoajan kanssa, hallita omia pilviresurssejaan tarjotun rajapinnan tai käyttöliittymän kautta.
- *Riippumattomuus päätelaitteesta*. Verkon yli pilvipalveluun tarjottava rajapinta on toteutettu siten, että sitä voidaan käyttää millä vain standardilla päätelaitteella (kuten mobiililaitteella tai työasemalla).
- *Fyysisten resurssien jakaminen eri käyttäjien kesken*. Pilvitarjoajan laskentaresurssit jaetaan eri käyttäjien välillä läpinäkyvästi, ja jaottelua voidaan muuttaa joustavasti. Käyttäjä ei lähtökohtaisesti voi valita mitä resursseja juuri hänen pilvilaskentaansa käytetään, vaan ainoastaan niiden määrän. Palveluntarjoaja voi kuitenkin mahdollistaa esimerkiksi resurssien maantieteellisen sijoittelun, esimerkiksi maa- tai palvelinkeskuskohtaisesti.
- *Palveluiden joustavuus ja skaalautuvuus tarpeen mukaisesti*. Asiakas voi omien tavoitteidensa mukaisesti kasvattaa tai pienentää itselleen varattujen resurssien määrää nopeasti ja joustavasti.

- *Käytön monitorointi*. Laskentaresurssien käyttöä valvotaan kunkin palvelun kannalta sopivilla suureilla ohjelmallisesti. Monitoroinnin tuottama tieto tuo läpinäkyvyyttä sekä asiakkaalle (oman käyttöasteen arviointi) että palveluntarjoajalle (käyttöön perustuva laskutus).

NIST:n pilvimallin määritelmä kuvaa kolme pilven eri palvelutasoa (*service model*), jotka kuvaavat tarjottavien pilviresurssien tai -sovellusten abstraktiotasoa. Alimman tason IaaS-palvelut vastaavat lähinnä virtualisoitua IT-infrastruktuuria, kun taas korkeamman abstraktiotason SaaS-sovellukset tarjoavat tilaajalleen valmiin sovellusympäristön. Palvelutasot käydään tarkemmin läpi seuraavassa alaluvussa, sillä niillä on merkittävä vaikutus toteutettavan pilvipohjaisen sovelluksen rakenteeseen.

NIST määrittelee pilvipalveluille neljä eri käyttöönottomallia (*deployment model*) jotka kuvaavat pilvimallisten palveluiden tilaus- ja ylläpitoprosesseja. Käyttöönottomallit ovat yksityinen-, yhteisöllinen, julkinen sekä hybridimallinen pilvi. Yksityinen pilvi on organisaation sisäiseen käyttöön hankittu pilvi, jonka omistus- ja ylläpitovastuu on organisaatiolla itsellään, kolmannella osapuolella tai näiden yhdistelmällä, ja se sijaitsee joko paikallisesti organisaation omissa tiloissa, tai kolmannen osapuolen pilvialustalla. Yhteisöpilvi on yhteisön, tai yhteisiä etuja hakevien asiakastahojen muodostaman yhteisön käyttöön rakennettu pilvipalvelu. Omistus- ja ylläpitovastuu on yhteisöllä itsellään, kolmannella osapuolella tai näiden yhdistelmällä. Sijaitsee joko paikallisesti yhteisön omissa tiloissa, tai kolmannen osapuolen pilvialustalla. Julkinen pilvi on yleisön käyttöön suunnattu pilvipalvelu, jonka omistaa ja jota hallinnoi yksityisyritys, akateeminen- tai julkisyhteisö tai jokin yhdistelmä näistä. Julkista pilveä ylläpidetään palveluntarjoajan infrastruktuurilla. Käyttöönottomalleista viimeinen, eli hybridipilvi muodostetaan kahden tai useamman eri tyyppisen pilven yhdistelmä, siten että eri pilvet pysyvät erillisinä, mutta niitä yhdistää standardoitu tai valmistajakohtainen tekniikka, joka mahdollistaa niiden välisen tietoliikenteen. [10]

Koska pilvimalliin voidaan ottaa useita eri lähestymistapoja, ja markkinoilla on runsaasti eri tyyppisiä palveluita, voidaan pilvimallin hyödyntämiselle nähdä useita eri tavoitteita ja motivaatioita. Yleistäen voidaan kuitenkin jaotella pilvimallin etuja esimerkiksi seuraavasti [11]:

- *Kustannussäästöt*. Luopumalla omasta IT-infrastruktuuristaan yritys voi siirtää sen vaatimia kiinteitä kustannuksia (energia, työvoima, laitteiden uusiminen) pilvitarjoajan vastuulle.
- *Skaalautuvuus ja joustavuus*. Yrityksen omaa pilviympäristöä voidaan kasvattaa tarpeen mukaan, ja liiallisesta kapasiteetista voidaan hankkiutua helposti eroon.
- *Luotettavuus ja korkea saatavuus*. Pilvitarjoaja voi tarjota tietojen hajauttamista esimerkiksi erillisille fyysisille palvelimille, tai maantieteellisesti erillisiin datakeskuksiin. Näin voidaan varmistua, että palvelu on käytettävissä eritasoisista häiriöstä huolimatta.

- *Ylläpidettävyys*. Matalan tason palvelujen, kuten laitteiston ja käyttöjärjestelmän ylläpito siirtyy pilvitarjoajalle. Pilvessä sijaitsevia palveluita voidaan tyypillisesti käyttää standardien rajapintojen (API) kautta, joten myös paikallisesti asennettavien ohjelmistojen määrä henkilöstön päätelaitteilla vähenee. Ohjelmistojen konfigurointeja voidaan tehdä keskitetysti päätelaitekohtaisuuden sijaan.
- *Saavutettavuus*, esimerkiksi mobiililaitteilta ja kiinteiden verkkojen ulkopuolelta. Kun tiedot eivät sijaitse enää yrityksen omien verkkojen sisällä, on niihin pääsy helpompi tarjota esimerkiksi etätyöntekijöille tai sidosryhmille. Mahdollistaa myös integraatioita eri sovellusten välillä standardeja rajapintoja hyödyntäen.

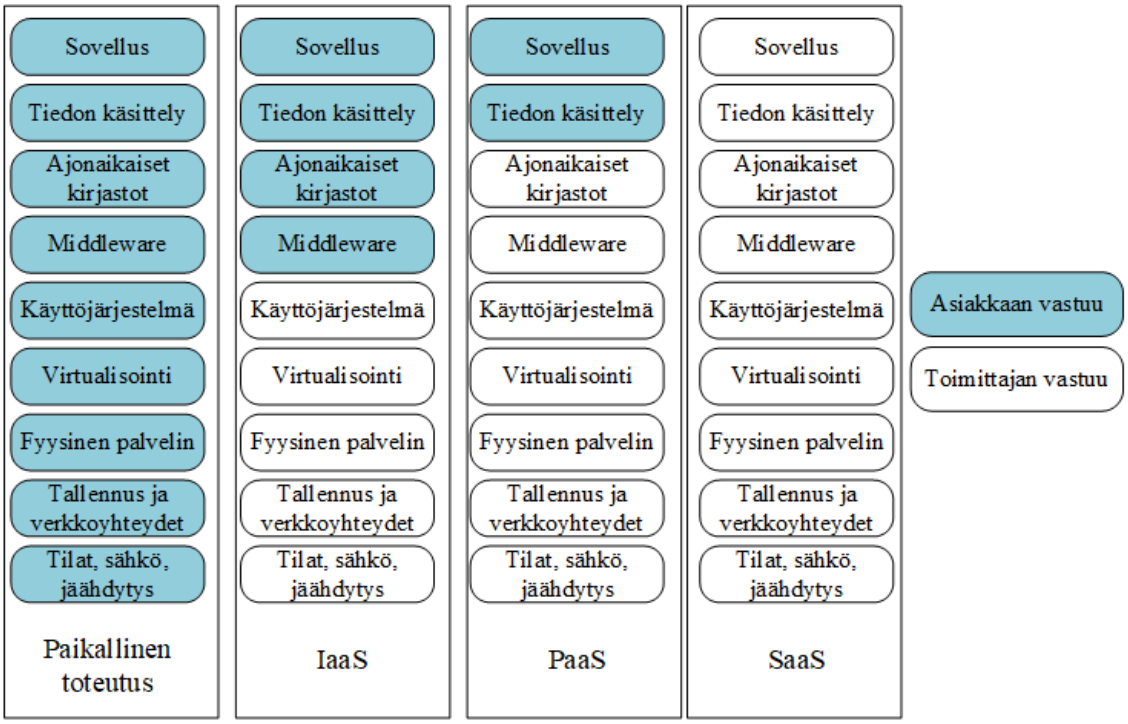
Palvelupohjaisuus ja käyttöön perustuvat hinnoittelumallit myös mataloittavat kynnystä siirtyä pilvimalliin asteittain ilman merkittäviä kertainvestointeja. Yritys voi esimerkiksi rakentaa testiympäristön vaatimattomilla resursseilla ja pienillä kustannuksilla. Lopputuloksen miellyttäessä voidaan varsinainen tuotantoympäristö rakentaa täysin vastaavasti mutta suuremmassa mittakaavassa enemmän resursseja hyödyntäen.

Pilvipohjaisten sovellusten riskejä voidaan johtaa esimerkiksi tietoturvallisuuden perusvaatimuksista kuten tiedon luottamuksellisuudesta, eheydestä ja saatavuudesta. Pilvipalveluiden käyttäjä joutuu näiden vaatimusten suhteen luottamaan pilvitarjoajan kykyyn varmistaa näiden vaatimusten täyttyminen. Esimerkiksi pilviympäristössä tapahtuva tietovuoto vaarantaa heti asiakasyrityksen sinne tallettaman datan luottamuksellisuuden. Vastaavasti virhe pilvitarjoajan ympäristössä voi johtaa datan eheyden tai saatavuuden menettämiseen, eikä pilviympäristöön tallennettuun dataan lähtökohtaisesti ole yhtä laajoja pääsymahdollisuuksia kuin yrityksen omille palvelimille tallennetulla datalla. [12] Onkin pilvisovelluksen tilaajan vastuulla selvittää miten pilvitarjoaja lupaa vastata eri tyyppisiin riskeihin, ja irtisanoutuuko se mahdollisista ongelmatilanteista. Parempi palvelutaso ja riskeihin varautuminen voi myös tarkoittaa korkeampia kustannuksia.

2.2.1 Pilven palvelutasot

Tietotekniikassa loppukäyttäjän näkemän sovelluksen taustalla on aina pino teknologioita, jotka yhdessä mahdollistavat lopullisen sovelluksen toiminnan alkaen fyysiseltä laitteistotasolta. Pilvipalvelun loppukäyttäjän näkökulmasta pilvi on abstraktio, eli jokin osa loppusovelluksen mahdollistavasta teknologiapinosta on piilotettu asiakkaalta. Tästä voi olla sekä hyötyä että haittaa suunniteltavan sovelluksen kannalta: toisaalta ylläpitovastuuta eri tasoista saadaan ulkoistettua, toisaalta osa käytettävistä teknologioista lyödään lukkoon ulkoisen osapuolen toimesta. Eri tasoisia abstraktioita kutsutaan palvelutasoiksi. NIST-määrittelyn mukaan pilvipalveluiden palvelumallit ovat IaaS (*Infrastructure as a Service*), PaaS (*Platform as a Service*), ja SaaS (*Software as a Service*) [10]. Koska loppukäyttäjä ei valitsemansa palvelumallin mukaan välttämättä edes halutessaan voi vaikuttaa palvelumallin mukaisesti piilotettuihin tekniikoihin, on tarkoituksenmukaisen palvelumallin valinta tärkeää pilviarkkitehtuurin suunnittelussa. Kavis esittääkin [13], että erityisesti yrityksen pilvistrategiaa suunnittelevien henkilöiden tulisi sisäistää pilvilas-

kennan palvelumallien eroavaisuudet. Riskinä on muuten yrityksen pilvisiirtymän muodostuminen joukoksi siiloutuneita ohjelmistoja, jotka eivät yhdessä pysty tuomaan niitä etuja mitä pilvistrategialla on alun perin lähdetty tavoittelemaan. [13]



Kuva 6: Pilvilaskennan palvelumallit. Mukaillen [14]

Kuvassa 6 esitetään tyypillisen pilviarkkitehtuuriin perustuvan sovelluksen teknologiapino, sekä eri palvelumallien tarjoama vastuunjako asiakkaan ja palveluntarjoajan välillä [14].

Pinon alimmat kolme riviä sisältävät arkkitehtuurin fyysiset kerrokset: palvelimen ylläpitoon tarvittavat fyysiset tilat, sekä niiden sähkö- ja jäähdytysenergian. Palvelimien tarvitsema tallennustila, sekä fyysiset ja loogiset verkkoyhteydet sijaitsevat myös fyysisellä tasolla. Sovelluksen käytettävissä olevan laskentakapasiteetin muodostavat kolmannella rivillä olevien fyysisten palvelimien prosessoriytimet, keskusmuisti ja väylät.

Fyysisen tason päällä sijaitsee tyypillisesti virtualisointikerros, joka mahdollistaa samojen fyysisten resurssien samanaikaisen jakamisen useiden käyttöjärjestelmien kesken. Käyttöjärjestelmä sijaitsee virtualisointikerroksen päällä, ja tarvittavat middleware-ohjelmistot sekä kirjastot sen päällä. Sovelluksen vaatimaa tiedon talletusta ja käsittelyä kuvataan toisiksi ylimmällä tasolla, joka voi käytännön sovelluksessa tarkoittaa esimerkiksi tietokantaa tai muuta tietovarastoa. Itse sovellus on kaikista ylimmällä tasolla, eli se on aina riippuvainen kaikista alemmista tasoista.

Yrityksen ylläpitäessä omaa datakeskustaan, se on vastuussa ja voi myös vaikuttaa kaikkiin tasoihin fyysiseltä laitteistotasolta lopullisen sovelluksen konfiguraatioon. Kirjallisuudessa tähän paikalliseen malliin viitataan englanninkielisellä termillä *on-premises*

deployment eli paikallinen toteutus. Varsinaisesti pilveen nojautuvat palvelumallit IaaS, PaaS ja SaaS käydään alla tarkemmin läpi.

IaaS, Infrastructure as a Service, infrastruktuuri palveluna.

IaaS tarkoittaa järjestelmän fyysisen tason ulkoistamista pilvitarjoajan vastuulle, tai kääntäen IT-infrastruktuurin hankkimista palveluna. Kuvan 6 mukaisesti IaaS-mallisena hankittavaan pilvipalveluun sisältyvät sovelluskerrokset käyttöjärjestelmätasoon asti. Tyypillisin IaaS-resurssi tarkoittaa Internetin yli saatavissa olevaa virtuaalikonetta, jonka päälle voidaan rakentaa tarvittava sovellus. IaaS-mallissa asiakas voi yleisesti vaikuttaa infrastruktuurin maantieteelliseen sijoitteluun. Esimerkiksi maailmanlaajuisesti pilvipalveluita tarjoava yritys voi antaa asiakkaan sijoittaa pilvi-infrastruktuurinsa tiettyyn maahan tai palvelinkeskukseen. Infrastruktuuriin katsotaan kuuluviksi ainakin seuraavat resurssit, ja parametrit, joiden varaan palvelun hinnoittelu on tyypillisesti rakennettu:

- Laskentaresurssit, kuten suoritinaika ja käytettävissä oleva muisti.
- Talletustila, esimerkiksi määrä ja suorituskyky.
- Verkkoyhteydet, verkon looginen rakenne ja suorituskyky.
- Käyttöjärjestelmä. Lisenssikustannukset voidaan sisällyttää muuhun hinnoitteluun. IaaS-tasolla asiakas on vastuussa ensiasennuksen jälkeisestä ylläpidosta, vaikka itse käyttöjärjestelmän asennus sisältyisikin palveluun.

PaaS, Platform as a Service, alusta palveluna

PaaS-mallin pilviresurssi tarjoaa jonkin tietyn ohjelmistoalustan, jonka päälle lopullinen sovellus voidaan rakentaa. Pilvitarjoaja tarjoaa PaaS-mallissa siis kaikki ohjelmistopinon tasot aina middlewareen ja sovelluksen vaatimiin kirjastoihin asti, kuten kuvassa 6 esitetään.

PaaS-malli mahdollistaa tyypillisesti nopeamman käyttöönoton kuin vastaava IaaS-toteutus, ja se sallii keskittymisen ensisijaisesti tarvittavan sovelluksen kehittämiseen, kun käyttöjärjestelmä ja sen päällä olevat väliohjelmistot tarjotaan jo valmiina osana palvelua [11].

Lähtökohtaisesti siis useimmat sovellukset, jotka voidaan rakentaa PaaS-resursseilla, voidaan toteuttaa myös IaaS-resurssien avulla mutta ei läheskään yhtä usein toisin päin. Riippuukin siis rakennettavan sovelluksen vaatimuksista, kumpi palvelumalli on oikea ratkaisu sovelluksen alustaksi.

Esimerkiksi pilvipalvelun avulla toteutettava tavallinen SQL-tietokanta voidaan toteuttaa hyödyntämällä joko IaaS- tai PaaS-resursseja. IaaS-mallissa hankitaan pilvessä sijaitseva virtuaalipalvelin valitulla käyttöjärjestelmällä ja tarvittavalla suorituskyvyllä ja sen päälle asennetaan haluttu SQL-palvelinohjelmisto, kuten PostgreSQL tai MySQL. Tietokantaohjelmistoon tehdään haluttu konfiguraatio ja huolehditaan että se on sitä tarvitsevien

osapuolten saavutettavissa, esimerkiksi tarpeellisen verkkokonfiguraation avulla. Lisäksi tulee varmistua, että tietokannan ja sitä ajavan palvelimen tietoturva on riittävä. Asiakkaan vastuulla on siis kaikki käyttöjärjestelmätasolta ylöspäin. Vastaavasti PaaS-mallissa voidaan hankkia suoraan pilvitarjoajan SQL-tietokantaresurssi, joka voidaan konfiguroida esimerkiksi pilvitarjoajan rajapinnan tai web-käyttöliittymän kautta. Asiakas ei voi vaikuttaa minkä käyttöjärjestelmän päällä tietokanta on, eikä välttämättä siihen mikä tietokanta se on. Esimerkiksi Microsoft Azuren tapauksessa PaaS-SQL tietokanta vastaa Microsoftin omaa SQL Server-toteutusta. PaaS-mallin tietokanta on kuitenkin nopeasti käytettävissä, eikä se vaadi varsinaisia ylläpitotoimia ohjelmistojen itsensä puolesta. Kannan tietosisällön eheydestä ja rakenteesta huolehtiminen on täysin asiakkaan vastuulla ja konfiguroitavissa, aivan kuten IaaS-mallisessa toteutuksessakin.

SaaS, Software as a Service, ohjelmisto palveluna

Palvelumalleista abstraktiotason kaikista korkeimmalle vie SaaS, eli kokonaisen ohjelmiston tarjoaminen palveluna. Kuvan 6 mukaisesti tämä tarkoittaa kaikkien sovelluspinon kerrosten olevan palveluntarjoajan vastuulla. Käytännössä SaaS-mallin sovellus tarkoittaa verkkoselaimen tai web-pohjaisen API:n kautta tarjottavaa ohjelmistoympäristöä. Käyttäjä ei ole tietoinen millaisen tietokannan tai käyttöjärjestelmän päällä sovellus toimii [11], eikä tiedä välttämättä edes käsittelemiensä tietojen maantieteellistä sijaintia. SaaS-ohjelmistot tarjotaan tyypillisesti käyttöön tai kiinteään aikaväliin perustuvalla hinnoittelulla.

SaaS-ohjelmiston käyttöönotto on nopeaa, eikä käyttäjien päätelaitteille tarvita muutoksia. Myös hinnoittelu on tyypillisesti melko joustavaa. Lisäksi sovellusarkkitehtuurin tietoturva ja lopullisen sovelluksen ylläpito ovat täysin toimittajan vastuulla. Saas-mallin ohjelmistot ovat myös lähtökohtaisesti alusta- ja sijaintiriippumattomia, eli niitä voidaan käyttää myös mobiililaitteilla ja yrityksen verkkoympäristön ulkopuolella. SaaS-ohjelmisto on kuitenkin tyypillisesti suljettu alusta, johon asiakas ei lähtökohtaisesti voi tehdä omia tarpeitansa vastaavia mukautuksia tai muuttaa sovelluksen toimintalogiikkaa. SaaS-ohjelmiston käyttäjä on siis vapautettu sovelluksen teknisestä ylläpidosta ja kehitystyöstä, mutta on vastaavasti riippuvainen ohjelmiston tarjoajan panostuksesta tuotteeseen. Esimerkiksi integraatiot yrityksen paikallisiin järjestelmiin voivat siten aiheuttaa vaikeuksia käyttöönotossa, mikäli tarkoituksenmukaista valmista rajapintaa ei ole olemassa SaaS-sovelluksessa.

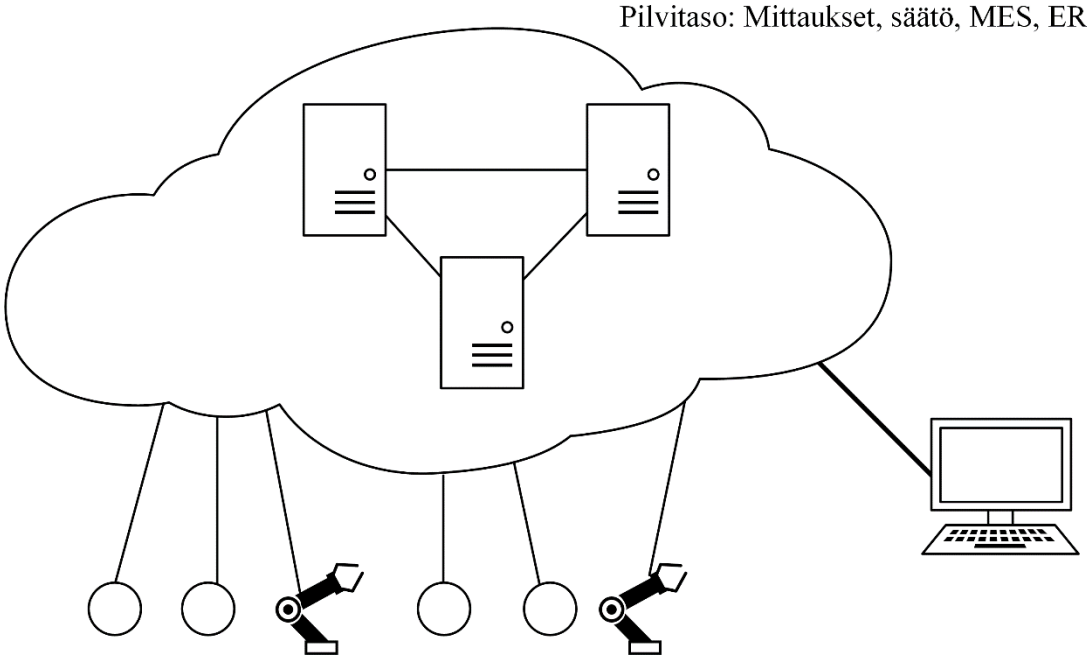
2.3 Teollisen internetin mallit

Teolliselle internetille on olemassa useita arkkitehtuuritason malleja, jotka kuvaavat esimerkiksi tehdastason tiedonkeruuta sekä kytkentää pilviympäristöön. Tässä kappaleessa käydään läpi näitä malleja ja niiden esittämiä sovellusarkkitehtuureja. Mallit toimivat myöhemmin case-arkkitehtuurien viitekehyksinä, ja ne auttavat ymmärtämään eri järjestelmien ja tietoliikenneyhteyksien sijoittelua koko järjestelmän tasolla. Tässä kappaleessa

esiteltävät mallit eivät ota itsessään kantaa tarkkaan toteutustapaan tai millä teknologioilla ne tulisi implementoida.

2.3.1 Suoraviivainen IoT-malli

Kuvassa 7 esitetään täysin pilvipohjainen teollisen esineiden internetin malli, jossa on käytännössä sovellettu suoraan esineiden internetin rakennemallia teolliseen ympäristöön: toimi- ja mittalaitteet on kytketty suoraan määrittelemättömällä teknisellä ratkaisulla pilviympäristöön, joka sisältää kaiken varsinaisen älyn ja laskennan. Myös koko järjestelmän käyttöliittymä ja hallinnointi on rakennettu pilviympäristöön, johon kiinnitytään asiakaspäätteellä. Malli on yksityiskohtaisuudessaan pitkälti samantasoinen kuin monien IoT-alustojen tarjoama arkkitehtuurimalli: teollisen ympäristön liittäminen osaksi pilvipalveluita vastaa kuvassa vain yksittäistä viivaa suoraan tehtaan lattiatasolta pilveen, eikä käytettyihin integraatiomenetelmiin oteta sen tarkemmin kantaa.



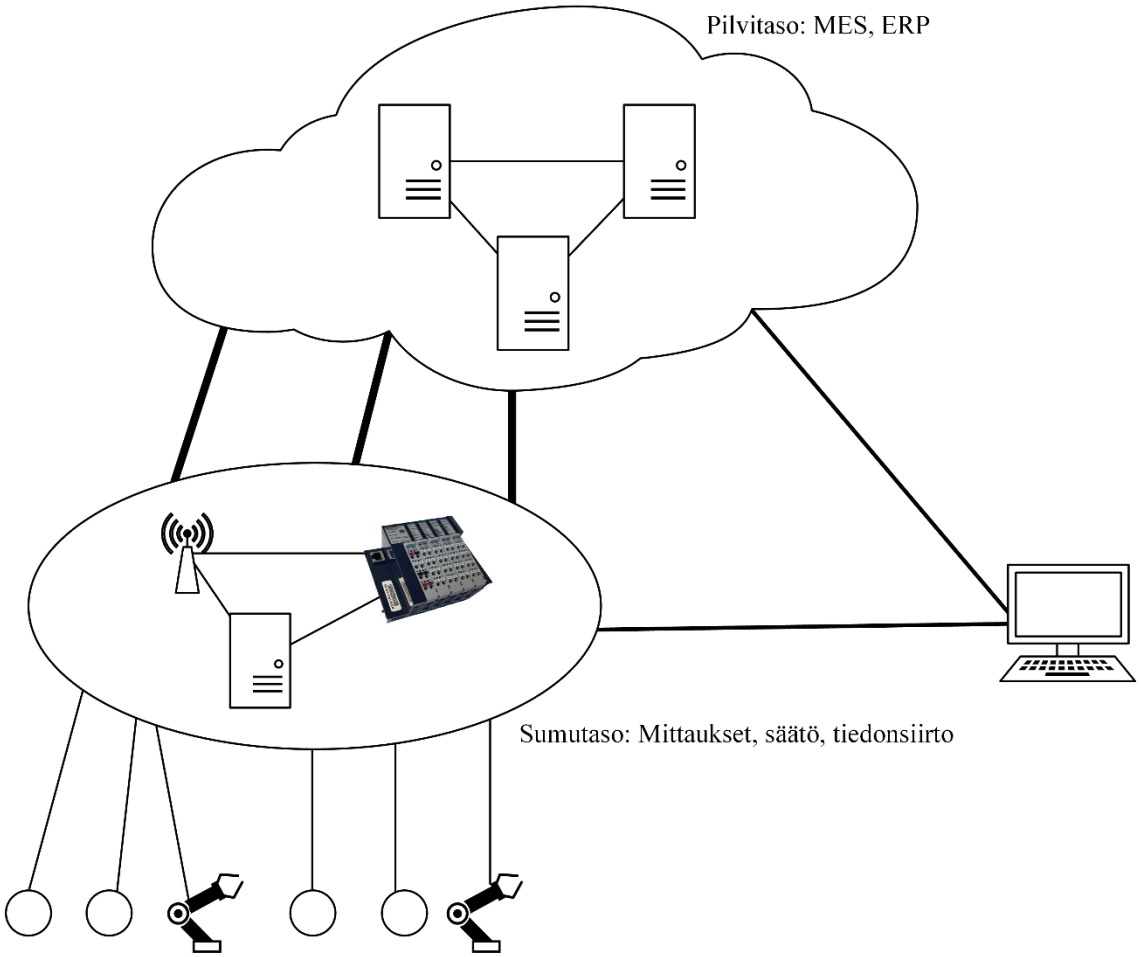
Kuva 7: Täysin pilvipohjainen IIoT-malli. Mukaillen [3].

Esitettyä mallia voidaankin kutsua automaatiopyramidin täydeksi vastakohdaksi, sillä pyramidin hierarkkisuus on täysin pilvipohjaisessa mallissa yksinkertaistunut ja litistynyt: kaikki tehdään pilvessä. Näin yksinkertaisen mallin sovittaminen mahdollisesti eri aikakausilta periytyviä legacy-järjestelmiä sisältävään teolliseen ympäristöön jättää auki useita kysymyksiä teknisen toteutuksen suhteen, ja Steiner et. al. esittävätkin mallin lähinnä perinteisen automaatiomallin täytenä vastakohtana, ja mainitsevat olevan epätodennäköistä, että teolliset toimijat pyrkisivät käytännön toteutuksissa näin äärimmäiseen malliin [6]. Haasteena mallin käytännön toteutukselle on etenkin yhteyden rakentaminen kenttätason laitteilta pilveen, sillä tyypillisesti niillä ei ole tarvittavaa laitteistokapasiteet-

tia pilven vaatimaan IP-pohjaiseen liikennöintiin, eikä turvallisuussyistä eristetyistä automaatioverkoista ole suoraa väylää julkiseen internetiin, jonka yli pilvipalvelut tarjotaan käytettäviksi. Kokonaisen tehtaan tai muun tuotantolaitoksen koko anturoinnin ja toimilaitteiden muuntaminen IP-pohjaista liikennöintiä tukevaksi on merkittävä investointi, eikä välttämättä myöskään kaikkien toimintojen osalta edes mahdollinen nykyisten markkinoilta löytyvien tuotteiden puitteissa. Lisäksi matalan tason tehdasjärjestelmien tuottaman datan esitystapa ja mittaustaajuus voivat vaihdella huomattavasti, mikä aiheuttaa lisähaastetta sen mallintamiseen ja käyttöön pilviympäristössä. Jokseenkin homogeenisten ja IP-pohjaisten laitteiden seurantaan ja datan keruuseen malli sinänsä on toimiva.

2.3.2 Sumumalli

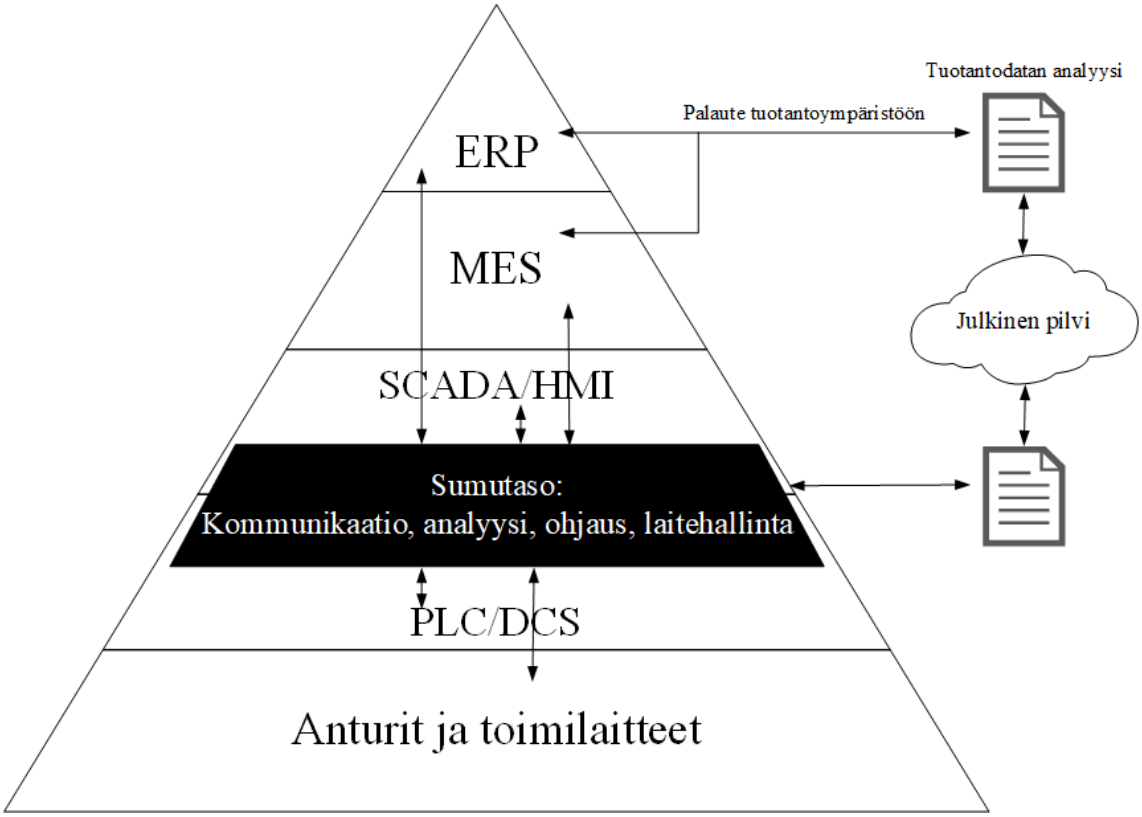
Täysin pilvipohjaiselle IIoT-mallille käytännön teollisuusjärjestelmiin sopivammaksi vaihtoehdoksi Steiner et. al. esittävät ns. sumumallin (*Fog Computing*) hyödyntämistä [6]. Sumumallissa pilvimallin ydintekniikoita ja toimintamalleja kuten virtualisointia ja kehittyneitä verkkoratkaisuja hyödynnetään paikallisissa järjestelmissä. Bonomi et al. määrittelevät sumumallin ”Tyypillisesti verkon reunalla sijaitsevaksi virtualisoiduksi alustaksi, joka tarjoaa laskentaa, tiedon talletusta ja verkkopalveluja matalan tason laitteiden ja perinteisten pilvidatakeskusten välillä” [15]. Järjestelmän sumutason muodostaa joukko heterogeenisiä laitteita, joita yhdistää käytännössä vain kyky tarjota laskentaresursseja sekä tiedon talletusta ja käsittelyä verkkoyhteyden yli [16]. Tämä rakenne esitetään kuvassa 8.



Kuva 8: Sumu- ja pilvitasoille hajautettu IIoT-malli. Mukaillen [6].

Heterogeenisyys on yksi sumutasoa määrittävistä tekijöistä, ja yksi syy sen potentiaalille teollisessa ympäristössä: sumutason toiminnallisuus voidaan rakentaa jo olemassa olevien laitteiden ja protokollien varaan. Teollisuuslaitoksen ympärille rakennettavaa sovellusta ajatellen sumumallin muita tärkeitä ominaisuuksia ovat puhtaaseen pilvimalliin verrattuna esimerkiksi reaaliaikavaatimusten helpompi täyttäminen pienempien välimatkojen ja luotettavampien tunnettujen verkkojen myötä, sekä mahdollisuus eristää tuotantoympäristön tietoliikenne omaan tietoturvalliseen aliverkkoonsa.

Matt sovittaa sumumallin automaatiopyramidin päälle kuvan 9 mukaisesti:



Kuva 9: Sumumalli automaatiopyramidissa. Mukaillen [16].

Matt'n mukaan sumumalli käytännössä toteuttaa SCADA/DCS- ja PLC-tasojen välissä sijaitsevan uuden kerroksen automaatiopyramidiin. Sumutaso siis tuo kokonaan uuden toiminnallisen kerroksen automaatiojärjestelmään, eikä se synny pelkästään tuomalla älykkyyttä jo olemassa oleviin laitteisiin [16]. Sumukerros voidaan siis nähdä eräänlaisena palvelukerroksena eri automaatiotasojen välillä: kuvan 9 mukaisesti sumutasolla on suora yhteys jokaiseen automaatiopyramidin muuhun tasoon, minkä lisäksi se voi hyödyntää pilvessä sijaitsevia, tyypillisesti huomattavan laajoja, laskentaresursseja. Sumutason palvelut eivät siis ole riippuvaisia niitä ympäröivistä PLC/DCS- ja SCADA/HMI-tasojen tarjoamista palveluista ja datasta, vaan ne voivat käyttää joustavasti myös anturi- ja toimilaitetasojen raakadataa, tai yritystason MES- ja ERP-järjestelmien jalostuneempaa informaatiota. Vastaavasti myös alempien ja ylempien kerrosten järjestelmissä voidaan hyödyntää sumutason tarjoamia palveluita suoraan joustavammin kuin konfiguroimalla koko pyramidin läpi kulkeva toiminnallinen polku eri tasoisten järjestelmien läpi. Jossain määrin sumutasolle rakennettava toiminnallisuus myös korvaa ja syrjäyttää automaatiopyramidin hierarkkisia palveluita.

Sumutaso tarjoaa käytännön kannalta realistisemman keinon hyödyntää pilvipalveluita tehtaan kenttätasolla, kuin IoT-mallit, jotka esittävät kenttätason kytkemistä suoraan pilveen. Sumutason palveluilla tarvittava pilvitoiminnallisuus voidaan tarjota kenttätasolla jo olevien laitteiden ymmärtämän rajapinnan kautta ja toteutuksessa voidaan huomioida

myös tietoturva ja suorituskyky sen tarvitsemalla tasolla. On kuitenkin kyseenalaista, voidaanko enää puhua automaatiopyramidin hierarkkisuudesta, kun sumutaso käsittelee kaikkia kerroksia suoraan, mikä litistää tiedonvälitystä eri tasojen välillä

2.3.3 Reunamalli

Reunamalli *(Edge computing*) kuvaa yleensä pilvessä tapahtuvan tiedon prosessoinnin ja sovellusälyn tuomista järjestelmän reunalle, eli lähemmäs sen syntypaikkaa, jolloin saadaan esimerkiksi tiedonsiirron viiveitä minimoitua [17]. Reunalla voidaan esimerkiksi itsessään ”tyhmän” toimilaitteen läheisyydessä päätellä toimilaitteen vikaantuvan pian, ja se voidaan esimerkiksi kytkeä pois päältä jopa millisekuntien viiveellä. Pilvipohjaisella toteutuksella viive tiedonkeruun, analytiikan ja päätöksen palauttamisen takaisin toimilaitteelle jälkeen saattaisi olla liian suuri ja johtaa siten toimilaitteen vikaantumiseen. Teollisen internetin järjestelmässä reuna siis tarkoittaa tiedon prosessointia automaatiojärjestelmän alimpien tasojen välittömässä läheisyydessä. IIC:n mukaan reunan tarkan sijainnin määrittely on kuitenkin tapauskohtaista, ja reuna kuvaa enemmänkin loogista järjestelmän tasoa kuin tarkkaa fyysistä sijaintia. [19]

Reuna- ja sumumallien välinen ero ei aina ole täysin selkeä sillä molempia termejä voidaan eri lähteissä käyttää jokseenkin sekaisin, mutta esimerkiksi Linthicum määrittelee reunan niiksi palveluiksi, jotka mahdollistavat tiedon käsittelyn järjestelmän reunalla ja vähentävät näin tarvetta siirtää suuria tietomääriä reunan ja pilven välillä. Tässä jaottelussa sumumalli kuvaa enemmänkin reunalla olevaa virtuaalista IT-infrastruktuuria, kuten laskenta- ja talletuskapasiteettia sekä talletustilaa, jota voidaan käyttää reunamallisten palveluiden pohjana. [17]

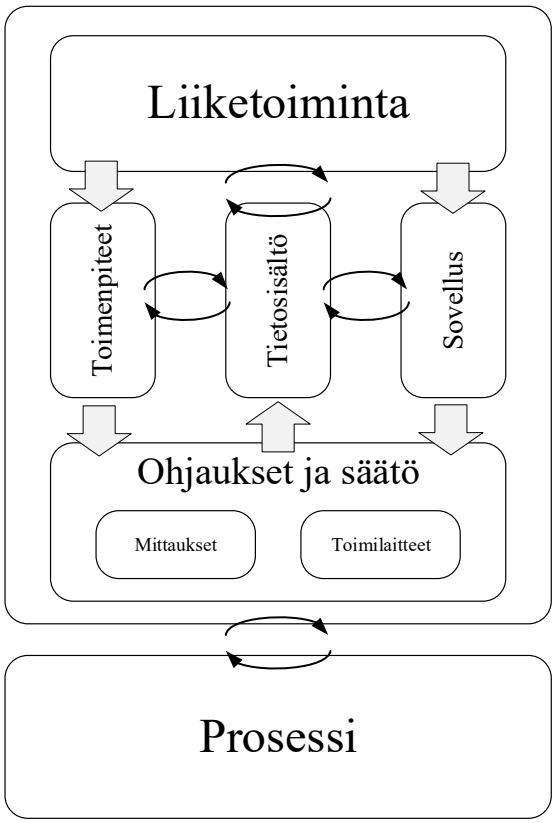
2.3.4 IIC IIoT-referenssiarkkitehtuuri

IIC (*Industrial Internet Consortium*) on teollisuusyritysten ja julkisten toimijoiden yhteenliittymä, jonka tavoitteena on edistää teollisen internetin menetelmien yleistymistä ja alan yritysten verkostoitumista [20]. IIC kehittää ja julkaisee referenssiarkkitehtuureja ja käytännön toteutusesimerkkejä, joiden se katsoo lisäävän avoimuutta ja vievän eteenpäin koko alan kehitystä. Koska IIC ei ole standardoimisjärjestö, sen julkaisemat mallit ovat ennemminkin suuntaa antavia raameja käytännön toteutuksille ja niitä voidaan soveltaa käytäntöön joustavasti. Mallit eivät ota kantaa tarkkaan tekniseen toteutukseen. IIC:n lisäksi muitakin toimijoita on hahmottelemassa alan termistöä ja konsepteja. Suomalaista näkökulmaa edustaa esimerkiksi elinkeinoelämän tutkimuslaitoksen ETLA:n kooste ” Suomalainen teollinen internet – haasteesta mahdollisuudeksi” [21]. IIC:n malli käydään tässä tarkemmin läpi, sillä se on myös muissa viitteissä paljon käytetty lähde ja malli on yleiskäyttöinen kuvaamaan monia eri toteutusmahdollisuuksia.

IIC:n IIoT-referenssiarkkitehtuuri tarkastelee teollisen internetin järjestelmää neljästä eri näkökulmasta: liiketoiminta, käyttö, toiminnallisuus ja käytännön toteutus. Liiketoimintanäkökulma tarkastelee järjestelmän istuvuutta kokonaisliiketoimintaan, sekä järjestelmään tehtyjen investointien kannattavuutta. Käyttönäkökulma toteuttaa järjestelmän liiketoimintaodotukset määrittelemällä tarpeelliset käyttö- ja järjestelmävaatimukset. Nämä vaatimukset toimivat syötteenä järjestelmävaatimuksille, jotka määrittelevät järjestelmän toiminnallisuuden [3].

Toiminnallinen näkökulma

IIC:n referenssimallin toiminnallinen näkökulma huomioi kaikkien järjestelmän sidosryhmien sitä koskevat toiminnalliset tarpeet. Toiminnallinen näkökulma on mallissa jaettu viiteen eri alueeseen, jotka yhdessä muodostavat järjestelmän toiminnallisuuden: liiketoiminta, toimenpiteet (operaatiot), tietosisältö, sovellus sekä ohjaukset ja säätö. Toiminnallinen näkökulma esitetään kuvassa 10.



Kuva 10: Teollisen internetin järjestelmän toiminnalliset alueet. Mukaillen [3]

Toiminnallisen näkökulman kukin alue (*Domain*) koostuu useammista toiminnoista (*Function*) jotka yhdessä toteuttavat alueen toiminnallisuuden. Teollisten järjestelmien laajuuden ja monimutkaisuuden vuoksi IIC jaottelee toiminnallisen näkökulman siten että se on sovellettavissa useille eri teollisuuden aloille ja niiden erityisvaatimuksille. Käytännön toteutuksessa yksi järjestelmä voi siis toteuttaa useita toiminnallisia alueita, tai use-

ampi järjestelmä voi toteuttaa yhden toiminnallisen alueen yhdessä. Kaikki toiminnallisuudet eivät välttämättä esiinny kaikissa järjestelmissä [3]. Malli ei myöskään pakota muuttamaan järjestelmän jo toimivia osia: teollisen internetin järjestelmän tarvittavat toiminnot voidaan edelleen toteuttaa vastaavasti kuin nykyisessä automaatiojärjestelmässä, eikä niiden siirtämistä esimerkiksi pilvipohjaisiksi vaadita. Kuten automaatiopyramidissakin, myös IIC-mallissa itse prosessi kuvataan teollisen järjestelmän alapuolelle erillisenä alueena, johon vaikutetaan vain järjestelmän kautta. Toiminnallisen tason eri alueet IIC:n referenssimallissa ovat kuvan 9 mukaisesti ohjaukset ja säätö, toimenpiteet, tietosisältö, sovellus sekä liiketoiminta. [18]

Ohjaukset ja säätö -alueelle kuuluvat tyypillisesti antureiden luenta sekä prosessin ohjaaminen takaisinkytkennän ja säätöalgoritmin avulla. Alueelle kuuluvat yksittäiset toiminnot ovat:

- Kommunikaatio. Suorittaa protokollamuunnoksia alemman ja ylemmän tason protokollien, kuten kenttäväylien ja IP-pohjaisten pilviprotokollien välillä mahdollistaen kahdensuuntaisen tietoliikenteen. Toimintoa suorittaa esimerkiksi ohjelmistopohjainen yhdyskäytävä.
- Mallinnus tai reuna-analytiikka (*Edge analytics*). Käsittelee järjestelmän tiloja, ehtoja ja vasteita ja mallintaa sen toimintaa näiden perusteella.
- Järjestelmän resurssien hallinta (*Asset management*). Huolehtii esimerkiksi järjestelmän komponenttien ohjelmisto- ja firmwarepäivityksistä ja konfiguraatiosta, sekä näihin liittyvästä säännöstöstä.
- Toimeenpano. Säätää ohjattavaa järjestelmää valitun algoritmin perusteella. Algoritmi voi olla tyypiltään suoraviivainen kuten PID, tai hienostuneempi, perustuen esimerkiksi koneoppimiseen ja data-analytiikkaan.

Toimenpiteet-alueen toimintojen tavoitteena on saattaa alempiin ohjaustasoihin suoraan liittyviä toimintoja käytettäviksi verkkoyhteyksien yli. Alue pyrkii mahdollistamaan sen kattamien palveluiden käytettävyyden kustannustehokkaasti ja tietoturvallisesti fyysisestä paikasta ja järjestelmän mittakaavasta riippumatta. Toimenpide-alueelle kuuluvia toimintoja ovat [18]:

- Järjestelmänhallinta tarjoaa rajapintoja, joilla ohjaukset- ja säätö- alueen resurssienhallintafunktioita voidaan soveltaa kentällä sijaitseviin resursseihin etäyhteyksien yli. Esimerkiksi ohjelmistopäivityksiä voidaan tarjota ja hallita keskitetysti eri tuotantolaitoksille yhdestä sijainnista käsin.
- Monitorointi ja diagnostiikka on joukko toimintoja, joilla mahdollistetaan vikojen ja ongelmatilanteiden tunnistaminen ja ennakointi. Toiminnolla on riippuvuussuhde historia- ja reaaliaikatietoon perustuvaan ennustamiseen.
- Ennakointi on jatkumoa monitoroinnin ja diagnostiikan funktioille, ja sen roolina on toimia ennustavan analytiikan moottorina teollisen internetin järjestelmässä.

Perustuu suurten data-aineistojen historialliseen ja ennustavaan analyysiin, jonka perusteella voidaan mahdollisesti ennakoida esimerkiksi tuotantolaitteiden potentiaalisia vikatiloja.
- Optimointifunktiot tähtäävät resurssien suorituskyvyn ja tehokkuuden parantamiseen. Ihannetilanteessa resurssien luotettavuus ja suorituskyky maksimoidaan ja energiankulutus minimoidaan. Optimointiin kuuluvia toimintoja ovat automaattinen tiedonkeruu, järjestelmän tapahtumien kuten hälytysten ja ongelmien havaitseminen ja käsitteleminen, sekä älykkyyden soveltaminen ongelmatapausten juurisyiden löytämiseen.

Tietosisältö-alueen toiminnoilla muiden toiminnallisten alueiden keräämää raakadataa pyritään jalostamaan informaatioksi, jota käytetään ohjauksen takaisinkytkentänä prosessin vakauttamiseen tai parantamiseen. Kerätystä informaatiosta voidaan edelleen tehdä johtopäätöksiä, jolloin saadaan tietoa ja ymmärrystä kohteena olevasta järjestelmästä ja sen toiminnasta. Ymmärrys palvelee erityisesti liiketoiminnan päätöksentekoa ja helpottaa sen ohjaamista parempaan tehokkuuteen. IIC:n referenssimalli määrittelee alueen toiminnot seuraavasti [18]:

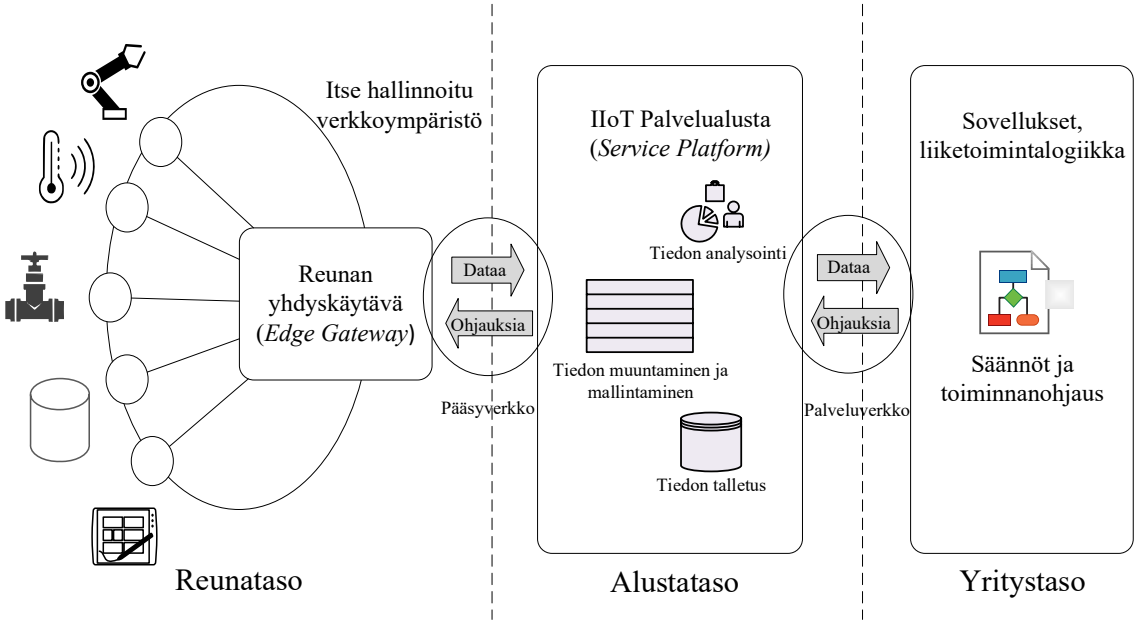
- Raakadataan liittyvät funktiot muokkaavat muiden toiminta-alueiden, käytännössä pitkälti ohjaukset- ja säätöalueen tuottamaa mittausdataa yhtenäisempään ja hyödyllisempään muotoon. Yksittäisiä raakadatalle tehtäviä toimenpiteitä ovat suodattaminen, puhdistaminen, duplikaattien poistaminen sekä syntaksiset ja semanttiset muunnokset.
- Dataa voidaan käsitellä joko reaaliaikaisina tietovirtoina tai se voidaan tallettaa ja analysoida eräajoina myöhemmin. Formaalimmin referenssiarkkitehtuuri määrittelee datan tallennuksen, säilyvyyden ja jakelun toiminnot.
- Analytiikan toimintoihin kuuluvat mallintaminen, analytiikka ja kehittyneen datankäsittelyn menetelmät. Myös analytiikan toimintoja voidaan käyttötapauksen mukaan soveltaa joko tietovirtoihin tai dataeriin, ensimmäisen tuottaessa tyypillisesti hälytyksiä ja muita käsiteltäviä tilanteita sovellusalueelle, ja jälkimmäisten tulosten hyödyttäessä liiketoiminnan toimintoaluetta.

Sovellusalue toteuttaa liiketoiminnan asettamia vaatimuksia ja hallinnoi sovelluslogiikkaa. Sovellusalueen sovelluksilla ei ole mahdollisuutta tai oikeuksia ohjata laitteita tai järjestelmän prosesseja suoraan, vaan ne toimivat neuvoa antavassa roolissa. Sovellusalueelle liittyvät myös API-rajapintoihin ja käyttöliittymiin liittyvät toiminnot.

Liiketoiminta-alueen toiminnot mahdollistavat yhteensopivuuden ja sen vaatimat integraatiot teollisen internetin järjestelmän ja automaatiopyramidin ylimpien kerrosten kuten ERP- ja CRM (*Customer Relationship Management,* asiakkuuksien hallinta) -järjestelmien kanssa.

Toteutusnäkökulma

IIoT-referenssiarkkitehtuurin toteutusnäkökulmaan kuuluu kolmitasoinen topologia, joka jaottelee teollisen internetin järjestelmän korkealla tasolla kolmeen osaan: reunatasoon (*Edge Tier*), alustatasoon (*Platform Tier*) ja yritystasoon (*Enterprise Tier*) [3]. Kuten toiminnallisten alueiden määrittelyissä, myöskään referenssiarkkitehtuurin toteutusnäkökulman arkkitehtuurimalleissa ei pyritä määrittelemään tarkkaa arkkitehtuuria, jota jokaisen teollisen internetin sovelluksen tulisi noudattaa. Esimerkiksi kolmitasoisen referenssiarkkitehtuurin tasoja vastaavia osia voi käytännön toteutuksessa olla useita rinnakkaisia, jotka yhdessä toteuttavat niitä vastaavan referenssitason toiminnallisuuden [18]. Näin referenssiarkkitehtuuri pyrkiikin enemmän ohjaamaan toimintojen loogista sijoittelua ja tietovirtojen reittejä kuin toimimaan käytännön järjestelmäarkkitehtuurina. Kolmitasoinen referenssiarkkitehtuuri esitetään kuvassa 11.



Kuva 11: Kolmitasoinen IIoT-malli. Mukaillen [3]

Kolmitasoisen mallin reunatasolla kerätään, koostetaan ja siirretään teollisen ympäristön kuten automaatiojärjestelmän ja sen sisältämien laitteiden dataa reunatason ja alustatason väliseen yhdyskäytävään (*Edge Gateway*). Referenssiarkkitehtuurin toiminnallisista alueista reunatasolle kuuluvat ensisijaisesti ohjaukset ja säätö -alueen toiminnot. Reunatasolla voi esiintyä useita erityyppisiä ja useiden eri valmistajien suljettuja teollisuusprotokollia ja tiedon lähteet voivat sijoittua automaatiopyramidin (kuva 5) eri tasoille. Reunatasolla voidaan esimerkiksi lukea suoraan mittaus- ja ohjausdataa ohjelmoitavilta logiikoilta tai jalostuneempaa tietoa tehtaan paikallisesta MES-järjestelmästä. Myös tiedonsiirtoväyliä voi reunatason sisällä olla rinnakkaisesti useampia, kuten paikallisia verkkoyhteyksiä ja kenttäväyliä tai suoria yhteyksiä langattomiin antureihin. Reunatason tiedonlähteitä ovat esimerkiksi toimilaitteet, anturit, paikalliset tietokannat sekä laitoksen hen-

kilökunnan käyttöliittymät, kuten vikatietojen syöttöpaneeli. Edellisessä kappaleessa esitettyjen sumu- ja reunamallien tarjoamat toiminnot sisältyvät pääosin reunatasolle, ja nämä tasot voivat itsessään toteuttaa joitakin alustatason toimintoja esimerkiksi raakadatan käsittelyyn ja analyysiin liittyen. Näin voidaan mahdollisesti vähentää reuna- ja alustatasojen välillä siirrettävän datan määrää ja sen myötä pilvialustan käytöstä aiheutuvia kustannuksia, sekä saavuttaa muita reunamallin etuja.

Reunatason yhdyskäytävän ja alustatason välillä sijaitsee pääsyverkko, joka viittaa tyypillisesti esimerkiksi yrityksen omaan verkkoon, julkisen internetin päällä (VPN) tai sen ohessa toimivaan yksityiseen verkkoon, joka on vain yrityksen omassa käytössä, tai langattomaan mobiiliverkkoon [18]. Pääsyverkon tehtävä on mahdollistaa kahdensuuntainen kommunikaatio tasojen välille: reunatasolta siirretään raakaa dataa ja telemetriaa alustatasolle ja alustatasolta ohjaustietoja ja -logiikkaa takaisin reunatasolle.

Alustatasolle sijoittuvat pääasiassa tietosisältö- ja toimenpidealueiden toiminnot, eli muun muassa datan käsittely, data-analyysi, laitteiden hallinta ja ennusteiden tekeminen. Alustataso käsittelee raakoja tietovirtoja ja muodostaa niistä informaatiota yritystason toimintojen käyttöön. Alustataso myös välittää ja tulkkaa yritystasolta tulevat ohjaukset reunatasolle sen ymmärtämässä muodossa.

Alusta- ja yritystasoja yhdistää palveluverkko, joka voi olla pääsyverkkoa vastaava yksityinen väylä, tai suoraan julkisen internetin kautta reitittyvä yhteys [18]. Palveluverkon tehtävänä on tarjota tietoturvallinen yhteys alusta- ja yritystason palveluiden sekä sovelluksen loppukäyttäjien välillä.

Viimeisen kolmanneksen arkkitehtuurimallissa muodostaa yritystaso, joka toteuttaa yritys- ja sovellusalueiden toiminnallisuuksia, eli integraatioita muihin liiketoiminnan järjestelmiin ja loppukäyttäjälle näkyvän osuuden sovelluksesta joko käyttöliittymän tai rajapinnan muodossa. Yritystaso hyödyntää pääasiallisesti alustatason palveluiden tuottamaa tietoa raakadatan sijaan, ja tekee sen perusteella järjestelmää koskevia päätöksiä. Yritystason päätökset ohjataan takaisin alustatasolle, ja sen kautta tarvittaessa reunatasolle. Näin ylätason päätös koskien esimerkiksi tuotannon aikataulutusta tulkkaantuu esimerkiksi yksittäistä toimilaitetta koskevaksi ohjausviestiksi.

3. TEKNISET RATKAISUT

Tässä luvussa käydään läpi teollisen internetin järjestelmässä käsiteltävän datan luonnetta, sekä teknisiä ratkaisuja, joiden avulla mahdollistetaan sen käsittely käytännössä. Lisäksi esitellään kaupalliset ohjelmistot ja alustat, jotka toimivat pohjana seuraavan luvun käytännön arkkitehtuureille.

3.1 Käsiteltävä tieto

Teollisen internetin järjestelmän käsittelemän tiedon luonne muotoutuu pitkälti järjestelmälle asetettujen tavoitteiden ja vaatimusten mukaisesti. Tämän myötä eri tyyppisiä tehtäviä suorittavat järjestelmät voivat käsitellä dataa useilla keskenään eriävillä tavoilla, eikä ole yhtä selkeästi oikeinta tapaa järjestää teollisen internetin järjestelmän datan siirto, käsittely ja talletus. Tässä kappaleessa listataan datan eri ominaisuuksia, jotka on huomioitava teollisen internetin järjestelmän tietoliikenteen ja tiedonsiirron integraatoiden suunnittelussa. Tunnistettuja ominaisuuksia ovat tiedonsiirron kriittisyys ja reaaliaikavaatimukset, tiedon jäsentely ja käsittelymuoto, datan määrä, sekä tiedonsiirron suunnat järjestelmätasolla.

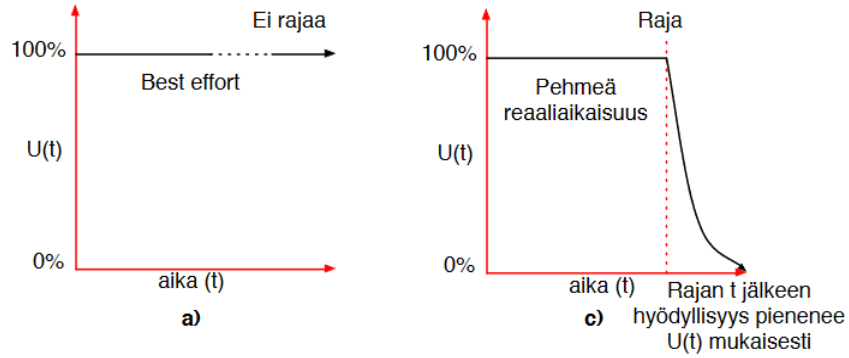
3.1.1 Tiedonsiirron kriittisyys ja reaaliaikavaatimukset

Tiedonsiirron kriittisyys kuvaa kuinka merkittäviä ongelmia järjestelmän toiminnalle mahdolliset tiedonsiirron viiveet, puutteet ja katkokset aiheuttavat. Kriittisyyden arvioinnissa voidaan arvioida myös järjestelmän itsensä kriittisyyttä kokonaisuutena, josta voidaan johtaa sen tiedonsiirtoon kohdistuva riskiarvio ja reaaliaikavaatimukset.

Kriittiseksi luokiteltavan järjestelmän häiriötilanteista voi seurata merkittäviä negatiivisia vaikutuksia itse järjestelmään tai sitä ympäröiviin muihin järjestelmiin, sekä mahdollisesti myös sen fyysiseen ympäristöön. Järjestelmän tiedonsiirtovaatimukset on siis arvioitava tätä taustaa vasten, ja varmistettava että se kykenee täyttämään ne, jotta tiedonsiirrosta ei muodostu pullonkaulaa järjestelmän kokonaissuorituskyvyn kannalta.

Reaaliaikavaatimukset voidaan jaotella karkeasti kahteen osaan, pehmeisiin ja koviin vaatimuksiin. Kuvassa 12 esitetään pehmeän reaaliaikaisuuden muodot saapuneen tiedon

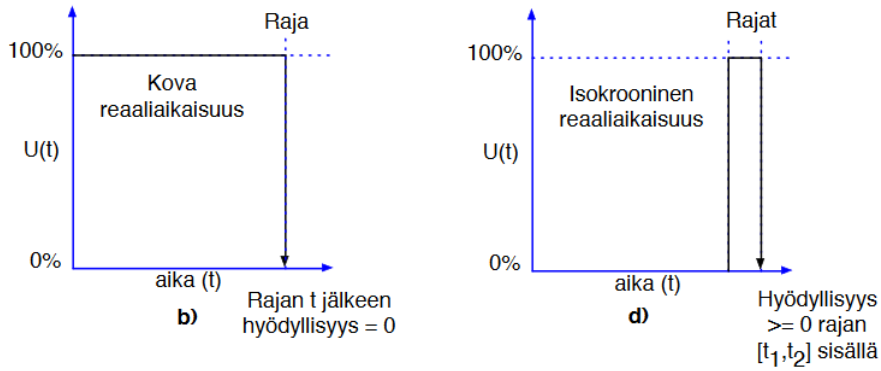
hyödyllisyyttä ajan funktiona kuvaavan käyrän $U(t)$ avulla.



Kuva 12: Pehmeät reaaliaikavaatimukset. Mukaillen [22]

Kaikista kevein reaaliaikavaatimus on best-effort tilanne, jolloin reaaliaikaisuudelle ei ole erityisiä vaatimuksia, vaan saapuneen tiedon hyödyllisyyteen ei vaikuta sen saapumishetki. Varsinainen pehmeä reaaliaikavaatimus kuvaa tilannetta, jossa datan oletetaan saapuvan tietyllä hetkellä, mutta myös vanha tai viivästynyt data on käyttökelpoista vaikkakaan ei välttämättä yhtä hyödyllistä. Ei-kriittisessä ja reaaliaikavaatimuksiltaan pehmeäksi luokiteltavassa tiedonsiirrossa tiedonsiirron häiriöt tai yksittäisten datapisteiden puuttuminen eivät aiheuta ongelmia järjestelmän toiminnalle. Esimerkiksi historiadataa käsittelevä data-analytiikkasovellus voi kyetä toimimaan pehmeillä reaaliaikavaatimuksilla, sillä sen datan syntymis- ja käyttöhetkien välillä voi olla esimerkiksi useita tunteja tai päiviä.

Tiedonsiirrolta enemmän vaativien järjestelmien tarpeita kuvaavat kovat reaaliaikavaatimukset esitetään kuvassa 13. Kovat reaaliaikavaatimukset ovat tyypillisiä automaatioympäristöissä, esimerkiksi ohjaus- ja säätösovelluksissa. Niitä voidaankin pitää eräänä vedenjakajana IT-järjestelmien ja teollisuusjärjestelmien välillä [22].



Kuva 13: Kovat reaaliaikavaatimukset. Mukaillen [22]

Varsinainen kova reaaliaikavaatimus kuvaa tilannetta, jossa odotetusta saapumishetkestä myöhästyvä data on järjestelmän näkökulmasta hyödytöntä. Esimerkiksi ajanhetkellä $t + 1$ saapuva mittaussignaali on hyödytön ajanhetkellä t tapahtuvaa säädön laskentaa varten.

Hieman enemmän joustovaraa antaa isokrooninen reaaliaikaisuus, joka määrittelee hyödyllisyyden tietylle aikavälille. Isokroonisen reaaliaikavaatimuksen myötä järjestelmän tiedonsiirron viiveelle voidaan määritellä jokin toleranssiarvo, jonka alittavat datapisteet voidaan hyväksyä eikä niiden sisältämän viiveen katsota estävän järjestelmän toimintaa.

3.1.2 Datan jäsentely ja käsittelymuoto

Data voidaan lukea järjestelmään suoraan sen alkulähteeltä, eli tyypillisesti joltakin tehtaan alijärjestelmältä tai anturilta. Kun rinnakkaisia datalähteitä ja niiden tuottamaa dataa on paljon, on kyseessä jäsentelemätön raakadata. Jäsentelemätön data voidaan esittää esimerkiksi nimi-arvopareina, tai taulukkomuodossa, jolloin vastaavat tekstimuotoiset tiedostot ovat tyypillisesti JSON ja CSV. On huomattava, että tässä käsitellään nimenomaisesti tekstimuotoisia esityksiä – esimerkiksi valmistajakohtaiset suljetut protokollat voivat käsitellä dataa binäärimuotoisena tiedonsiirron tehokkuuden maksimoimiseksi tai liiketoiminnallisista syistä. Tällöin on täysin protokollan kehittäjästä kiinni, avaako se protokollaansa muiden sovellusten luettavaksi.

Ohjelma 1 on tekstimuotoinen JSON-pseudokoodiesitys, joka sisältää erään mallin jäsentelemättömän datan käsittelyyn sovellusten välisessä liikennöinnissä ja dokumenttipohjaisessa datan varastoinnissa:

```
{
 "timestamp": |SERVERTIMESTAMP|,
  "values": [
   |#each VALUES|
    {"|TAGNAME|": |VALUE|,"t": |TIMESTAMP| } |#unless @last|,|/unless|
   |/each|
  ]
}
```

Ohjelma 1. *Raakadatan JSON-muotoon generoinnin pseudokoodiesitys.*

Ohjelman 1 uloin JSON-objekti kuvastaa yksittäistä viestiä tai dokumenttitietokannan tietuetta. Objekti sisältää viestin luoneen palvelimen aikaleiman `|SERVERTIMESTAMP|` sekä taulukkotietueen `values`, joka sisältää objektitietueen jokaisen datapisteen jokaiselle mittaukselle. Yksittäinen values-taulukon sisältämä objekti koostuu datapisteen nimestä `|TAGNAME|` ja arvosta `|VALUE|` sekä aikaleimasta `|TIMESTAMP|` joka viittaa kyseisen datapisteen luentahetkeen. Liiteessä A on esitetty vastaavaa rakennetta noudattava kokonainen viesti oikealla datalla. Liitteen viestissä aikaleimat noudattavat UNIX-aikaa, jossa nykyhetki esitetään millisekunteina vuoden 1970 alusta, eli aikaleima `1502954301198` vastaa UTC-0 aikaa `17.8.2017 7:18:21.198`. Liitteen viestin arvot ovat OPC-palvelimen simulointiominaisuuden tuottamia lukuja.

Vastaavan rakenteen CSV-muotoisen esityksen otsikkorivi olisi muotoa:

```
Servertimestamp,TagName,Value,Timestamp
```

CSV-esityksessä jokaista mittaustietuetta kohden viestiin kirjoitettaisiin uusi rivi, joka sisältäisi vastaavan datan. Myös tämä esitys kuvataan liitteessä A.

Vaihtoehto jäsentelemättömälle datalle on esimerkiksi laskennan tai analyysin tuloksena syntynyt jäsennelty data. Jäsennellyn datajoukon rakenne on tyypillisesti etukäteen tunnettu. Rakenne myös tyypillisesti pysyy samana, eikä vaihtele niin kuin jäsentelemätön datajoukko saattaa tehdä. Jäsennellyssä datassa esiintyy usein myös relaatioita, eli toisiinsa liittyviä tietoja. Näin ollen jäsennelty data talletetaan tyypillisesti taulukkomuotoiseen relaatiotietokantaan, jossa sitä voidaan käsitellä SQL-rajapintojen avulla. Myös tiedonsiirto relaatiotietokantaan ja siitä ulos on yleensä mahdollista SQL-palvelimen tarjoamien rajapintojen avulla. Vaihtoehtoisesti dataa voidaan kannasta viedä esimerkiksi CSV-tekstimuotoon ulkoista käsittelyä varten. Jäsenneltyä ja rakenteellista dataa käsitellään tyypillisesti automaatiopyramidin ylemmillä kerroksilla kuten MES ja ERP-järjestelmissä.

3.1.3 Datan määrä

Datan käsittelyltä ja siirrolta vaadittavaan kapasiteettiin vaikuttaa huomattavasti siirrettävän data-aineiston koko eli rinnakkaisten datapisteiden määrä ja tiedon luku- ja lähetystiheys. Tiheä sykli datan luennassa ja lähetyksessä kuormittaa datan lähdejärjestelmää sekä siirtoteitä, ja se on huomioitava myös pilvessä olevien resurssien skaalassa ja siitä aiheutuvissa kustannuksissa. Esimerkiksi tuhannella datapisteellä ohjelman 1 JSON-muotoisen viestin lähettäminen kerran sekunnissa vastaa karkeasti noin vajaan 10 gigatavun siirtomäärää vuorokaudessa, mikä saattaa jo ylittää heikkojen datalinkkien kuten mobiiliyhteyksien kapasiteetin.

3.1.4 Tiedonsiirron suunnat ja reitit

Riippuen järjestelmän vaatimuksista voi sen käsittämä tiedonsiirto olla yhden- tai kahdensuuntaista. Esimerkiksi tuotantolaitoksen dataa pilviympäristöön big data-mielessä arkistoiva sovellus tarvitsee vain yksisuuntaisen väylän reunatasolta alustatasolle, mutta prosessiin takaisinpäin vaikuttava sovellus vaatii kahdensuuntaisen liikennöinnin, jotta dataa saadaan myös yritys- ja alustatasoilta takaisin reunatasolle. Tiedonsiirron suunnat on huomioitava eri alijärjestelmien välisten rajapintojen, sekä tiedonsiirtoverkkojen suunnittelussa.

3.2 Tiedonsiirtotekniikat ja protokollat

Eri aikakausilta periytyviä legacy-tekniikoita sisältävän teollisen internetin järjestelmälle on ominaista laajempi protokollakirjo järjestelmän eri tasoilla verrattuna toimistoympäristön IT-järjestelmään. Teollisen internetin tiedonsiirto jaotellaan tässä karkeasti neljään osaan:

- Teollisuusprotokollat, joita esiintyy eri aikakausilta periytyvien tehdasjärjestelmien yhteydessä
- Yleiset IT-protokollat, joita esiintyy IT- ja pilvisovelluksissa sekä esimerkiksi MES- ja ERP-järjestelmissä.
- IoT- ja IIoT-protokollat
- Järjestelmän eri tasoja yhdistävät tunnelointi- ja verkkoratkaisut.

Seuraavissa alaluvuissa tehdään katsaus kunkin kategorian tärkeimpiin protokolliin, ja lopuksi muodostetaan käsitys miten eri protokollat sijoittuvat IIC:n teollisen internetin referenssimallia toteuttavan järjestelmän eri tasoille.

3.2.1 Teollisuusprotokollat

Tehdasympäristön protokollat koostuvat pitkälti perinteisistä ja moderneista teollisuusprotokollista, joita jo olemassa olevat teollisuusympäristön laitteet tukevat, ja joita hyödyntäen ne jo kommunikoivat. Tehdastason protokollat noudattavat jossain määrin automaatiopyramidin hierarkiaa: pyramidin alatasojen kommunikointi perustuu kenttäväyliin ja valmistajakohtaisiin protokolliin, ja näiden päällä toimivat järjestelmät hyödyntävät esimerkiksi OPC-standardia kommunikoinnissaan.

Kenttäväylät

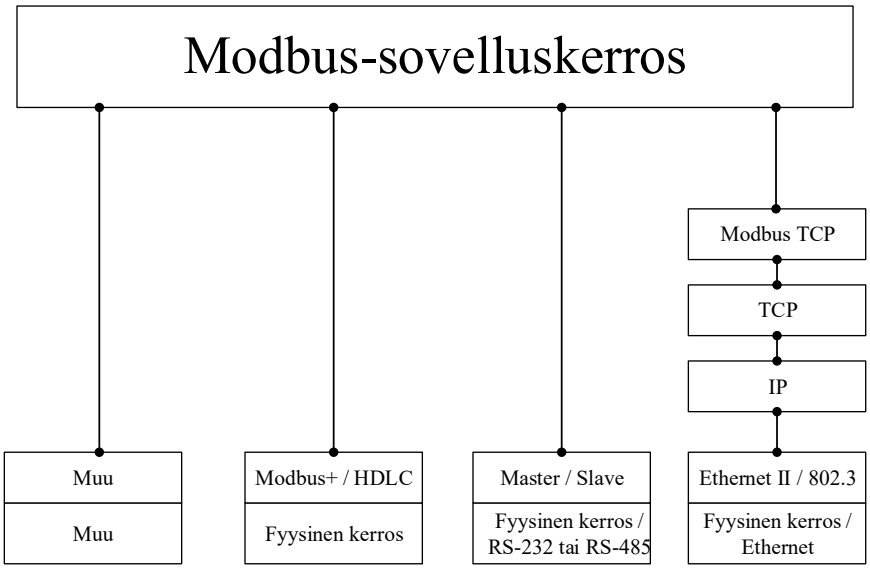
Kenttäväylä on kattotermi tyypillisesti teollisen automaatiojärjestelmän alatasoilla käytetyille tiedonsiirtoväylille ja protokollille. Kenttäväylät siirtävät tietoa ja ohjaussignaaleja prosessin toimilaitteiden ja mittauspisteiden sekä niitä ohjaavien logiikoiden välillä, ja niitä on kehitetty huomioimaan automaatiosovellusten erityisvaatimuksia esimerkiksi reaaliaikaisuuden suhteen. Verrattuna useimpiin tiedonsiirtoprotokolliin kenttäväylät ovat tyypillisesti synkronisia, eli väylällä liikennöidään kiinteän kokoisilla datapaketeilla, ja väylää käyttävät kommunikaation osapuolet on ajastettu keskuskellon avulla toimimaan keskenään samalla taajuudella. Tämä mahdollistaa esimerkiksi reaaliaikaiset vasteet hälytysten käsittelyä varten [3]. Kenttäväylä-käsite kattaa useita eri verkkotopologioita ja tiedonsiirtomenetelmiä.

Väylät voidaan jaotella esimerkiksi täysin analogisiin väyliin ja älykkäämpiin digitaalisiin väyliin. Analogisia väyliä ovat esimerkiksi paineilmapulsseihin ja 4-20mA virtaviesteihin perustuvat mittausjärjestelmät ja digitaalisiksi älykkäiksi väyliksi voidaan katsoa

esimerkiksi HART, Foundation Fieldbus, Profibus sekä Modbus. Lisäksi kenttäväylätasolla esiintyy valmistajakohtaisia suljettuja protokollia. [23]

Joitakin kenttäväyläprotokollia voidaan myös kapseloida Ethernet-väylän ja TCP/IP-protokollapinon sisään esimerkiksi perinteisen sarjaliikenneväylän sijaan, jolloin ratkaisua voidaan kutsua teolliseksi Ethernetiksi.

Esimerkiksi Modbus-standardi sisältää tuen kommunikointiin TCP/IP-protokollan päällä (kuva 14), jolloin protokolla pysyy samana mutta fyysinen kaapelointi ja siirtokerros voidaan toteuttaa Ethernet-tekniikoilla. Tämä mahdollistaa myös Modbus-protokollan käytön esimerkiksi langattoman wlan-yhteyden yli.



Kuva 14: Modbus kommunikaatiopinot. Mukaillen [24]

Teollinen Ethernet suoraviivaistaa teollisen internetin sovellusten arkkitehtuuria, sillä TCP/IP-pohjainen liikenne mahdollistaa suoran kommunikaation protokollatasolla esimerkiksi reunatason yhdyskäytävälle ilman muunnoksia eri protokollapinojen välillä. Myös fyysinen liitettävyys IT-pohjaisiin laitteisiin yksinkertaistuu standardin kaapeloinnin tai langattomien siirtoteiden myötä. Ethernet on kuitenkin oletuksena asynkroninen protokolla [3], eli väylää pitkin lähetettävä viesti voi koostua eri taajuudella lähetettävistä vaihtelevan kokoisista paketeista, jotka koostetaan viestiksi signaaliin merkittyjen aloitus- ja lopetusbittien avulla. Sen avulla ei siis voida vastata koviin reaaliaikavaatimuksiin kuten nopeisiin ohjauksiin ilman protokollaan tehtäviä muutoksia, jota voidaan nimittää kattotermillä reaaliaikaethernet.

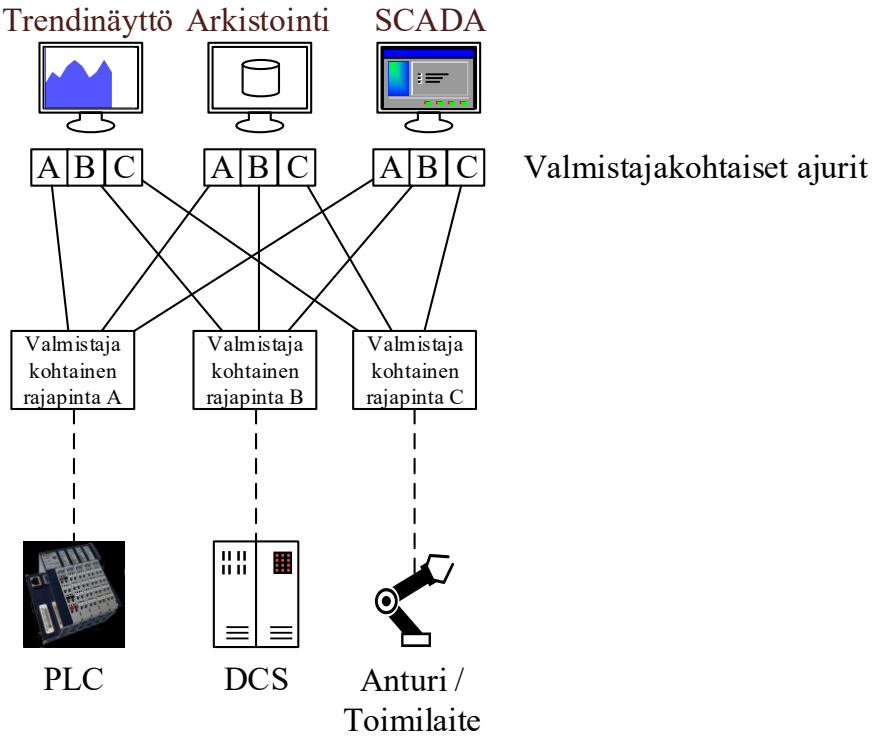
OPC

OPC (*Open Platform Communications,* alun perin *OLE for Process Control*) on teollisuuteen ja erityisesti automaatiosovelluksiin kehitetty tiedonsiirtostandardi, joka pyrkii tarjoamaan yhteisen rajapinnan eri valmistajien laitteiden väliseen kommunikaatioon.

OPC pyrkii vähentämään teollisuusympäristön valmistajariippuvuuksia ja mahdollistamaan esteettömän tiedonkulun automaatiojärjestelmän eri osien välille. OPC-määrittelyä kehittää ja ylläpitää OPC Foundation [25].

OPC-standardi koostuu useammasta määrittelystä, joista asiakas-palvelin-tyyppiseen prosessidatan reaaliaikaiseen siirtoon keskittyvät OPC DA (*Data Access*) ja OPC UA (*Unified Access*). OPC DA on selvästi standardin käytetyin osa, ja se on toteutettu lähes kaikkiin OPC-tuotteisiin [26]. Muita OPC standardin osia ovat esimerkiksi hälytysten käsittely (OPC AE), historiadatan käsittely (OPC HDA), XML-pohjainen rajapinta (OPC-XML-DA) sekä OPC.NET joka tarjoaa integraation Microsoftin .NET-ohjelmistorajapintoihin. Teollisen internetin tiedonsiirron näkökulmasta olennaisimpia näistä määrittelyistä ovat OPC DA ja OPC UA, sillä ne tarjoavat yhtenäisen protokollan ja tietomallin reaaliaikadatan käsittelyyn järjestelmän reunatasolla. Modernina protokollana OPC UA mahdollistaa myös integraatioita reunatason ulkopuolelle, esimerkiksi reunamallisiin sovelluskohteisiin, joissa osa järjestelmän älystä sijaitsee pilviympäristössä.

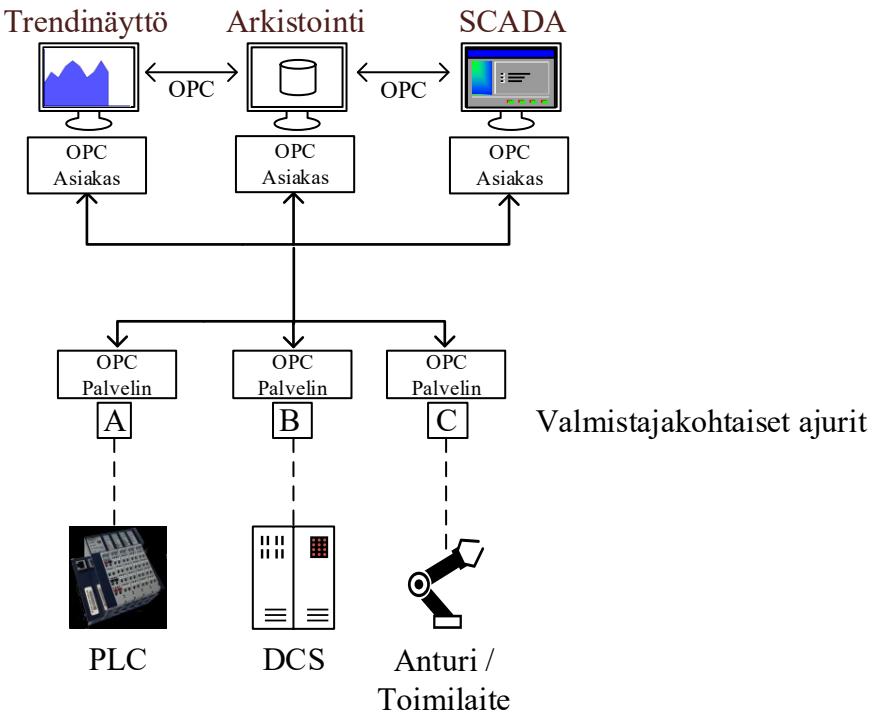
OPC-standardi määriteltiin ensimmäisen kerran vuonna 1996. Nykyisin alkuperäinen standardi tunnetaan erotuksena uudemmista nimellä OPC Classic. OPC Classic perustuu Microsoft Windowsin COM/DCOM-komponenteille (*Distributed Component Object Model*), ja on siten täysin Windows-sidonnainen [26]. Perusongelma, jonka ratkaisemiseen OPC on kehitetty, esitetään kuvassa 15.



Kuva 15: Valmistajariippuvuudet automaatiojärjestelmässä.

Kuvan 15 mukaisessa tilanteessa tehdasympäristön eri järjestelmät ja laitteet tarjoavat kukin oman, tyypillisesti valmistajakohtaiseen suljettuun koodiin perustuvan rajapinnan. Tämä johtaa tilanteeseen, jossa näiden järjestelmien dataa käyttäviin sovelluksiin joudutaan toteuttamaan jokaista valmistajakohtaista rajapintaa vastaavat ajuriohjelmistot sekä mahdolliset muunnokset eriävien dataformaattien välillä. Sovellusten toteutusvaiheessa lisääntyneen työmäärän lisäksi järjestelmän ylläpidettävyys heikkenee, sillä esimerkiksi rajapintoihin tehtävät muutokset ja päivitykset joudutaan toteuttamaan jokaiseen järjestelmään aina erikseen. Myöskään vaakasuuntaiseen integraatioon samantasoisten järjestelmien kuten kuvan esittämien arkistointi- ja trendinäyttösovellusten välillä ei ole tarjolla selkeää yhteistä rajapintaa, jonka kautta ne voisivat vaihtaa tietoja keskenään.

Kuvassa 16 esitetään OPC:n ratkaisumalli valmistajariippuvaiseen arkkitehtuuriin. Mallissa kullekin laitteelle tarjotaan OPC-rajapinta, joka piilottaa valmistajakohtaiset rajapinnat standardien OPC-rajapintojen taakse. Kapseloinnin voi hoitaa esimerkiksi järjestelmä itse tarjoamalla OPC-pääsyn dataansa, tai se voidaan toteuttaa laitteen rajapinta-ajurin ja OPC-palvelintoiminnallisuuden sisältävän middleware-ohjelmiston avulla.



Kuva 16: OPC-kommunikaation perusrakenne

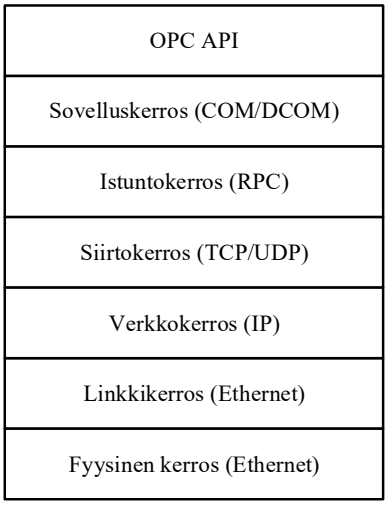
OPC:n avulla myös alempien tasojen dataa käyttävät ylemmän tason sovellukset voivat vaihtaa tietoja keskenään, sillä tietojen vaihto tapahtuu protokollaan sidotun tietomallin avulla. Tämä mahdollistaa myös vaakasuuntaiset integraatiot, esimerkiksi kuvan 15 trendinäyttösovellus voi noutaa tietoja arkistointisovellukselta OPC HDA-rajapinnan avulla. Yksinkertaistettu OPC DA-tietomalli yksittäiselle datapisteelle esitetään taulukossa 1.

Taulukko 1: OPC DA datapisteen tietomalli. Mukaillen [27].

Muuttuja	**Sisältö**
Arvo (Value)	Muuttujan arvo (useita perustietotyyppejä)
Aikaleima (Timestamp)	Kellonaika / Päivämäärä
Luotettavuus (Reliability)	Hyvä / Tuntematon / Huono

OPC DA pitää siis kirjaa pelkän mittausarvon lisäksi myös mittauksen ajankohdasta sekä siitä voidaanko lukemaan luottaa. Esimerkiksi tiedonsiirto-ongelmien esiintyessä OPC-palvelinohjelmisto voi asettaa luettujen datapisteiden luotettavuuden tilaan huono, jolloin tieto välittyy myös muille dataa käsitteleville sovelluksille.

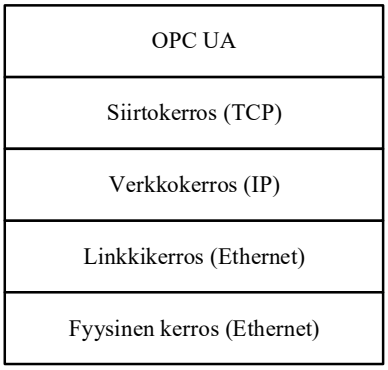
OPC Classicin rajoitteet liittyvät pitkälti sen sovelluskerroksen riippuvuuteen Windowsin COM/DCOM-kirjastoihin, jotka ovat suljetun lähdekoodin toteutuksia. Tämä estää standardin kanssa yhteensopivien sovellusten toteuttamisen avoimen lähdekoodin menetelmin tai esimerkiksi nykyisten OPC Classic-sovellusten kääntämisen muille kuin Windows-alustoille kuten Linuxille. OPC Classicin OSI-protokollapino esitetään kuvassa 17.



Kuva 17: OPC Classicin protokollapino. Mukaillen [28].

OPC Classic-protokollapinon istuntokerros perustuu RPC (*Remote Procedure Call*)-tekniikalle, jonka perustoiminnallisuuteen kuuluu dynaaminen porttien neuvottelu palvelinyhteyksiä muodostaessa [28]. Tämä tekee OPC Classicin käytöstä hajautetussa verkkoympäristössä vaikeaa tai jopa mahdotonta, sillä automaatioympäristön verkkoja tiukasti rajaavat palomuurit pitäisi käytännössä avata kokonaan, jotta dynaaminen porttien neuvottelu olisi mahdollinen yhteyden muodostustapa. OPC Classic ei myöskään sisällä protokollatason tietoturvamäärittelyitä, vaan se nojaa Windowsin DCOM-kirjastojen turvallisuuteen [28]. Esimerkiksi mahdolliset käyttäjämäärittelyt joudutaan OPC Classicin tapauksessa tekemään suoraan Windows-käyttöjärjestelmään ja määrittelyjen täytyy vastata toisiaan kaikilla asiakas-palvelinyhteydessä olevilla osapuolilla, jotta OPC Classic-yhteys voidaan muodostaa. Vaihtoehtoisesti käyttäjämäärittelyistä ja sen tarjoamasta tietoturvallisuudesta joudutaan luopumaan kokonaan.

OPC UA on uusin OPC-standardin sisältämä määrittely, joka irrottautuu OPC Classicin Windows-sidonnaisuudesta ja pyrkii tekemään protokollasta yleiskäyttöisemmän teollisuusympäristön tiedonsiirtotarpeisiin. OPC UA rakentuu täysin TCP/IP-protokollapinon päälle hyläten sekä istuntokerroksen RPC:n että sovelluskerroksen Windows-kirjastosidonnaisuuksineen. OPC UA:n OSI-protokollapino esitetään kuvassa 18.



Kuva 18: OPC UA protokollapino. Mukaillen [28].

Sovelluskerroksen ja COM/DCOM-komponenttien irrottaminen protokollan määrittelystä mahdollistaa OPC UA:n toteuttamisen myös avoimen lähdekoodin kirjastojen avulla, sekä alustoille, jotka eivät kykene tai joissa ei haluta käyttää Windowsia. Teollisen internetin näkökulmasta mielenkiintoisesti OPC UA mahdollistaa protokollan implementoinnin myös kevyisiin kenttälaitteisiin, jolloin huomattava osuus tehtaan järjestelmien välisestä liikennöinnistä voitaisiin rakentaa TCP/IP-pohjaisena. Tämän myötä myös integraatio pilvipohjaisiin sovelluksiin helpottuu.

OPC UA:n TCP/IP-pohjaisuuden myötä voidaan sen liikenne ohjata käyttämään tiettyä yksittäistä tietoliikenneporttia [29]. Tämä mahdollistaa OPC UA-yhteyksien reitittämisen palomuurien lävitse sillä avattavia portteja tarvitaan vain yksi.

Protokollaa voidaan käyttää myös OPC Classic-tiedonsiirron tunnelointiin, mikäli suora OPC DA-yhteys asiakkaan ja palvelimen välillä ei ole verkon rakenteesta tai palomuureista johtuen mahdollinen. Tässä mallissa OPC DA-asiakasohjelmisto lukee tarvittavia tietoja OPC DA-palvelimelta ja julkaisee ne uudestaan käyttäen OPC UA-palvelinrajapintaa. Näin tiedot voidaan lukea sopivasti konfiguroidun palomuurin lävitse OPC UA-asiakasohjelmistolla. Mikään ei protokollan puolesta estä myöskään OPC UA:n reitittämistä esimerkiksi julkisen internetin lävitse.

OPC Classiciin verrattuna OPC UA huomioi myös tietoturvan jo protokollan määrittelyssä. OPC UA:n tietoturvamäärittely jakaantuu karkeasti kahtia sovellus- ja kommunikaatiotasoille. Kommunikaatiotason turvallisuus määrittelee sovellustason käyttöön turvallisen ja salatun siirtotien, jota käytetään asiakkaan ja palvelimen väliseen liikennöintiin. Kommunikaatiotaso sisältää määrittelyt tiedonsiirron luotettavuudelle ja yhtenäisyydelle, sekä sovelluksen autentikoinnille, joka voi perustua salasanaan tai sertifikaatteihin.

Sovellustason tietoturvamäärittelyt pitävät sisällään käyttäjien hallinnan ja tunnistamisen [27].

3.2.2 IT-protokollat

Teollisen internetin järjestelmän alusta- ja sovellustasojen toiminnot sijaitsevat arkkitehtuurimallin mukaan tyypillisesti automaatioympäristön ulkopuolella ns. toimistoverkoissa tai pilviympäristössä. Näitä verkkoja yhdistävä tekijä on OSI-mallin alatasojen teknisten ratkaisuiden suurempi homogeenisyys automaatioverkkoihin verrattuna. Tuotannon kannalta ei-kriittisessä toimistoverkossa kommunikaatio perustuu pitkälti Ethernet-väylälle, TCP/IP-protokollapinolle ja näiden päällä liikennöiville IT-protokollille.

Tietokantayhteydet, MES ja ERP-integraatiot

Suoran kenttä- ja SCADA/HMI-tasojen tuottaman tiedon lisäksi teollisen internetin sovelluksessa saatetaan tarvita dataa myös yrityksen MES- tai ERP-järjestelmistä, kuten esimerkiksi näiden järjestelmien sisältämiä tilaus- ja tuotetietoja tai valmiiksi laskettuja tuotannon tunnuslukuja. Tyypillisesti tiedot on tallennettu relaatiotietokantaan, ja ne voidaan noutaa tavanomaisin SQL-kyselyin joko suoraan kantaan yhdistäen, tai käyttäen rajapintana esimerkiksi ODBC:tä. Myös järjestelmiin takaisinpäin kirjoittaminen esimerkiksi pilvisovelluksen analyysin palauttamista varten voidaan tehdä samaa reittiä.

HTTP ja REST

HTTP (*Hypertext Transport Protocol*) on tekstimuotoinen asiakas-palvelinmalliin perustuva protokolla, joka toimii pitkälti kaiken web-pohjaisen datasiirron perustana. HTTP sijoittuu OSI-mallin sovelluskerrokselle ja se toimii TCP/IP-pinon päällä. HTTP on alun perin kehitetty web-sivujen noutoon www-palvelimilta, mutta sen käyttö on laajentunut käsittämään myös monipuolisempaa tiedonsiirtoa ja M2M-liikennöintiä.

REST (*Representational State Transfer*) on arkkitehtuurimalli sovellusten ja järjestelmien välisten rajapintojen toteuttamiseen. Mallista on tullut erityisen suosittu modernien web-sovellusten aikana, sillä se pyrkii mahdollistamaan kevyen ja helposti toteutettavan tavan yhteensopimattomien sovellusten ja hajautettujen järjestelmien osien väliseen kommunikointiin. REST ei ole varsinainen standardi, eikä se arkkitehtuurimallina ota kantaa protokolliin tai toteutustapaan, mutta käytännön REST-mallia noudattavat API-rajapinnat toimivat tyypillisimmin HTTP-protokollan päällä. REST-mallin mukainen rajapintasuunnittelu on moderneissa web- ja M2M-sovelluksissa pitkälti syrjäyttänyt vanhemmat SOAP- ja XML-RPC-pohjaiset ratkaisut. [30]

3.2.3 IoT ja IIoT-protokollat

Tässä esiteltävät MQTT ja AMQP, sekä OPC UA Pub/Sub ovat kaikki jokseenkin moderneja protokollia, jotka kaikki on kehitetty M2M- ja IoT-tyyppisiin viestinvälityssovelluksiin. MQTT ja AMQP ovat yleisesti tuettuja sekä eri tyyppisissä reunatasolle sopivissa yhdyskäytäväsovelluksissa, että pilvialustojen IoT-tarjonnassa. OPC UA Pub/Sub on OPC UA-standardin melko tuore laajennus, joka mahdollistaa protokollan viennin matalan prosessointikyvyn sensoreille ja toimilaitteille.

MQTT

MQTT (*Message Queue Telemetry Transport*) on M2M (*Machine-To-Machine*) ja IoT-käyttöön suunniteltu kevyt tilaaja-julkaisijamalliin (*publish-subscribe*) perustuva binäärimuotoinen tiedonsiirtoprotokolla. Protokollan version 3.1.1 määrittely on julkaistu avoimena ISO-standardina (ISO/IEC 20922) vuonna 2016. [31] MQTT on suunniteltu ajettavaksi kevyillä laskentaresursseilla ja ympäristöissä, joissa virrankulutus on rajoittava tekijä. Protokolla voidaan keveytensä ansiosta integroida osaksi useita eri tyyppisiä laitteita tai sovelluksia. Teollisessa käytössä mielenkiintoisia sovelluksia ovat esimerkiksi MQTT-protokollaa tukevat ohjelmoitavat logiikat, jotka kykenevät suoraan lähettämään viestejä esimerkiksi pilvisovellukselle. MQTT on perusrakenteeltaan yleiskäyttöinen, ja se on agnostinen viestin sisällön suhteen, eli viestin hyötykuorma voi olla esimerkiksi JSON- tai XML-muodoissa, tai esimerkiksi sovelluskohtaisessa binäärimuodossa [32].

MQTT pyrkii korkeaan tiedonsiirron hyötysuhteeseen pitämällä datapaketin sisältämän hyötykuorman mahdollisimman suurena verrattuna protokollan vaatimiin kehyksiin. HTTP-protokollaan verrattuna MQTT on kevyempi vaihtoehto esimerkiksi IoT-laitteen telemetriatietojen siirtämiseen verkon yli, ja se soveltuu suurten data-aineistojen reaaliaikaiseen tiedonsiirtoon hyvän hyötysuhteensa ansiosta. MQTT toimii suoraan TCP/IP-protokollapinon päällä, eikä sen määrittely ota kantaa millä ohjelmointikielellä se toteutetaan [31]. Protokollan viestiliikenne voidaan salata hyödyntäen TLS-standardia. Avoimia MQTT-kirjastoja eri kielillä on saatavissa useille eri alustoille.

Tilaaja-tuottajamallissa viestit välitetään erillisen välittäjän (*broker*) avulla, joka tyypillisesti vastaa palvelin-asiakasmallin palvelinta. Välittäjä pitää kirjaa sille määritetyistä aiheista (*topic*), joihin tuottajat (*publisher*) voivat julkaista viestejä ja joihin tilaajat (*subscriber*) voivat rekisteröityä saadakseen tiedon uusista viesteistä [31]. Yksittäiselle välittäjälle voidaan julkaista esimerkiksi tuhannen anturin mittaustietoja, mutta yksittäiset tilaajat voidaan rekisteröidä lukemaan ainoastaan niille olennaisten anturien tietoja. Tilaaja-tuottajamalli vähentää täten tarpeetonta liikennöintiä tarjoten juuri tarvittavat viestit oikeille osapuolille. Viestinvälitys on myös asynkronista, eivätkä tilaajat tai tuottajat ole tietoisia toisistaan. Tilaaja voi siis noutaa viestejä välittäjältä tarpeen mukaisesti, eikä viestinvälitys tuottajalta tilaajalle ole riippuvaista siitä kykenevätkö osapuolet viestimään samanaikaisesti.

AMQP

AMQP (*Advanced Message Queuing Protocol*) on MQTT:n lisäksi toinen IoT- ja M2M-sovelluksiin sopiva tilaaja- julkaisijamalliin perustuva tiedonsiirtoprotokolla. AMQP:n määrittely ei ota kantaa välittäjäpalvelun (*broker*) toteutukseen, vaan määrittelee ainoastaan mekanismit sovellusten väliseen viestinvaihtoon. Tämä jättää joustavuutta protokollan toteuttamiselle eri tyyppisille alustoille ja laitteille, mutta saattaa monimutkaistaa toteutusvaihetta, ja tehdä asiakassovelluksista liian raskaita rajoittuneille laitealustoille. [32]

AMQP tarjoaa joitakin piirteitä, jotka tekevät siitä sopivan esimerkiksi IoT-sovelluksen taustapalvelun viestinvälityksen protokollaksi. Näitä ovat esimerkiksi TLS-salaukseen perustuva turvallisuus, mahdollisuus kahdensuuntaiseen viestintään, useiden loogisten viestintäsessioiden ylläpito saman yhteyden päällä, sekä eri alustojen välillä siirrettävissä oleva toteutus, joka sisältää myös datan esitystavan. Adryan ja Konigseder esittävät AMQP:n soveltuvan parhaiten useita eri viestinvälityssemantiikkoja käyttäjän monimutkaisen IoT-taustapalvelun protokollaksi, sillä se tarjoaa paljon vapauksia protokollan toteuttamisen suhteen. AMQP voi toimia myös sovelluskohtaisen viestintäkirjaston pohjana. [32]

OPC UA PubSub

OPC UA Publish-Subscribe on OPC UA-standardin melko tuore laajennus, jonka tavoitteena on mahdollistaa OPC UA:n hyödyntäminen myös rajoittuneilla laitteistoalustoilla ja muissa ympäristöissä, joissa täyden OPC UA-palvelimen toteuttaminen ei ole mahdollista. [33]

OPC UA PubSub käyttää tiedonvälityksen protokollana joko MQTT:a tai AMQP:aa, jotka molemmat itsessään tarjoavat määrittelyn tilaaja- julkaisijamallin viestinnälle. OPC UA PubSub hyödyntää siirtokerroksella UDP:tä TCP:n sijaan, mikä vähentää viestinvälityksen viiveitä. UDP-pohjaisuus mahdollistaa IP-protokollan multicast-mekanismien avulla tiedonvälityksen suoraan esimerkiksi tietylle PubSub-julkaisijalle rekisteröityneille tilaajille. Näin yksittäiseltä julkaisijalta vaadittavat laskentaresurssit saadaan minimoitua, kun viestiliikenteen välitystä tehdään alempien protokollien mekanismien avustuksella. [34]

OPC UA PubSubin käytön suurimmat edut saavutettaneen tehtaan lattiatasolla, kuten älykkäiden mitta- ja toimilaitteiden viestintäprotokollana. Protokolla mahdollistaa niiden viestinnän suoraan esimerkiksi pilvisovellukselle WAN-verkkojen (*Wide Area Network*) tai internetin yli, sillä se sisältää suoraan esimerkiksi MQTT:n tai AMQP:n tietoturvaominaisuudet ja IP-pohjaisuuden. Protokollan käytöstä saataneen etuja myös, mikäli lattiatason yläpuolella olevan järjestelmät ovat OPC UA-pohjaisia, jolloin käytännössä kaikki tehtaan tietoliikenne perustuu samaan standardiin.

3.2.4 Tunneloinnit ja verkkoratkaisut

Teollisen internetin järjestelmä on jo lähtökohtaisesti hajautettu järjestelmä, jonka toiminnan mahdollistava tietoliikenne jakautuu useisiin fyysisiin ja loogisiin verkkoihin. Etenkin reunatasolla, johon järjestelmän tiedonkeruutoiminnot tyypillisesti sijoittuvat, verkkoympäristöt ovat tyypillisesti tiukasti eristettyjä julkisesta internetistä sekä fyysisesti että palomuurein. Tässä kappaleessa selvitetään tunnelointeja ja verkkoratkaisuja, joilla tietoa voidaan välittää suljettujen verkkojen välillä tietoturvallisesti.

VPN-tunneloinnit, virtuaaliset verkot

Kuvan 11 kolmitasoisen IIoT-mallin reuna- ja alustatasoja yhdistää pääsyverkko, joka mahdollistaa reitityksen reunatason yhdyskäytävän ja pilvessä sijaitsevan palvelualustan välillä. Käytännön pääsyverkkona voi toimia esimerkiksi yrityksen tuotantoverkot julkisesta internetistä eristävä DMZ (*Demilitarized Zone*) -alue, jolta voidaan tehdä tarvittavat palomuuriavaukset julkisen internetin suuntaan, ja johon voidaan reitittää sisempien verkkojen liikennettä.

Pääsyverkon ja alustatason välinen yhteys voi olla joko osa yrityksen sisäistä verkkoa tai julkisen internetin yli reitittyvä tunnelointiratkaisu eli VPN (*Virtual Private Network)* [18]. VPN eli yksityinen virtuaaliverkko muodostetaan ohjelmallisesti kahden tietokoneen tai muun verkkolaitteen välille. VPN-protokollia ja toteutuksia on useita, mutta tyypillisiä ominaispiirteitä VPN-tunneleille ovat reitittyminen julkisen internetin yli ja vahva salaus. VPN voidaan toteuttaa OSI-mallin siirto-, verkko- ja sovellustasoilla riippuen järjestelmän tarpeista. VPN-yhteyksiä voidaan käyttää sekä yksittäisten työasemien kytkemiseen etäkohteessa sijaitsevaan sisäverkkoon, tai useampien sisäverkkojen kytkemiseen näennäisesti samaksi verkoksi [35]. Eräs käyttötapaus VPN:n käytölle teollisen internetin järjestelmässä on kytkeä reunatason yhdyskäytävä VPN:n yli pilvessä sijaitsevaan virtuaaliverkkoon, jolloin se voi kommunikoida kaikkien pilvessä sijaitsevien virtuaaliverkon jäsenten kesken ilman monitasoisia palomuuri- tai reitityskonfiguraatioita. VPN:n salausominaisuuksien ansiosta liikenne on myös lähtökohtaisesti turvallista, vaikka kaikki tunnelin sisällä käytetyt protokollat eivät itsessään olisikaan salattuja.

Koska VPN tyypillisesti ohjataan julkisen internetin yli, ei sen tarkkaa reititystä voida määritellä, eikä se välttämättä pysy muuttumattomana. Tämä voi aiheuttaa ongelmia esimerkiksi korkeiden reaaliaikavaatimusten toteuttamisen suhteen.

Dedikoidut väylät

Vaihtoehtona VPN:n käytölle pääsyverkon ja alustatason palveluiden välille pilvipalveluiden toimittaja voi yhdessä teleoperaattorien kanssa tarjota myös yksityistä tiedonsiirtoväylää. Tällöin asiakasyrityksen ja pilvioperaattorin verkkojen välille rakennetaan jul-

kisesta internetistä eristetty väylä, jonka kapasiteetti ja reititys on asiakasyrityksen päätettävissä ja täysin sen käytettävissä. Esimerkiksi Microsoft ja Amazon Web Services tarjoavat tämänkaltaista palvelua nimillä Expressroute ja Direct Connect.

3.2.5 Tiedonsiirtotekniikoiden jakautuminen eri tasoille

Perinteistä automaatiopyramidia noudattavassa järjestelmässä eri tiedonsiirtoprotokollien- ja tekniikoiden jakautuminen järjestelmän eri kerroksille on jokseenkin suoraviivaista ja hierarkkista. Pyramidin alatasoilla esiintyy kenttäväyliä, keskitasoilla valmistajakohtaisia protokollia sekä OPC Classicia, ja pyramidin ylätasoilla kuten MES- ja ERP-järjestelmissä IT-protokollia.

IIC:n teollisen internetin järjestelmän referenssimallin määrittelemillä kolmella kerroksella, reuna- alusta- ja pilvitasoilla protokollien jakautuminen ei välttämättä ole näin suoraviivaista, vaan esimerkiksi erilaisten protokollamuunnoksia tekevien yhdyskäytäväohjelmistojen avulla tyypillisiä IT-protokollia, kuten http:ta tai IoT-protokollia kuten MQTT:a, voidaan hyödyntää jo reunatason sisäisessä viestiliikenteessä. Toisaalta eri tunnelointiratkaisut tehtaan ja pilven, tai reuna- ja alustatasojen välillä mahdollistavat joidenkin teollisuusprotokollien, kuten Modbus/TCP:n viennin sitä tukevalle pilvisovellukselle asti, mikäli tämä olisi hyödyllistä järjestelmän toiminnan kannalta

Erityisesti OPC UA:n odotettavissa oleva yleistyminen tehdasjärjestelmissä niiden uusiutumisen myötä tasoittaa teollisen internetin järjestelmän protokollahierarkiaa. OPC UA PubSub-laajennuksen myötä lähes koko tehtaan viestiliikenne aina pilveen asti olisi mahdollista toteuttaa vain yhden protokollaperheen avulla. Muutos ei kuitenkaan ole odotettavissa hetkessä, esimerkiksi taloudellisten syiden takia. Näin ollen menetelmät nykyisten ns. legacy-järjestelmien ja modernien sovellusten välisen viestinnän mahdollistamiseen pysyvät tarpeellisina jatkossakin [36].

3.3 Ohjelmistot ja alustat

Tässä kappaleessa esitellään kappaleen 4. esimerkkiarkkitehtuurien osina toimivat ohjelmistot ja alustat.

3.3.1 Kaupallinen julkinen pilvialusta: Microsoft Azure

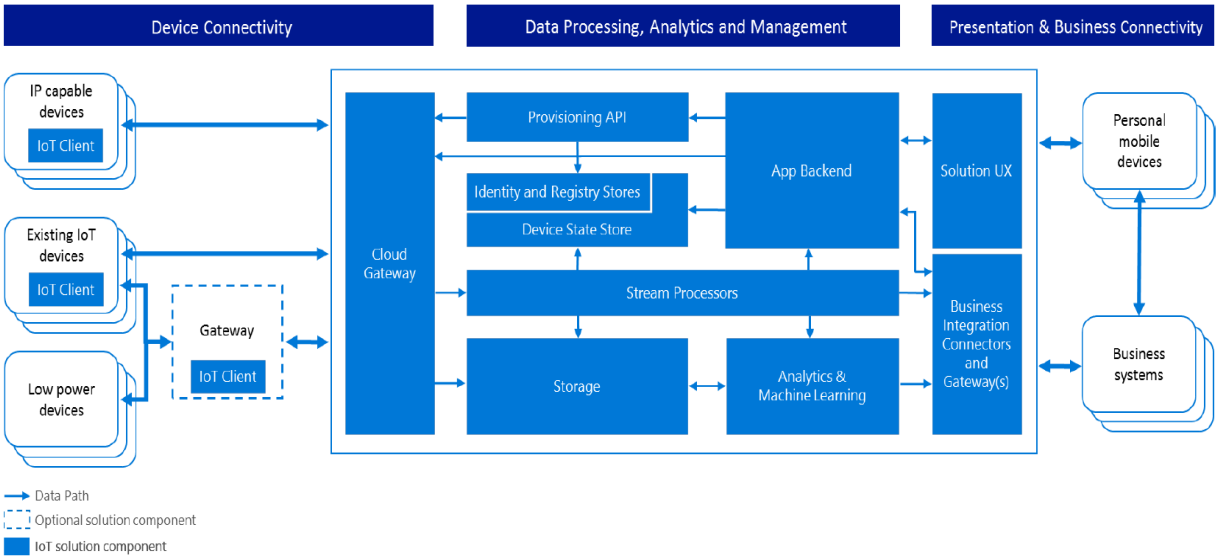
Azure on Microsoftin vuonna 2010 nimellä Windows Azure lanseeraama ensisijaisesti yrityskäyttöön suunnattu julkinen pilvipalvelu. Windows-etuliitteestä palvelun nimessä Microsoft luopui vuonna 2014, tarkoituksenaan alleviivata yrityksen keskittymistä palveluiden tarjoamiseen pelkkien ohjelmistojen sijasta [37]. Palveluliiketoiminnan osana Azureen on sen olemassaolon aikana rakennettu vahva tuki myös Microsoftin ulkopuolisille ja avoimen lähdekoodin tekniikoille kuten Linuxille.

Azure pyrkii olemaan yleiskäyttöinen pilvialusta, joka tarjoaa laajan valikoiman erityyppisiä pilvipalveluita eri palvelutasoilla ja erilaisiin asiakastarpeisiin. Maantieteellisesti Azure hajautuu 50 alueeseen (*region*), joka on Microsoftin mukaan enemmän kuin millään muulla pilvitarjoajalla [38]. Yksittäinen Azure-alue koostuu useammasta paikallisesti yhteen liitetystä datakeskuksesta. Azure-kehittäjän kannalta maantieteellinen hajautus on läpinäkyvää, ja esimerkiksi maailmanlaajuisesti asiakkaita palveleva sovellus voidaan hajauttaa toimimaan useilla eri Azure-alueilla samanaikaisesti.

Microsoft jaottelee Azuren palvelutarjonnan kolmeentoista kategoriaan, jotka sisältävät yhteensä satoja eri tyyppisiä palveluita [39]. Tarjonta kattaa tyypillisiä IaaS-resursseja, kuten virtuaalikoneita ja -verkkoja, tallennuskapasiteettia, sekä lukuisia PaaS-ratkaisuja esimerkiksi web-sovellusten ylläpitoon ja hallintaan. Alusta itse käyttää palveluista nimeä resurssi (*resource*). Azuren ensisijainen käyttöliittymä on selaimen kautta käytettävä Azure portal, jonka kautta eri tyyppisiä resursseja voidaan hallinnoida yhtenäisen käyttöliittymän kautta. Azure tarjoaa resurssien hallintaan ja konfigurointiin myös tekstipohjaisen REST-rajapinnan, jonka avulla resurssien hallinnointia voi esimerkiksi automatisoida.

Teollisen internetin sovellusten kehittämisen kannalta erityisen relevantteja Azuren palvelukategorioita ovat esimerkiksi analytiikka (*analytics*), tekoäly ja koneoppiminen (*AI + Machine Learning*) ja esineiden internet (*Internet of Things*). Nämä kategoriat sisältävät PaaS-tason ratkaisuja laiterajapinnan ja viestiliikenteen hallintaan ja erityisesti datan käsittelyyn Azuren sisällä.

Microsoft tarjoaa IoT referenssiarkkitehtuurin, joka sisältää yleisiä suunnitteluperiaatteita Azureen pohjautuvan IoT-sovelluksen rakentamiseen. Arkkitehtuuri esitetään kuvassa 19:



Kuva 19: Microsoftin IoT referenssiarkkitehtuuri [40]

Referenssiarkkitehtuurissaan Microsoft pilkkoo sovelluksen pienempiin loogisiin osioihin, joihin tarjotaan Azuresta löytyviä ratkaisuvaihtoehtoja. Ylimmällä tasolla sovellus on jaettu laiterajapintaan (*Device connectivity*), datan käsittelyyn, hallinnointiin ja analytiikkaan (*Data Processing, Analytics and Management*) sekä esitys- ja integraatiokerrokseen (*Presentation & Business Connectivity*) [40]. Microsoftin kolmiosainen jaottelu vastaa hyvin pitkälle kuvan 11 mukaista IIC:n IIoT-referenssimallia, jossa laiterajapintaa vastaa kenttätaso, datan käsittelyä, hallintaa ja analytiikkaa alustataso, ja esitys- ja integraatiokerrosta yritystaso. Koska arkkitehtuurikaavion osat esittävät sovelluksen loogisia osia, voi yksi konkreettinen Azure-resurssi toimia useammassa kaavion esittämässä roolissa ja toisaalta yksittäinen toiminnallisuus voi olla useamman Azure-resurssin yhdessä toteuttama.

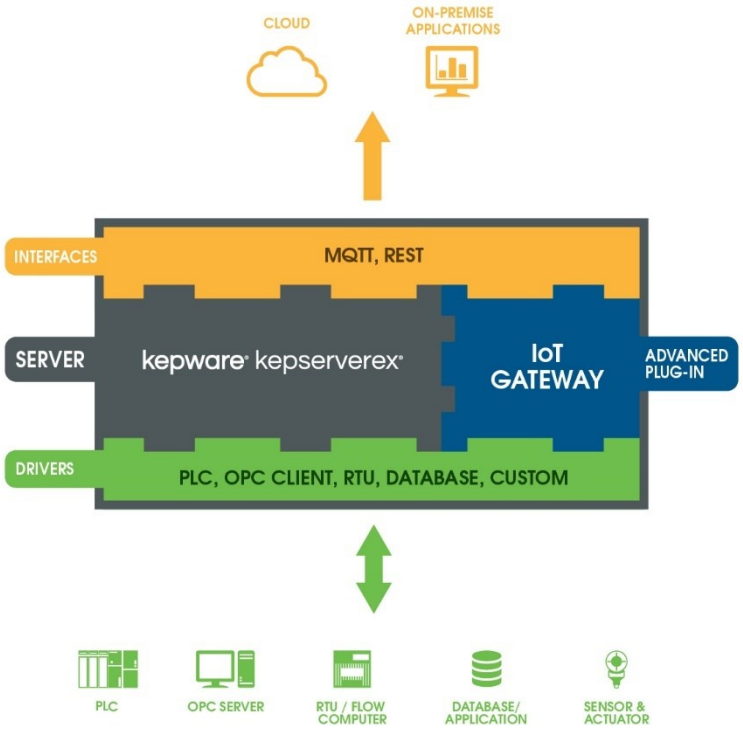
Laiterajapintatasolla eri tyyppisiä IoT-sovelluksen tietolähteitä, eli Azure-termistöllä laitteita (*device*) on luokiteltu niiden IP-kykeneväisyyden mukaan. Suoraan IP-verkkoihin yhdistämiseen kykenevät laitteet voidaan arkkitehtuurin mukaan kytkeä sellaisenaan pilviyhdyskäytävään (*Cloud Gateway*), kun taas vähemmän älyä sisältävät laitteet esitetään yhdistettäviksi erillisen yhdyskäytävän kautta. Microsoft ei tarjoa varsinaista kenttätason yhdyskäytävää, vaan toteaa sen arkkitehtuurissa tarkoittavan kolmannen osapuolen tarjoamaa sovellusta tai laitteistoa, joka mahdollistaa yhteyden pilven ja laitteiden välillä. Paikallinen yhdyskäytävä voi olla jokin valmis tuote, joka tukee Microsoftin tarjoamia protokollia ja rajapintoja, tai se voi olla Microsoftin tarjoamaan valmiiseen koodiin perustuva tarpeeseen kehitettävä sovellusratkaisu. Pilvitasolla yhdyskäytävän roolissa Azuressa toimii IoT Hub, joka tarjoaa rajapinnan datan lähettämiseen ja viestien käsittelyyn, sekä työkaluja etälaitteiden hallinnointiin.

Datan käsittelytaso koostuu täysin Microsoftin tarjoamista Azure-komponenteista, ja sen voidaankin katsoa sen olevan koko arkkitehtuurin ja Microsoftin palvelutarjonnan ydin. Taso sisältää palveluita laitteiden ja niiden identiteettien hallintaan, tietovirtojen hallintaan, tiedon tallennukseen ja sen analysointiin sekä koneoppimiseen. Sovelluksen mahdollinen käyttöliittymä, eli Microsoftin termein *UX*, user experience, esitetään rakennettavaksi Azuren web-sovelluspalvelun (*Web App Service*) varaan. Microsoft korostaa arkkitehtuurimallissaan, että tyypillisen IoT-sovelluksen toimintalogiikka (*Business logic*) jakaantuu useammille eri komponenteille, jotka yhdessä tuottavat sovelluksen tarjoamia palveluita yksittäisen monoliittisen back-end palvelun sijaan. Tätä voidaan nimittää yleisemmin mikropalveluarkkitehtuuriksi (*micro-services*).

Esitys- ja integraatiokerros tarjoaa työkaluja datan visualisointiin ja sen viemiseen toisiin järjestelmiin, jotka voivat Microsoftin mukaan sijaita joko pilvessä tai yrityksen omilla palvelimilla. Microsoftin ensisijainen visualisointityökalu on Power BI, jossa dataa voidaan visualisoida kausiluontoisesti historiadataan perustuen tai reaaliaikaisesti. Integraatioita ulkopuolisiin järjestelmiin tarjotaan erilaisten valmiiden ja konfiguroitavissa olevien API:en kautta.

3.3.2 Reunatason kommunikaatiorajapinta KepServerEX

KepServerEX on yhdysvaltalaisen PTC:n omistaman Kepwaren ohjelmistotuote, joka keskittyy tarjoamaan yleiskäyttöisen kommunikaatiorajapinnan teollisten järjestelmien väliseen kommunikointiin. KepServerEX tukee suurta määrää standardeja ja toimittajakohtaisia protokollia, ja kykenee myös itse toimimaan OPC DA ja UA-palvelimena, jolloin myös sen lukemia tietoja voidaan edelleen lukea toisesta OPC-sovelluksesta käsin. KepServerEX kykenee siis esimerkiksi toimimaan tulkkina suljettua protokollaa käyttävän logiikan ja OPC-yhteyttä tukevan SCADA-järjestelmän välillä. [41]



Kuva 20: KEPServerEX-kommunikaatiorajapinta. [42]

Teollisen internetin sovelluksia varten kehitetty KepServerEX-liitännäinen IoT Gateway sisältää MQTT-tuen, sekä palvelin-asiakastoiminnallisuudet sisältävän REST-rajapinnan. Lisäksi IoT Gateway sisältää suljetun lähdekoodin always on-protokollaan perustuvan integraation PTC:n ThingWorx-alustalle. IoT Gateway mahdollistaa siis liitynnät ja tiedonsiirron myös teollisuusympäristön ulkopuolisiin IT-sovelluksiin ilman että niihin tarvitsee rakentaa tukea teollisuusprotokollille. Kaikki IoT Gatewayn protokollat tukevat myös kahdensuuntaista liikennettä, eli esimerkiksi pilven välityksellä toimiva web-sovellus voi kirjoittaa arvoja KepserverEX:n REST-rajapintaan, jolloin ne välitetään suoraan esimerkiksi logiikan muistipaikkoihin.

4. CASE-ARKKITEHTUURIT

Tässä kappaleessa käydään läpi kolme teollisen internetin esimerkkisovellusarkkitehtuuria ja pyritään löytämään niiden toteutuksista olennaiset elementit. Arkkitehtuureja vastaavat järjestelmät on toimitettu niitä tuotantoympäristöissään hyödyntäville asiakasyrityksille.

Arkkitehtuureista ensimmäinen kuvaa suppeamman sovelluksen, joka käsittää tiedonkeruun ja tiedonsiirron reunatasolta alustatasolle sekä varsinaisen alustatasolla SaaS-komponenteilla rakennettavan sovelluksen. Kaksi jälkimmäistä kuvaavat tiedonsiirtoa ja varastointia reunatasolta alustatasolla yleiskäyttöisessä tiedostoformaatissa tai standardirajapintojen taakse siten että dataa voidaan noutaa eri tyyppisistä sovelluksista käsin. Nämä järjestelmät hyödyntävät PaaS- ja IaaS-tasoisia pilviresursseja.

Kustakin arkkitehtuurista käydään läpi toteutuksen lähtökohdat eli miksi ja mihin tarkoitukseen se on suunniteltu ja niiden tekniset toteutukset käydään läpi tarpeellisella tarkkuudella, jotta nähdään miten ne eroavat toisistaan. Koska arkkitehtuureja vastaavat järjestelmät on niiden hankkijayrityksillä käytössä todelliseen tuotantoympäristöön liitettyinä, on käytettävissä myös näkemyksiä niiden soveltuvuudesta käyttötarkoituksiinsa. Näistä kokemuksista ja teknisistä mahdollisuuksista jäsennellään kunkin arkkitehtuurin vahvuudet ja heikkoudet, sekä arvioidaan, olisivatko niissä esiintyvät ratkaisut mahdollisesti yleiskäyttöisiä myös muissa teollisuuden internetin sovelluksissa.

4.1 Case 1: SaaS-pohjainen tuotannon etenemisen seuranta

SaaS-pohjaisen tuotannon seurannan visualisointiratkaisun tilannut asiakasyritys A on kiinnostunut digitalisoimaan tuotantoaan ja haluaa toteuttaa pilottihankkeen kerätäkseen käytännön kokemuksia pilviteknologioista soveltamalla niitä omaan tuotantoprosessiinsa. Tuotantolaitoksella on ilmennyt tarve erään tuotantoprosessin osan reaaliaikaiselle seurannalle, ja tämä tarve halutaan ratkaista nimenomaisesti pilvitekniikoita hyödyntäen, vaikka vastaavia ratkaisuja onkin aiemmin rakennettu paikallisesti toimivilla tekniikoilla. Pilvipohjaisesti toteutettua näyttöä voidaan seurata myös tuotantotilan ulkopuolella ilman etäyhteyksiä ja se nähdään yksinkertaisempana ylläpidettävänä, varsinkin jos ratkaisu laajennetaan yhtenäiseksi useille eri tuotantolinjoille tai laitoksille. Tuotannon tilaa seuraava näyttö on myös asiakkaan näkökulmasta matalan riskin kohde, jonka epäonnistuminen ei aiheuta merkittäviä ongelmia, eikä sovellus vaadi mittavia investointeja tai muutoksia tuotantolaitokselle.

4.1.1 Toteutus

Asiakkaan A jo olemassa oleva IT-infrastruktuuri on pitkälti Microsoft-sidonnaista aina automaatioverkon prosessiasemista lähtien, joten Microsoftin tarjoamat pilvityökalut ovat luonnollinen valinta toteutuksen pohjaksi. Asiakkaan näkökulmasta tärkeä ominaisuus on esimerkiksi integroituminen AD-pohjaiseen autentikointiin, jolloin nykyiset käyttäjät voidaan liittää helposti sovelluksen käyttäjiksi. Microsoftin BI-työkalu Power BI tarjoaa mahdollisuuden reaaliaikaisten näyttöjen (*real time dashboard*) luomiseen [43], joten lopullinen näyttö on arkkitehtuurissa toteutettu sen avulla.

Sovelluksen kohteena oleva prosessi on kappaletavaraa tuottava linja, ja näytöllä esitettävät muuttujat koostuvat tuotannon reaaliaikaisista toteutumatiedoista, MES-tietokannan tuotantotilauksen tiedoista sekä ERP-tietokannan tuotetiedoista. Toteumatiedot luetaan suoraan tuotantoa ohjaavilta ja seuraavilta logiikoilta ja muu data kyseisten järjestelmien SQL-tietokannoista. IIC:n referenssimallin ohjaukset- ja säätöalueen kommunikaatiotoimintoa vastaavia toiminnallisuuksia, eli käytännössä tietolähteiden lukemista toteutuksessa suorittaa tehdasverkossa sijaitsevalla virtuaalipalvelimella ajettava KepServerEX, jonka Siemens TCP/IP-ajuri mahdollistaa logiikoiden luennan tehdasverkon yli, ja ODBC-asiakasajuri tietokantojen lukemisen ODBC-rajapinnan kautta. Ohjelmisto käsittelee tietoja OPC-tietomallin avulla, joten sekä logiikoiden muuttujat että tietokannan solut voidaan esittää yhtenäisessä muodossa, vaikka ne lähdejärjestelmissä ovatkin täysin erilaisissa muodoissa: logiikan muistipaikkojen arvoina sekä tietokantataulujen riveinä.

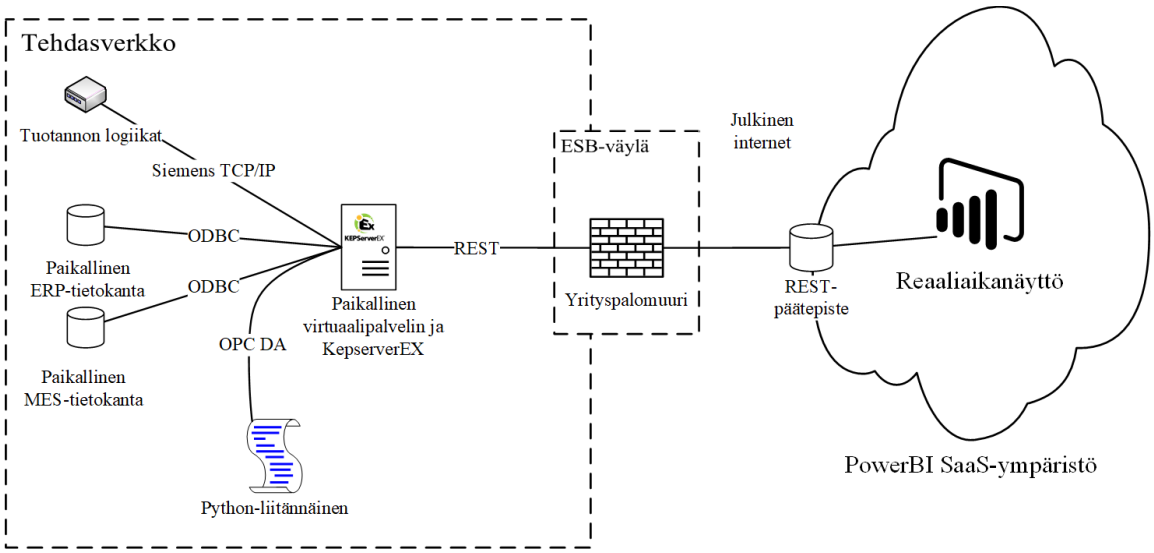
Kuten kappaleessa 2.2.1 kuvataan, SaaS-palvelumallissa tarjotun pilvisovelluksen toiminnallisuus on täysin sen tarjoajan määriteltävissä. PowerBI:n SaaS-ympäristön tarjoama reaaliaikanäyttötoiminnallisuus ei itsessään riitä täyttämään asiakasyrityksen järjestelmävaatimuksia, eikä ympäristöön ole mahdollista toteuttaa sovelluskohtaisia mukautuksia. Kehitettävän ratkaisun kannalta selkeitä ympäristön asettamia rajoitteita on kaksi:

- Yksittäisten datapisteiden sisältämän informaation yhdistely ja ryhmittely. Reaaliaikanäyttöä varten datan tulisi saapua ympäristöön juuri siinä muodossa kuin se halutaan esittää. Esimerkiksi kahdesta OPC-datapisteestä, jotka kuvaavat valmistettavan kappaleen leveyttä ja pituutta ei ympäristön sisällä voida suoraan yhdistellä kappaleen ulottuvuuksia kuvaavaa yksittäistä tietuetta.
- Tiedolle tehtävä pitkäaikainen laskenta ja kumulointi. Tuotannon edistymistä kuvaavia tunnuslukuja tulisi voida kumuloida jatkuvasti näytölle, mutta ympäristö ei sisällä mahdollisuutta historiadatan käsittelyyn ja summaamiseen. Laskurit pitäisi pystyä myös nollaamaan tilauksen vaihtuessa, mutta ympäristö ei tarjoa työkaluja tämänkaltaisen logiikan kirjoittamiseen.

Ratkaisuksi ympäristön aiheuttamiin rajoitteisiin järjestelmään on toteutettu Python-kielinen lisäosa, joka muokkaa dataa jo reunatasolla siihen muotoon kuin se alustatasolle

halutaan. Lisäosa lukee raakadataa KepserverEX:n OPC DA-rajapinnasta paikallisesti samalla palvelimella, ja kirjoittaa muokatun datan sille varattuihin muistipaikkoihin samaa rajapintaa käyttäen. Lisäosa myös laskee näytölle tarvittavien laskurien arvoja, ja pitää kirjaa niiden muutoksista ja huolehtii niiden oikea-aikaisista nollauksista.

Järjestelmän komponentit, käytetyt protokollat sekä tiedonsiirtomenetelmät esitetään kuvassa 21. Verratessa toteutuksen arkkitehtuuria IIC:n referenssimalliin (Kuva 11), voidaan huomata, että varsinkin reuna- ja alustatasojen osalta se on hyvin samankaltainen. Tehdasverkossa sijaitsevat eriävät tiedon lähteet yhdessä reunan yhdyskäytävänä toimivan KepserverEX:n kanssa muodostavat toteutuksen reunatason. PowerBI:n verkon ylitse tarjottava SaaS-ympäristö muodostaa järjestelmän alusta- ja yritystasot, painopisteen ollessa alustatason toiminnoissa. Tiedon suunta on toteutuksessa kuitenkin vain tehdasjärjestelmistä kohti pilviympäristöä, reaaliaikanäyttö ei mahdollista esimerkiksi tiedon syöttöä ja välittämistä edelleen alijärjestelmille. Voidaan kuitenkin katsoa, että sovelluksen tuottama tieto tehtaan operaattorille helpottaa kuitenkin päätöksentekoa, ja on siten osaltaan vaikuttamassa takaisinkytkennän omaisesti myös itse tuotantoon, ja siten järjestelmän alustatasolle.



Kuva 21: SaaS-pohjaisen reaaliaikanäytön järjestelmäarkkitehtuuri

Sovelluksen reuna- ja alustatasoja vastaavien tehdasverkon ja PowerBI-pilviympäristön välinen kommunikaatio on toteutettu KepserverEX:n IoT Gateway-liitännäisen ja PowerBI:n tarjoaman REST-rajapinnan avulla. Tiedonsiirtoformaattina on PowerBI:n rajapinnan määrittelemä JSON-muotoinen viesti, ja tiedot lähetetään vain niiden muuttuessa viestiliikenteen määrän minimoimiseksi. Nopeimmat muuttujat päivittyvät tietolähteillä kymmenien sekuntien taajuuksilla, ja hitaimmat kuten aktiivinen tilaustieto tuntien väleillä. Yhteensä visualisoitavia muuttujia on joitakin kymmeniä, joista osa on Python-lisäosan tuottamia koosteita. Luettavia ja prosessoitavia raakadatapisteitä on siten enemmän kuin mitä välitetään alustatasolle.

Tiedonsiirto tehdasverkossa sijaitsevan palvelimen ja PowerBI SaaS-ympäristön välillä ei ole suoraan mahdollista, sillä tehdasverkko on rajattu julkisesta internetistä useilla eri verkkokerroksilla ja palomuureilla. Suora reititys vaatisi siis monimutkaisia asetuksia verkkojen välisiin reitityksiin ja useita palomuuriavauksia. Ongelman ratkaisuna toteutuksessa hyödynnetään asiakasyrityksen konsernitason ESB-väylää (*Enterprise Service Bus*), joka toimii integraatioväylänä yrityksen sisäverkoissa ja niiden ulkopuolella sijaitsevien palveluiden välillä. Arkkitehtuurissa ESB-väylän voidaan katsoa vastaavan IIC-mallin pääsyverkkoa, joka välittää tietoa reuna- ja alustatasojen välillä. Käytännössä PowerBI-ympäristön REST-rajapinnan URL konfiguroidaan ESB-väylälle kohdeosoitteeksi, ja IoT Gatewayn kohdeosoitteeksi asetetaan ESB-väylän tarjoama sisäinen URL. Tietoliikenteen alku- ja päätepisteiden kannalta väylän olemassaolo on siis täysin läpinäkyvää, mutta se yksinkertaistaa viestiliikenteen reititystä huomattavasti, eikä tehdasverkon tietoliikennettä tarvitse paljastaa sen ulkopuolelle. Väylä mahdollistaisi myös tiedostoformaattiin tehtäviä muunnoksia ja muidenkin protokollien välisiä integraatioita.

IIC:n referenssimallin toiminnallisista alueista ratkaisu sisältää kommunikaation, raakadatan ja sovellustason toimintoja. Kommunikaatiotoiminnon käsittämiä toimintoja sisältyy järjestelmän reunatasolle, käytännössä siis luettaessa tietoa paikallisista tietolähteistä. Raakadatalle tehtävää muokkausta edustaa ratkaisussa Python-liitännäinen, jonka avulla tieto muokataan valmiiksi esityskelpoiseen muotoon alusta- ja sovellustasolle. PowerBI-ympäristö toteuttaa sovellusalueen toiminnallisuuksia, eli täyttää liiketoiminnan sille asettamia vaatimuksia ja toimii sen ensisijaisena käyttöliittymänä loppukäyttäjälle.

4.1.2 Tulokset

SaaS-pohjaisen tuotannon visualisointinäytön toteutuksen suurimmiksi eduiksi katsottiin asiakkaan puolesta sen integroituminen pilvitasolla nykyisiin Microsoft-pohjaisiin palveluihin, sekä pilvipohjaisen ympäristön itsessään tuomat edut, tässä tapauksessa erityisesti sen siirrettävyys ja saavutettavuus. Näytön suorituskyky arvioitiin riittävän hyväksi sen käyttötapausta, eli tuotannon seurantaa varten. Viive arvojen päivittymiseen tehtaan OPC-palvelimelta itse näytölle on joidenkin sekuntien, kuitenkin tyypillisesti alle kymmenen sekunnin luokkaa, joka riittää hyvin ympäristössä esiintyvien muuttujien vaihtumistaajuuksien esittämiseen. Hyödynnettäessä valmiita sovellusratkaisuja sekä alusta- että reunatasolla on sovelluksen pystyttäminen nopeaa, eikä se vaadi kokonaista ohjelmistokehitysprojektia eri vaiheineen.

Suurimmaksi haasteeksi projektissa todettiin valitun SaaS-ympäristön rajoitteet tiedon esittämisen ja muokkaamisen suhteen. Rajoitteet eivät järjestelmää määritettäessä olleet selkeästi esillä PowerBI:n dokumentaatiossa, joten ne kohdattiin vasta kun näyttöä rakennettiin jo ympäristöön tuodusta datasta. Näitä puutteita paikattiin erikseen kehitettävällä lisäosalla, joka tarvitsee omat määrittely-, toteutus-, ja testausvaiheensa. Tämä luonnolli-

sesti vähentää valmiin alustan nopeasta käyttöönotosta saavutettuja etuja. Erillinen lisäosa heikentää myös ratkaisun ylläpidettävyyttä ja yleiskäyttöisyyttä. Kaikki alustat eivät myöskään välttämättä tarjoa sopivaa rajapintaa lisäosan kehittämiselle, tai tarjottu rajapinta voi olla vaadittavien ominaisuuksien suhteen rajoittunut.

Voidaankin siis katsoa SaaS-ympäristöön kehitettävän sovelluksen kehitysvaiheen olevan suoraviivainen ja kevyt niin kauan kun ympäristön rajoitteet eivät tule vastaan. SaaS-mallin lähtökohtaisen joustamattomuuden takia sen puutteita joudutaan joko paikkaamaan ympäristön ulkopuolella tai ne voivat estää jonkin vaatimuksen täyttymisen kokonaan. Näin ollen sovelluksen määrittelyvaiheen ja alustojen rajoitteiden syvällisen läpikäynnin merkitys alustavalinnan teossa korostuu, kun teollisen internetin sovellusta suunnitellaan toteutettavaksi valmiiden SaaS-ohjelmistojen varaan. Todennäköisesti myös eri vaatimuksia joudutaan punnitsemaan keskenään: jokin toinen SaaS-visualisointialusta olisi mahdollisesti tarjonnut parempia työkaluja datan käsittelyyn ja muokkaamiseen, mutta olisiko tällöin jouduttu luopumaan esimerkiksi Microsoft AD-tuesta?

4.2 Case 2: PaaS-tasoinen viestipohjainen sovellusarkkitehtuuri

Toisena case-kohteena esitellään Microsoft Azuren PaaS-sovelluskomponenteista koostuvaa sovellusarkkitehtuuria, jonka tiedonvälitys perustuu viestiliikenteeseen reuna- ja alustatasojen välillä. Toteutuksen lähtökohtana on asiakasyrityksen B halu kerätä useiden eri tuotantolaitosten automaatiojärjestelmien tuottamaa prosessidataa pilviympäristöön sinne kehitettävien data-analytiikkasovellusten hyödynnettäviksi. Kutakin sovellusta kehittämään on valittu erilliset yhteistyökumppanit, jotka ovat keskittyneet erityisesti data-analytiikkaan, mutta asiakasyritys tarvitsee tiedon siirtoon tuotantolaitoksilta pilveen teknisen ratkaisun ja toteutuksen. Asiakasyritys B on valinnut pilvialustakseen Microsoft Azuren ja pyrkii keskittämään pilvitoimintonsa siihen. Tiedonsiirtotarpeet vaihtelevat sovelluksittain muutamasta sadasta datapisteestä muutamaan tuhanteen datapisteeseen. Datan tarkkuusvaatimus vaihtelee sovelluksittain sekuntitasolta minuuttitasolle.

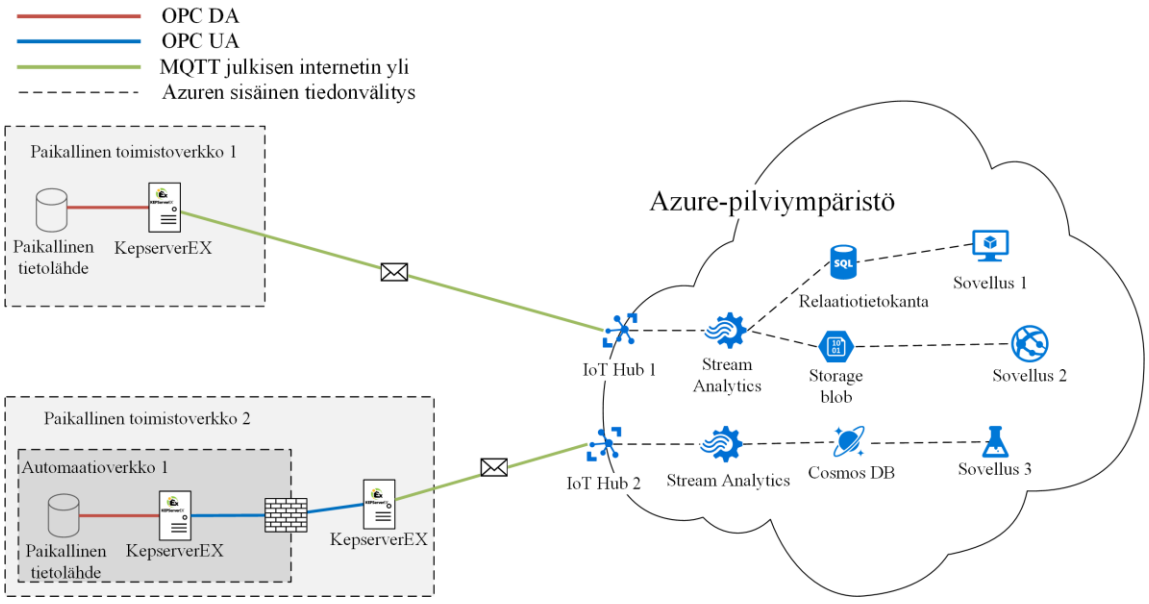
4.2.1 Toteutus

Kullekin sovellukselle on toteutuksessa rakennettu oma tiedonsiirtoväylä, joka kattaa tiedon välityksen tehtaan paikallisesta tietolähteestä pilvessä sijaitsevaan tietovarastoon. Dataa viedään väylän läpi pilveen jatkuvana viestiliikenteenä, jonka lähetystaajuus on määritelty sovellusten tarpeiden mukaisesti vaihdellen välillä 1/s – 1/min.

Lopullisten sovellusten analytiikkaansa tarvitsema data kerätään tehdasverkoissa sijaitsevista paikallisista tietolähteistä, jotka tässä yhteydessä ovat automaatiojärjestelmien omia aikasarjatietokantoja, jotka tarjoavat standardin OPC-rajapinnan. Tätä IIC:n referenssimallin mukaista reunatason kommunikaatiotoimintoa, eli tietojen keruuta paikallisista

tietolähteistä toteuttaa arkkitehtuurissa KepServerEX -ohjelmisto. KepServerEX ei sinänsä ota kantaa mistä dataa luetaan, esimerkiksi tuotannon logiikoiden suora luenta niiden valmistajakohtaisella protokollalla olisi myös toteutettavissa ratkaisuun, mikäli valmista OPC-rajapintaa ei automaatiojärjestelmän puolesta olisikaan tarjolla. Tämä tuo joustavuutta arkkitehtuurin käytännön soveltamisessa myös teknisesti eriäviin ympäristöihin.

Kuvassa 22 esitetään arkkitehtuurin käytännön toteutus kahdella eri kokoonpanolla. Ensimmäisessä dataputkessa KepServerEX ja paikallisen tietolähteen OPC DA-rajapinta sijaitsevat molemmat kevyemmin eristetyssä toimistoverkossa, josta yhteys internetin yli Azureen voidaan muodostaa yksinkertaisella palomuuriavauksella. Koska sekä tietolähde että sitä lukeva KepServerEX sijaitsevat samassa verkossa eikä niiden välissä ole palomuuria, voidaan tietolähdettä lukea suoraan OPC DA-protokollalla. Monimutkaisempi tilanne esitetään kuvan 22 alemmassa dataputkessa, jossa OPC DA-rajapinta tarjotaan palomuurilla eristetyn automaatioverkon sisällä. Tietoturvasyistä suoraa yhteyttä automaatioverkon sisältä julkiseen internetiin ei sallita, eikä OPC DA-protokollan rajoitteitten takia sen tietoliikennettä voida reitittää palomuurin lävitse toimistoverkossa sijaitsevalle KepServerEX-asennukselle. Ratkaisuna käytetään OPC UA-tunnelointia, jolloin automaatioverkon sisään tehty toinen KepServerEX asennus lukee tietolähdettä OPC DA-rajapinnan kautta suoraan, toimien samalla itse OPC UA-palvelimena. Koska OPC UA rakentuu TCP/IP-pinon päälle, voidaan se reitittää kulkemaan palomuurin lävitse ainoastaan yhdellä porttiavauksella. Näin ollen perinteinen OPC DA-tietoliikenne voidaan tunneloida modernin OPC UA-protokollan avulla automaatioverkosta toimistoverkkoon, ja sieltä edelleen välittää julkisen internetin yli Azureen.



Kuva 22: Teollisen tiedon siirto pilveen ja käsittely PaaS-resursseilla.

Kuten edellisen kappaleen SaaS-visualisointisovelluksen toteutuksessa, myös tässä arkkitehtuurissa KepServerEX, tai tarkemmin sen IoT Gateway-komponentti, välittää kerättyä dataa viestipohjaisesti Azuressa sijaitsevalle alustatasolle. Azuren ulkoisena rajapintana toimii IoT Hub, johon viestit välitetään MQTT-protokollalla julkisen internetin yli. IoT hub sisältää myös työkalut siihen yhdistävien laitteiden tai järjestelmien autentikointiin ja hallitsemiseen. IoT Hub siis osaltaan toteuttaa IIC:n mallin resurssienhallintatoimintoja. Reunatason KepserverEX:n IoT Gateway-liitännäisen ja alustatason Azure IoT hubin voidaan yhdessä katsoa toteuttavan arkkitehtuurissa reunan yhdyskäytävän toiminnot siinä mielessä kuin miten IIC:n referenssimalli ne määrittelee.

Alustatason sisällä väylä rakentuu Azuren PaaS-komponenteista, joista osa sisältyy Microsoftin IoT-palvelutarjontaan, ja osa on yleiskäyttöisiä tietovarastoja. IoT hubilta viestiliikennettä välitetään Azuren sisäisesti eteenpäin Stream analytics-komponentin avulla. Stream analytics on Azuren reaaliaikaista viestinvälitystä tarjoava palvelu, joka kuuntelee IoT Hubia ja välittää viestejä toisille palveluille, tässä tapauksessa eri tietovarastoille tallennettaviksi [44]. Stream analytics tarjoaa myös tiedon koostamiseen ja muokkaamiseen liittyviä palveluita, mutta koska kehitettävän ratkaisun vaatimuksena on kerätä nimenomaisesti raakadataa, siirretään viestit tässä toteutuksessa sellaisinaan eteenpäin. Kohdesovellusten tarpeiden mukaisesti käytettyjä Azuren PaaS-tietovarastoja ovat SQL-relaatiotietokannat, NoSQL-kannat kuten CosmosDB sekä Azure Storage Account, jonka jäsentelemättömään Blob Storageen tietoa tallennetaan suoraan tekstimuotoisiksi tiedostoiksi. Stream analytics osaa automaattisesti viedä IoT Hubin tarjoaman JSON-muotoisen sanoman edelleen eri tietovarastoihin vaatien vain pieniä konfiguraatiomuutoksia. Viesti voidaan tulkita esimerkiksi SQL-relaatiotietokannan riveiksi vastaavasti kuin liitteessä A JSON-sanoma muunnetaan CSV-muotoiseksi. Arkkitehtuurin käytännön toteutukseen ei siis juurikaan vaikuta missä muodossa tai millaisen rajapinnan takaa dataa lopulta halutaan käyttää.

4.2.2 Tulokset

PaaS-tasoisessa pilvisovelluksessa käytettyjen alustojen peruskonfiguraatio on pilvitarjoajan vastuulla, joten toteutuksen käytännön rakentaminen on suoraviivaista ja koostuu enemmän eri komponenttien konfiguroinnista kuin varsinaisesta sovelluskehityksestä. Avoimien protokollien ansiosta väylä ei ole myöskään täysin kiinteä, vaan sitä voidaan mukauttaa dataa käyttävien sovellusten tarpeisiin. Tiedonsiirron viiveet koko dataputkelle on arvioitu toteutuksessa melko pieniksi: viive datan luentahetkestä tehtaan aikasarjakannasta sen kirjoittamiseen pilvessä sijaitsevaan tietokantaan on normaalin toiminnan aikana noin kymmenen sekunnin luokkaa. Tässä toteutuksessa esiintyvien sovelluskohteiden kannalta tämän suuruusluokan viiveet ovat merkityksettömiä, sillä data-analytiikka keskittyy tässä sovelluksessa historiadataan ja kokonaisten aikavälien tietoaineiston analysointiin, eivätkä pienet viiveet datan saapumisessa hankaloita sen toimintaa, sillä datan

mukana tehdasjärjestelmiltä kuljetettavien aikaleimojen avulla sovelluksen analytiikka-osuus pystyy päättelemään datan todelliset mittaushetket. Nopeampia toimenpiteitä suorittavaa sovellusta varten myös tiedonsiirron viiveet olisi arvioitava uudestaan ja optimoitava sitä tarpeen mukaisesti. PaaS-komponentit mahdollistavat nopean skaalautumisen pystysuunnassa ja tarjoavat melko kattavat monitorointitoiminnot, joten järjestelmän pullonkaulojen tunnistaminen ja korjaaminen alustatason sisäisesti on suoraviivaista.

Haasteena jatkuvassa reaaliaikaisen datan siirrossa ja sen käytössä rinnakkaisissa sovelluksissa on ratkaisun ylläpidettävyys. Koska dataväylä on rakennettu sovelluskohtaisesti ja se koostuu useista eri PaaS-komponenteista ja sovellustuotteista, joudutaan mahdolliset konfiguraatiomuutokset tekemään tyypillisesti useaan eri paikkaan.

Myös datan mallintamiseen liittyvät ongelmat nousevat esille, kun reunatasolta poistuttaessa siirrytään pois OPC-protokollan itsessään tarjoamasta tietomallista ja siirrytään esimerkiksi pelkkään tekstipohjaiseen datan esitykseen. Kun dataa käsitellään pilvessä vain tekstipohjaisina avain-arvopareina tai tietokannan tietueina, ei ole enää automaattisesti käytettävissä työkaluja datajoukon eheyden tarkkailuun. Esimerkiksi yksittäisen datapisteen lukemisen epäonnistuessa se yksinkertaisesti puuttuu pilveen siirrettävästä viestistä. Vikatilanne pitäisi pystyä tunnistamaan siitä, että mittaus puuttuu datajoukosta, mutta tämä vaatisi tiedonsiirron ja kertyneen datajoukon eheyden jatkuvaa tarkkailua, mikä ei ole itsessään sisällytettynä ratkaisuun. Vastaavassa tilanteessa reunatason OPC-kommunikaatiossa puuttuva datapiste säilyy tietojoukossa, mutta sen OPC laatuparametri muuttuu, joka indikoi ongelmia sen luennassa.

Pilveen kerätyn tietojoukon täydentäminen jälkikäteen ei ole ratkaisussa itsessään mahdollista, vaan puuttuvia tietoja joudutaan täydentämään manuaalisesti esimerkiksi tehtaan tietolähteestä tulostettavalla otteella, joka sisältää puuttuvat tiedot. Ratkaisu tarvitsisi siis kyvyn puskuroida dataa katkoksen ajalta, ja yhteyden palauduttua lähettämään puskurin sisällön pilveen. KepserverEX:n IoT Gateway pitää sisällään lyhyen 10 tuhannen mittauksen puskurin lyhyitä yhteyskatkoja varten, mutta esimerkiksi jo tuhannella datapisteellä tämä puskuri riittää vain n. 10 sekunnin katkoksen ajaksi, jonka jälkeen tietoa menetetään.

Ratkaisussa törmätään helposti myös IoT-terminologian ja teollisen internetin vaatimusten ristiriitaan. Vaikka pilvipohjaiset PaaS-mallin IoT-resurssit tarjoavatkin valtavia laskentaresursseja ja tallennustilaa, ei suoranaisia työkaluja teollisen datan käsittelyyn juurikaan ole tarjolla. Siinä missä IoT-sovellus voi seurata esimerkiksi tuhatta identtistä kuluttajalaitetta, pitäisi teollisen internetin sovelluksen ymmärtää mihin prosessin osaan mikäkin mittaus yhdistetään, ja miten sitä tulisi käsitellä. Esimerkiksi kanoninen tietomalli, johon sisään luettava heterogeeninen data muunnetaan, ja jota eri sovellukset voivat muuntaa tarvitsemaansa muotoon, voisi olla tarpeellinen. Malliin itseensä voisi siten sisällyttää datan käsittelyyn tarvittavan metadatan.

4.3 Case 3: Pilvipohjainen historian – datan keskittäminen ja jakelu sovelluksille

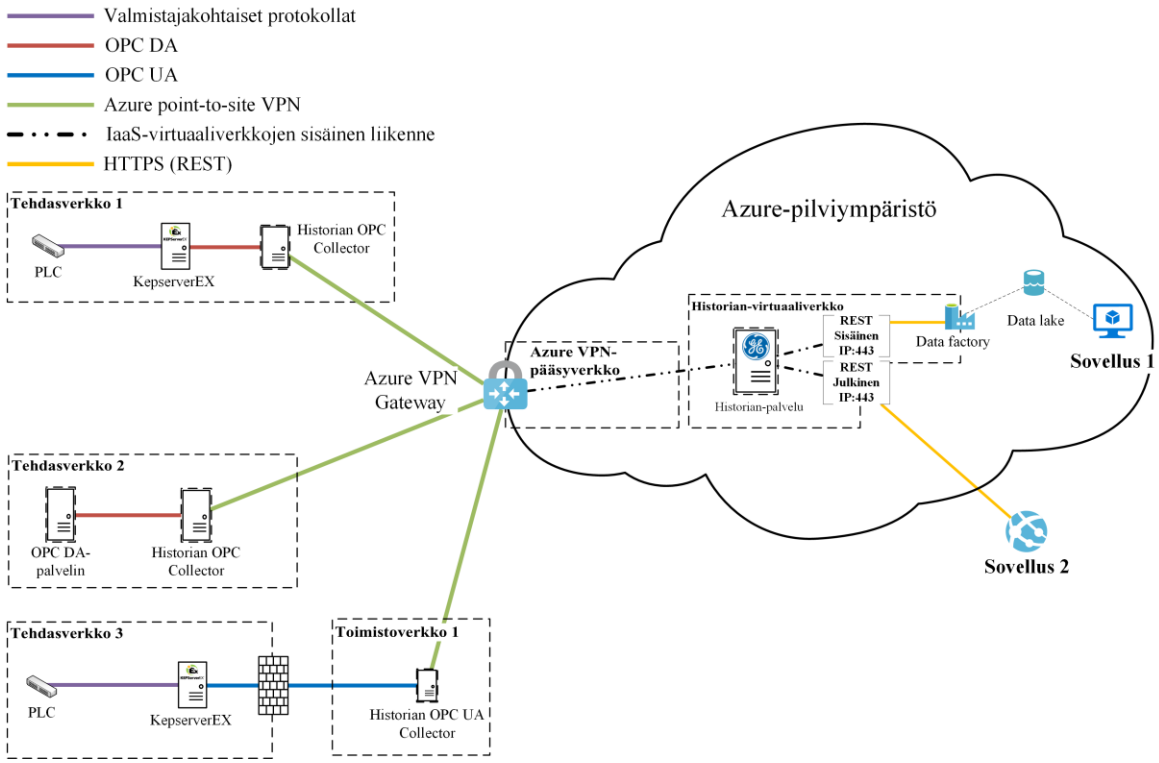
Kolmantena case-kohteena käydään läpi ratkaisu, jossa teolliseen ympäristöön suunniteltua aikasarjatietokantaa hyödynnetään pilviympäristössä sovellusten keskitettynä tietovarastona. Pilvipohjaiseen aikasarjakantaan perustuvan järjestelmän vaatimukset voidaan johtaa pitkälti aiemmin kuvatun PaaS-väylän heikkouksista. Tarpeena ja lähtökohtana sen kehittämiselle on mahdollistaa eri käyttötilanteisiin soveltuva ja vikasietoinen ratkaisu tiedon siirtoon ja varastointiin pilviympäristössä. Vastaavasti datan pitää olla eri tyyppisten sovellusten käytettävissä avoimen rajapinnan kautta, ja sen taajuutta ja formaattia tulee voida muokata sovelluskohtaisesti.

Siinä missä edellinen PaaS-mallin järjestelmistä koostuva arkkitehtuuri nojaa pilvitoimittajan PaaS-alustatarjontaan, siirretään tässä arkkitehtuurissa reunatason teollisten järjestelmien toiminnallisuutta pilviympäristöön. Näin automaatiojärjestelmältä asti periytyvä prosessidatan malli säilyy myös pilvessä, ja myös tietyt valvontamekanismit saadaan käyttöön.

4.3.1 Toteutus

Järjestelmän teknisen toteutuksen keskiössä sijaitsee GE Historian-aikasarjatietokanta, joka on ensisijaisesti prosessidatan varastointiin tarkoitettu tietovarasto. Historian mahdollistaa datan keräämisen useista eri lähteistä, tyypillisesti esimerkiksi automaatiojärjestelmän OPC-palvelimelta. Toteutuksen kannalta olennaisesti Historian voi tarvittaessa toimia täysin hajautetussa verkkoympäristössä, eikä tiedon keruun ja varastoinnin tarvitse tapahtua samassa verkkosijainnissa. Ensisijaisesti aikasarjatiedon arkistointiin kohdennetun sovelluksen etuna on myös kehittynyt tiedon kompressointi ja tästä saatu levytilan säästyminen esimerkiksi puhtaasti tiedostopohjaiseen varastointiin verrattuna. Järjestelmä on myös skaalautuva: käyttö voidaan aloittaa esimerkiksi joillakin sadoilla datapisteillä, ja laajentaa ratkaisua pala palalta käsittämään useita miljoonia datapisteitä suorituskyvyn heikentymättä. Pilviympäristössä skaalaus on erityisen joustavaa, sillä IaaS-virtuaalikoneen resursseja voidaan tyypillisesti kasvattaa ketterästi ylöspäin.

Historian tarjoaa uusimmasta versio 7:stä lähtien avoimen REST-rajapinnan, joka mahdollistaa pääsyn kaikkeen sen sisältämään dataan. REST-rajapinta mahdollistaa tuotteen integroinnin joustavasti osaksi erityyppisiä dataa kuluttavia palveluita, ja on ehdoton edellytys tuotteen soveltamisessa pilviympäristöön. Tuote sisältää itsessään myös REST-rajapintaa hyödyntävän web trend clientin, joka mahdollistaa sen arkistojen selailun ja visualisoinnin webpohjaisesti.



Kuva 23: Pilvipohjainen Historian-tiedonkeruuratkaisu

Kuvassa 23 esitettävässä käytännön toteutuksessa Historian on asennettu Microsoftin Azuressa sijaitsevaan IaaS-virtuaalikoneeseen, jonka käyttöjärjestelmänä on Windows Server 2016 Datacenter. Tiedon keruu tuotantolaitoksilta tapahtuu Historianin hajautetuilla *Collector*-sovelluskomponenteilla, jotka kytkeytyvät paikallisesti tarjolla olevaan tietolähteeseen, kuten OPC-palvelimeen, ja välittävät edelleen niiden tarjoamaa dataa Azuressa sijaitsevaan keskitettyyn Historian-arkistoon. Collectorit voidaan kytkeä lukemaan tietoa myös aiemmissa ratkaisuissa esitellyn KepServerEX:n OPC-rajapinnasta, jolloin dataa voidaan kerätä pilviympäristöön mistä vain järjestelmästä johon KepServerEX tarjoaa laitekohtaisilla ajureillaan pääsyn.

Kommunikaatio collectoreiden ja Historianin välillä tapahtuu Azuren virtuaaliverkon välityksellä siten että tuotantolaitoksella sijaitseva palvelin kytketään Azuren *Point-to-Site* VPN-yhteydellä osaksi Azuressa sijaitsevaa virtuaaliverkkoa, jossa myös Historian-palvelin sijaitsee. Näin liikennöinti on jatkuvasti salattua, eikä Historian-palvelimen internetiin päin julkista rajapintaa tarvitse pitää avoinna. VPN-tunnelin ansiosta verkon konfiguraatio näyttäytyy järjestelmän näkökulmasta yksinkertaisena, sillä palvelimet näkevät toisensa suoraan sisäverkon IP-osoitteilla. Yhteys on myös vikasietoinen, eli tuotantolaitoksen palvelin pyrkii yhdistämään sen uudelleen yhteyden katketessa. Vaihtoehtona julkisen internetin yli reititettävälle VPN-yhteydelle voidaan yhteys pilveen rakentaa myös pilvitarjoajan dedikoitua väylää kuten Azuren Expressroutea pitkin. Näin saavutetaan etuja yhteyden varmuuden ja tietoturvallisuuden suhteen, sillä kaikki liikenne tuotantolaitoksen ja pilven välillä välittyy tällöin täysin julkisesta internetistä eristettyä väylää

pitkin, jonka kapasiteetti on täysin sen hankkineen yrityksen käytettävissä. Vaativaa tuotantokäyttöä ajatellen dedikoidun väylän käyttö lieneekin suositeltavaa, mutta vastaavasti sen käyttökustannukset ovat huomattavasti yksinkertaista VPN-tunnelia korkeammat.

Datan jakelu sen kuluttajasovelluksille tapahtuu toteutuksessa Historianin REST-rajapinnan kautta. Rajapinta tarjoaa useita eri työkaluja datan noutamiseen. Esimerkiksi raakadata-APIn kautta voidaan kysyä kaikki kiinnostavien datapisteiden arvot halutulta aikaväliltä, jolloin rajapinta palauttaa kaikki aikavälin kerätyt datapisteet.

```
https://<historianservername>:8443/historian-rest-api/v1/datapoints/raw/{tag-
Names}/{start}/{end}/{direction}/{count}
```

Ohjelma 2. *Historian raakadata-API*

Raakadata-API saattaa palauttaa kerralla suuren määrän dataa, mikäli pyydettyjä datapisteitä on paljon, tai dataa pyydetään pitkällä aikavälillä. Esimerkiksi pitkän ajan trendinäyttöä piirtävän sovelluksen kannalta voi olla tehokkaampaa käyttää REST-rajapinnan tarjoamaa interpolointi-APIa, jolle voidaan määritellä palautettavien datapisteiden määrä ja intervalli pyydetyn aikavälin sisällä:

```
https://<historianservername>:8443/historian-restapi/v1/datapoints/interpola-
ted/{tagNames}/{start}/{end}/{count}/{intervalMs}
```

Ohjelma 3. *Historian interpolointi-API*

Interpolointi-API siis laskee raakadatasta esityksen, joka vastaa sille annettuja parametrejä.

Reaaliaikaista tietoa tarvitseva datan kuluttajasovellus voi hyödyntää REST-rajapinnan tarjoamaa viimeisin arvo-APIa, joka nimensä mukaisesti palauttaa pyydetyn datapisteen viimeisimmän arvon:

```
https://<historianservername>:8443/historian-rest-api/v1/datapoints/currenva-
lue/{tagNames}
```

Ohjelma 4. *Historian viimeisin arvo-API*

Reaaliaikaisuutta vaativan sovelluksen suunnittelussa on kuitenkin huomioitava Historianin oma datan keruutaajuus: vaikka käytetään viimeisintä Historianin tuntemaa arvoa, se ei välttämättä vastaa viimeisintä muuttujan arvoa itse automaatiojärjestelmällä. Päättelyä datapisteen kelpoisuudesta voidaan tehdä esimerkiksi sen aikaleiman ja nykyhetken välisestä erotuksesta.

Vaihtoehtona datan suoralle käytölle REST-rajapinnan kautta on käyttää Azuren integrointipalvelu Data Factorya. Myös Data Factory kytkeytyy Historianiin käyttäen sen REST-rajapintaa, ja sen avulla voidaan muodostaa säännöstö, joka kopioi dataa johonkin kohdepalveluun. Data Factoryn avulla voidaan esimerkiksi lukea jokaiselle Historianin tarjoamalle muuttujalle edellisen tunnin mittausarvot, muodostaa niistä JSON-sanoma ja

tallentaa se edelleen kerran tunnissa Azure tiedostopohjaiseen Blob Storageen. Näin dataa voidaan tuoda kohdesovellukseen halutun kokoisissa erissä ja kehittäjien kannalta helposti käsiteltävässä tiedostomuodossa.

Sovelluksen tarpeen mukaan Historiania voidaan käyttää tiedon pitkäaikaisen varastoinnin sijaan myös väliaikaisena puskurina määrittelemällä sen arkistoinnille maksimiaika kuten yksi viikko. Näin ratkaisulla saavutetaan edelleen kaikki sille luetellut edut, mutta vältytään tilanteelta, jossa pilveen muodostuu esimerkiksi tehtaan paikallisen aikasarjakannan kanssa identtinen datavarasto. Historian itsessään mahdollistaa myös arkistointiajan asettamisen yksittäisille datapisteille tai -joukoille, jolloin osa datasta voidaan kerätä pitkäaikaisesti ja osa vain puskurinomaisesti.

Vikatilanteiden kannalta Historianiin perustuva arkkitehtuuri tuo joitakin olennaisia etuja verrattuna esimerkiksi datan jatkuvaan virtauttamiseen. Ensinnäkin datan malli on jatkuvasti sovelluksen tiedossa, ja se tarjoaa itsessään työkalut esimerkiksi viimeisimpien mittausarvojen tarkasteluun kullekin datapisteelle. Näin esimerkiksi katkostilanteet datan luennassa voidaan jäljittää tietylle aikavälille. Ohjelmisto tarjoaa myös työkalut collectoreiden etäkonfigurointiin ja niiden tilan seurantaan. Eri vikatilanteista voidaan rakentaa hälytyksiä esimerkiksi organisaation tietohallinnolle.

4.3.2 Tulokset

GE Historianiin ja datan keskittämiseen perustuvan ratkaisun perusta on enemmän teollisuusohjelmistojen kuin puhtaasti pilvipohjaisten ratkaisujen maailmasta. Tämä tuo teollisen datan käsittelyyn pilvessä selkeitä etuja ja käytännön työkaluja, jotka puuttuvat lähtökohtaisesti IT-tekniikoilla rakennetuista pilvimallin sovellusresursseista. Koska Historian on itsessään suunniteltu suurien aikasarja-aineistojen talletukseen, voidaan yhden keskitetyn asennuksen avulla hallinnoida useista eri datalähteistä kerättyä dataa. Ohjelmisto tarjoaa myös työkalut datapisteiden nimeämiseen ja lajitteluun esimerkiksi niiden lähteen perusteella, jolloin dataa kuluttavan sovelluksen on helppo valita tarvitsemansa datat koko aineistosta. Historianin REST-rajapinnan kautta dataa voidaan jakaa kuluttaville osapuolille joustavasti.

Koska ratkaisun järjestelmäarkkitehtuuri nojautuu lähinnä IaaS-tasoisten virtuaalikoneiden ja -verkkojen päälle, ei se sinänsä vaadi pilvi-infrastruktuuria taustalleen. Sumulaskennan periaatteiden mukaisesti vastaavan järjestelmän voisi rakentaa myös yrityksen oman datakeskuksen varaan, mikäli esimerkiksi yrityksen politiikka ei sallisi datan vientiä sen omien palvelimien ulkopuolelle. Dataa kuluttava loppusovellus voisi silti sijaita pilvessä, ja yhteys siihen voitaisiin rakentaa esimerkiksi tunneloimalla http-yhteys yrityksen oman palvelimen ja pilvipohjaisen sovelluksen välillä. Tällöin kuitenkin menetetään mahdollisuus käyttää ohjelmistoa useampien eri verkkosijainneissa sijaitsevien datalähteiden yhteisenä tietovarastona.

Kääntöpuolena ratkaisu on kuitenkin teollisuuden ohjelmistoratkaisuna aloituskustannuksiltaan korkeampi kuin pilvimalliset resurssit, jotka voidaan tyypillisesti hankkia käyttöön pienellä juoksevalla kustannuksella ja skaalata ylöspäin tarpeen vaatiessa. Aivan pienelle yksittäiselle sovellukselle täysimittaisen Historian-järjestelmän hankinta ei välttämättä ole perusteltua, vaan sen edut tulevat paremmin esiin, kun sitä voidaan hyödyntää useiden eri sovellusten ja järjestelmien yhteisenä tietovarastona mahdollisesti koko yrityksen laajuisesti. Mikäli tehdastason paikallisessa tiedonkeruussa dataa kerätään jo aikasarjatietokantaan, voi lopputuloksena olla järjestelmä, jossa identtinen datamassa sijaitsee sekä tehtaan että pilven tietovarastoissa. Tämä ei sinänsä aiheuta järjestelmän kannalta toiminnallisia ongelmia, mutta se voidaan nähdä arkkitehtuuritason periaateongelmana: miksi kerryttää kahta identtistä tietovarastoa? Mikäli paikallinen aikasarjakanta ei kuitenkaan tarjoa pilvessä toimimiseen vaadittuja työkaluja ja tekniikoita, on erillisen pilveen tehtävän tiedonkeruun teko perusteltua.

5. JÄRJESTELMÄARKKITEHTUURI

Tässä kappaleessa koostetaan edellisten kappaleiden tiedoista näkemys siitä, mitä tuotantojärjestelmän ja pilvisovelluksen välisen tiedonsiirron integraation suunnittelussa tulee huomioida. Kappaleessa etsitään näkökulmia, joiden perusteella teollisen internetin järjestelmäarkkitehtuureja voidaan arvioida, ja joiden avulla järjestelmää koskevia haasteita voidaan luokitella. Lisäksi arvioidaan edellisen kappaleen case-arkkitehtuureja muodostettavien näkökulmia perusteella. Lopuksi pyritään muodostamaan kunkin näkökulman haasteisiin vastaava järjestelmäarkkitehtuurihahmotelma, joka olisi sovellettavissa useisiin eri käyttötapauksiin. Tavoite hahmoteltavalle arkkitehtuurimallille on toimi suunnittelun lähtökohtana eri tyyppisille teollisen internetin tiedonsiirron ja -käsittelyn toteutuksille, sekä kohdeyrityksen käytössä, että mahdollisesti yleisemminkin.

5.1 Näkökulmia järjestelmäarkkitehtuurin arviointiin

Integraatiototeutusta voidaan arvioida useista eri näkökulmista, ja onkin lopullisesta sovelluksesta ja sen käyttötarkoituksista kiinni mitkä näkökulmat nousevat olennaisimmiksi kokonaistoteutusta arvioitaessa. Eri näkökulmat voivat myös sisältää ristiriitaisuuksia toisiinsa nähden, esimerkiksi järjestelmään mallikkaasti toteutettu tietoturva koventamisineen voi haitata järjestelmän laajennettavuutta, ja vastaavasti kääntäen järjestelmään tehtävät rajapinnat, joilla mahdollistetaan yleiskäyttöisyyttä ja laajennettavuutta on huomioitava järjestelmän tietoturvan riskiarvioissa.

5.1.1 Vaatimusten täyttäminen

Jotta järjestelmä olisi tarkoituksenmukainen, tulisi sen täyttää sille asetetut odotukset ja tekniset vaatimukset. Vaatimukset voivat olla eksakteja, kuten esimerkiksi case-kohteen 1 vaatimukset koskien tiedon esittämistapaa, tai ne voivat liittyä järjestelmän suorituskykyyn tai kustannuksiin. Rakennettaessa teollisen internetin integraatioita voidaan vaatimuksia täyttää joko kehittämällä itse tarvittavia sovelluskomponentteja, tai käyttämällä valmiita ohjelmistotuotteita., erityisesti kun käytetään IaaS-pohjaisia pilviresursseja.

Järjestelmää suunniteltaessa käytettävissä olevia teknologioita joudutaan karsimaan vaatimusten perusteella, ja ne määrittävät myös kuinka iso osa järjestelmästä voidaan hankkia valmiina, kuten SaaS-ohjelmistoina tai valmiina integraatioratkaisuina. Valmiiden ohjelmistotuotteiden etu on käyttöönoton yksinkertaistuminen ja työmäärän väheneminen, sillä tarvittavat konfiguraatiot voidaan tehdä tuotteen itsensä avulla ilman ohjelmistokehitystyötä. Valmiilla tuotteilla on myös vakauttava vaikutus järjestelmään, sillä järjestelmän voidaan katsoa näiden komponenttien osalta koostuvan huolellisesti testatusta

ja tuotantokäyttöön valmiista ohjelmakoodista. Lisäksi valmiit ohjelmistotuotteet saattavat täyttää useita järjestelmävaatimuksia kerralla, jolloin ne myös keskittävät järjestelmäarkkitehtuuria kokoamalla useita eri toiminnallisuuksia ja integraatioita yhteen.

Valmiit ohjelmistotuotteet ovat kuitenkin niiden tarjoamien ominaisuuksien rajoittamia, ellei niiden toimittaja tarjoa niiden muokkaamiseen avoimia rajapintoja. Mikäli järjestelmän tarpeita täyttäviä ohjelmistotuotteita ei ole tarjolla, tai järjestelmän komponenteiksi valitut tuotteet eivät kykene itsessään täyttämään järjestelmälle asetettuja vaatimuksia, joudutaan puuttuvat tarpeet täyttämään itse kehitettävillä sovellusratkaisuilla. Käytettävissä olevien rajapintojen ja järjestelmien puitteissa itse kehitettävä sovellus tarjoaa laajemman mahdollisuuden järjestelmävaatimusten täyttämiseen kuin valmiit sovellustuotteet, mutta sen myötä järjestelmän toteutus muuttuu enemmän täyden ohjelmistokehitysprojektin suuntaan, minkä myötä esimerkiksi vastuu testaamisesta siirtyy ohjelmistoalustan- tai tuotteen toimittajalta järjestelmän itsensä toteuttajalle. Tämä tarkoittanee myös vaadittavan kehitystyön huomattavaa lisääntymistä järjestelmän kokonaistoimituksessa.

5.1.2 Tietoturva

Tietoturvanäkökulmassa tarkastellaan rakennettavan sovelluksen ympäröiviin järjestelmiin ja sen käsittelemään dataan kohdistamia tietoturvariskejä. Tehdastasolla riskit kohdistuvat sovelluksen datalähteinä käyttämiin tuotantojärjestelmiin, ja pilvitasolla erityisesti sovelluksen käsittelemään dataan. Tietoturvanäkökulma ottaa kantaa myös integraation ja datan käyttöoikeustasoihin ja siihen keillä on pääsy sen komponentteihin.

Tietoturvanäkökulmasta täysin hajautetun järjestelmäarkkitehtuurin voidaan katsoa olevan ongelmallinen tietoliikenteen monitoroinnin ja hallinnan kannalta. Laitteiden viestiessä suoraan julkisen internetin yli pilvisovellukselle on tietoliikenteen oltava lähtökohtaisesti salattua, jotta tiedot eivät ole kolmansien osapuolten luettavissa. Mikäli salaus toteutetaan VPN-tekniikoilla, ei organisaation tietohallinnolla tai muulla verkon ylläpitäjällä ole mahdollisuutta valvoa väylän liikennettä, kuten esimerkiksi sen sisällä käytettyjä protokollia. Verkkoyhteyksien täydellinen hajauttaminen myös pirstaloi verkon rakennetta, esimerkiksi suoralla mobiiliyhteydellä pilveen muodostettavat kommunikaatiolinkit ohittavat täysin laitoksen varsinaisille automaatioverkoille asetetut tietoturvarajoitteet, ja toimivat mahdollisena hyökkäysrajapintana laitoksen sisälle. Tietoturvallisuuden varmistamiseksi lieneekin siis kestävämpi ratkaisu keskittää pilveen rakennettavat tietoliikenneyhteydet käyttämään tunnettuja verkkoja, ja dokumentoimaan niiden rakenne sekä käytetyt protokollat. Näin voidaan varmistua, että luodut dataväylät pysyvät valvonnan alaisina ja ne eivät muodosta selkeästi heikommin suojattua aluetta muuten turvallisessa ympäristössä.

5.1.3 Ylläpidettävyys ja laajennettavuus

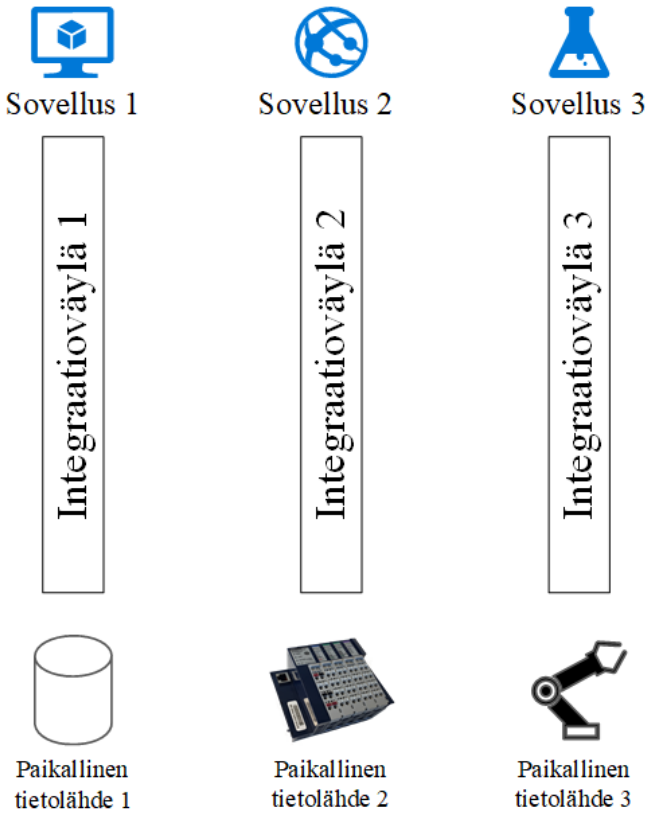
Tiedonsiirtoratkaisun ylläpidettävyys kuvaa miten helposti sen asetuksia tai rakennetta voidaan päivittää esimerkiksi muuttuneiden sovellusvaatimusten vuoksi, ja kuinka moneen eri komponenttiin muutoksia joudutaan tekemään. Laajennettavuus kuvaa mahdollisuuksia lisätä täysin uusia ominaisuuksia järjestelmään, tai skaalata sitä esimerkiksi suhteessa käsitellyn tietosisällön määrään. Jotta järjestelmä olisi kustannustehokas ja siten kannattava investointi myös pitkällä aikavälillä, tulisi sen olla ylläpidettävissä pienellä työmäärällä, ja laajennettavissa palvelemaan myös sen käytön aikana ilmeneviä uusia tarpeita.

Esimerkkinä ylläpidettävyydestä toimii uusien datapisteiden lisääminen automaatiojärjestelmältä pilvisovelluksen saataville: case-arkkitehtuurissa 2 muutokset joudutaan konfiguroimaan erikseen reunatason yhdyskäytävälle, jotta ne haetaan automaatiojärjestelmän rajapinnasta ja sisällytetään pilveen lähetettävään viestiin, ja mahdollisesti Azure-ympäristön stream analytics-komponentille, jotta se osaa tunnistaa lisätyn datapisteen saamastaan viestistä. Mikäli datan tallennuspaikkana käytetään relaatiotietokantaa, joudutaan mahdollisesti myös konfiguroimaan lisätty datapiste uutena tietokantataulun sarakkeena. Case-arkkitehtuurissa 3 vastaava datapisteen lisäys voidaan tehdä suoraan pilviympäristöstä käsin, sillä Historianin konfiguraatiosovellus mahdollistaa siihen kytketyn reunatasolla sijaitsevan collector-sovelluksen avulla paikallisen tietolähteen osoiteavaruuden selaamisen. Näin lisättävä datapiste voidaan hakea suoraan osaksi Historianin tiedonkeruuta suoraan pilviympäristöstä käsin, ja se on välittömästi saatavilla myös REST-rajapinnan kautta sitä tarvitsevalle sovellukselle.

Laajennettavuutta on esimerkiksi uuden paikallisen tietolähteen, tai kokonaisen tuotantolaitoksen lisääminen osaksi järjestelmän reunatasoa. Järjestelmän laajennettavuus voidaan arvioida hyväksi, jos lisäys ei itsessään aiheuta merkittävää lisätyötä ja se ei vaikuta järjestelmän aiemmin konfiguroituihin osioihin. Case-kohteista parhaiten laajennettava on case-kohde 3, sillä siinä uusi tietolähde voidaan käsitellä muista täysin erillisenä, ja siltä kerätty data saadaan arkistoon automaattisesti jo kerättyä dataa vastaavassa muodossa. Case-kohteissa 1 ja 2 joudutaan muutoksia tekemään jossain määrin jo olemassa olevaan toteutukseen pelkkien uusien asennusten lisäksi.

5.1.4 Yleiskäyttöisyys ja siiloutuminen

Arkkitehtuurin yleiskäyttöisyys kuvaa sen soveltuvuutta useiden eri sovellusten yhteiseksi tiedonsiirtoratkaisuksi. Mikäli datalla on pilvitasolla useita eri kuluttajia, on hyödyllistä, jos niiden tarpeet voidaan täyttää yhdellä integraatioratkaisulla useiden rinnakkaisten sijaan. Näin vältetään myös eri sovellusten välinen siiloutuminen, eli rinnakkaisten ja vaakasuunnassa keskenään yhteensopimattomien integraatiototeutusten syntyminen. Voimakas siiloutuminen eri integraatioiden välillä heikentää kokonaisratkaisun ylläpidettävyyttä, ja johtanee pitkällä aikavälillä esimerkiksi kasvaneisiin ylläpitokustannuksiin ja heikkoon laajennettavuuteen. Esimerkkitilanne siiloutumisesta esitetään kuvassa 24. Siiloutuneessa kokonaisarkkitehtuurissa jokaiselta tehtaan paikalliselta tietoläh-



Kuva 24: Integraatioratkaisujen siiloutuminen

teeltä on rakennettu oma integraatioväylänsä pilvisovellukselle, eivätkä väylät ainakaan suoraan tarjoa mahdollisuutta jakaa niiden välittämiä tietoja ristiin dataa tarvitseville sovelluksille. Esimerkiksi sovelluksen 3 tarvitessa tietoa paikalliselta tietolähteeltä 2, joudutaan yhteys integraatioväylään 2 toteuttamaan erillisenä sovelluskehityksenä joko sovellukseen tai integraatioväylään. Vastaavasti jokaiselle väylälle joudutaan rakentamaan

oma tietoliikenneyhteys tehtaan järjestelmiltä pilviympäristöön, jolloin eri integraatioväylien yhteisvaikutus tehtaan tietoverkkoihin ja palomuureihin kasvaa.

5.2 Case-arkkitehtuurien arviointi

Tähän arvioidaan kappaleessa 4 esiteltyjä case-kohteita edellisessä luvussa tunnistettujen kriteerien pohjalta, sekä listataan niiden merkittävimpiä eroja. Arviointi esitetään taulukossa 2.

Taulukko 2: Yhteenveto case-kohteista.

Case-kohde	**1: Tuotannon etenemisen seuranta**	**2: Viestipohjainen sovellusarkkitehtuuri**	**3: Pilvipohjainen Historian**
Pilviresurssien palvelutaso	SaaS	PaaS	IaaS
Reunatason tiedonkeruu	Logiikkaprotokollat, ODBC	OPC Classic, OPC UA, Logiikkaprotokollat	OPC Classic, OPC UA, Logiikkaprotokollat
Reuna- ja alustatasojen kytkentä	HTTP/REST yrityksen ESB-väylän läpi.	MQTT/TLS-liikenne toimistoverkosta internetin yli.	Azure Point-to-Site VPN + Historian Collector.
Toteutuksen vaativuus	Alhainen	Alhainen	Kohtalainen
Yleiskäyttöisyys ja laajennettavuus	Alhainen	Kohtalainen	Korkea
Ylläpidon vaativuus	Alhainen	Kohtalainen	Kohtalainen
Siiloutumisriski	Kohtalainen	Korkea	Alhainen
Aloituskustannus	Alhainen	Alhainen	Korkea
Juokseva kustannus	Kohtalainen	Korkea	Alhainen
Potentiaaliset turvallisuusriskit	Alhainen	Alhainen	Kohtalainen

Selkein ja eniten eri toteutusten järjestelmäarkkitehtuureja eriyttävä tekijä on niissä käytettyjen pilviresurssien palvelutaso. Perusrakenteeltaan kaikkien toteutusten reunatason tiedonkeruu noudatteli samoja suuntaviivoja, eli data kerättiin erillisen yhdyskäytäväohjelmiston avulla tehtaan alijärjestelmiltä ja siirrettiin eteenpäin pilviympäristöön, mutta pilvessä datan käsittely erosi huomattavasti juurikin käytetyn palvelutason vuoksi.

Ainoastaan kohteessa 2 reuna- ja alustatasojen välinen kytkentä kulkee suoraan julkisen internetin yli, tosin sekin TLS-salattuna. Kohteen 1 liikenne viedään ESB-väylää pitkin, ja kohteessa 3 hyödynnetään siirtokerroksen VPN-tunnelointia. Kohteiden 1 ja 2 viestiliikenne koostui JSON-muotoisista viesteistä sisällytettynä joko http/REST:n tai MQTT:n hyötykuormaksi, ja kohde 3 liikennöi käyttäen OPC-protokollaa.

Kohteen 1 käyttämän SaaS-pohjaisen sovellusarkkitehtuurin edut tulevat vastaan matalassa toteutuksen vaativuudessa, sillä SaaS-ohjelmistoa ei juurikaan tarvitse konfiguroida käyttöönottoa varten. SaaS-mallin myötä myös sovelluksen aloituskustannukset ovat matalat kuukausiperusteisen laskutuksen ansiosta, ja tietoturvaa voidaan pitää kohtuullisen hyvänä, sillä sovelluksen toimittaja varmistaa sen. Lisäksi SaaS-pohjainen arkkitehtuuri on vaivaton ylläpidettävä, sillä suuri osa ylläpitotyöstä on sovelluksen toimittajan vastuulla. Arkkitehtuuri ei kuitenkaan ole kovin helposti laajennettavissa tai sovitettavissa eri käyttötapauksiin ilman uutta kehitystyötä. Koska arkkitehtuurille on rakennettu oma väylä yrityksen ESB-väylän läpi, on siiloutumisen mahdollisuus olemassa, ellei samaa väylää kyetä hyödyntämään muillekin vastaaville toteutuksille.

Kohteen 2 PaaS-tasoisen sovellusarkkitehtuurin suurimmat edut ovat matalissa aloituskustannuksissa PaaS-mallisten pilviresurssien käyttöön perustuvan laskutuksen vuoksi. Myös PaaS-malliset resurssit tarjoavat jokseenkin kevyen toteutusvaiheen sovelluksen pystyttämiseen verrattuna uuden ohjelmistoprojektin aloittamiseen, sillä suuri osa toiminnallisuudesta on konfigurointia ja skriptaamista vaille valmista. PaaS-mallisessa toteutuksessa sovelluksen jatkuvan tuotantokäytön juoksevat kustannukset on kuitenkin arvioitava tarkasti etukäteen, sillä esimerkiksi Microsoft Azuren useiden PaaS-mallisten resurssien laskutus perustuu käsiteltyyn viestimäärään. Suurilla viestimäärillä PaaS-tasoisen sovellusarkkitehtuurin kustannukset voivat nousta korkeiksi, vaikka aloituskustannus olisikin mitättömän pieni. Riskinä PaaS-mallisessa dataväyläajattelussa on myös mahdollinen siiloutuminen, sillä mikäli datan eri kuluttajasovelluksille päädytään rakentamaan omia dataväyliä, joudutaan tilanteeseen, jossa ylläpidetään useita rinnakkaisia dataväyliä.

Case-arkkitehtuurissa 3 toteutettiin keskitettyyn historiatietokantaan ja IaaS-mallisiin pilviresursseihin perustuva järjestelmä. Taulukon 2 mukaisesti tämän ratkaisun merkittävimmät edut ovat hyvässä yleiskäyttöisyydessä ja laajennettavuudessa, matalassa siiloutumisriskissä, sekä alhaisissa juoksevissa kuluissa. Kääntöpuolina ratkaisu tarvitsee hieman enemmän panostuksia järjestelmän pystyttämis- ja ylläpitovaiheissa, sillä vastuu myös IaaS-tasoisesta infrastruktuurista on järjestelmän tilaajan vastuulla, ja mikäli ei käytetä sovellusvaatimusten toteuttamiseen valmista ohjelmistoa kuten GE Historiania, joudutaan myös toteuttamaan varsinaisen sovelluslogiikan toteuttaminen omana ohjelmistoprojektinaan IaaS-resurssien päälle, mikä tarkoittaa merkittävää panostusta ennen kuin sovellus on toimintakunnossa. Historian-ohjelmiston itsensä lisensointi aiheuttaa puhtaisiin pilvimallisiin resursseihin verrattuna kuitenkin jokseenkin korkeammat aloituskustannukset, mutta esimerkiksi PaaS-mallin jatkuvaan hinnoitteluun nähden ratkaisun käyttö saattaa silti tulla pitkällä tähtäimellä halvemmaksi, sillä IaaS-resurssien juokseva käyttökustannus on tyypillisesti melko matala.

5.3 Järjestelmäarkkitehtuurimallin hahmottelu

Tässä luvussa summataan tunnistettuja integraatiohaasteita ja hahmotellaan yleispätevä periaatetason integraatioarkkitehtuuri, joka vastaa, tai voidaan toteutusvaiheessa soveltaa vastaamaan useisiin tunnistettuihin haasteisiin, ja joka voi toimia teollisen internetin integraatioiden suunnittelun lähtökohtana. Malli pyritään rakentamaan siten, että se vastaisi kunkin aiemmin esitetyn arviointinäkökulman ongelmakohtiin. Mallissa ei oteta kantaa tarkkoihin teknologioihin tai toteutuksen yksityiskohtiin, jotta sen yleiskäyttöisyys ja sovellettavuus säilyvät.

5.3.1 Pääsy raakadataan ja yhteys pilveen

Teollisen internetin järjestelmä on lähtökohtaisesti riippuvainen siihen kytketyn tuotantojärjestelmän datasta. Ilman mielekästä dataa ei ole myöskään teollisen internetin järjestelmää. Näin ollen ensimmäinen lähtökohta pilvipohjaista järjestelmää suunniteltaessa onkin muodostaa käsitys siitä, onko haluttua dataa jo tarjolla lähdejärjestelmässä tai miten se olisi mahdollista saada tarjolle. Datan saatavuuteen liittyvät haasteet voidaan jakaa karkeasti kahteen osaan: teknisiin ja liiketoiminnallisiin.

Teknisiä haasteita datan saatavuudelle ovat automaatiojärjestelmien käyttämät mahdolliset suljetut protokollat ja rajapinnat, sekä toimiminen eristetyissä ja suojatuissa verkkoympäristöissä.

Vaikka esimerkiksi MQTT-protokollaa tukevia ohjelmoitavia logiikoita ja antureita on jo markkinoilla ja OPC UA mahdollistaa protokollatasolla suoran integraation tehtaan lattiatasolta pilvisovellukselle, eivät nämä modernit tekniikat ole vielä arkipäivää kaikissa teollisuusympäristöissä [32]. Tuotantojärjestelmissä vielä laajasti esiintyvät valmistajakohtaiset protokollat ja rajapinnat hankaloittavat datan keruuta lähdejärjestelmistä, sillä pilviympäristössä toimiva sovellus ei todennäköisesti voi keskustella suoraan teollisuusprotokollia käyttävien järjestelmien kanssa. Tehdastasolla käytettäviä datan lähteitä ja siten myös protokollia voi olla useita eri tyyppisiä, mikä entisestään monimutkaistaa tilannetta. Mikäli pilvisovelluksen kehittäjällä on tarvittava osaaminen ja vaadittavan protokollan määrittely tai jopa valmis ajurikirjasto käytettävissään, lienee mahdollista integroida teollisuusprotokollia osaksi pilvisovellusta, kunhan reuna- ja alustatasoja yhdistävä pääsyverkko sallii tämän. Teollisuusprotokollien toteuttaminen osaksi pilvisovellusta kuitenkin vaatinee käytännössä, että käytetyt protokollat ovat suoraan TCP/IP-pohjaisia, mikä rajaa esimerkiksi useat kenttäväylät ja perinteisen OPC DA:n pois vaihtoehtojen listalta.

Vaadittavan kehitystyön minimoimiseksi ja pilveen tehtävän sovellusmäärittelyn paisumisen välttämiseksi lieneekin hyödyllisintä suorittaa vaadittavat protokollamuunnokset jo tehdastasolla yhdyskäytävää hyödyntäen, jolloin reuna- ja alustatasojen välinen tietoliikenne voidaan toteuttaa täysin TCP/IP-pohjaisilla standardiprotokollilla. Esimerkiksi

kaikkien case-arkkitehtuurien reunatason tiedonkeruu perustuu yhdyskäytäväsovelluksen käytölle. Vaikka yhdyskäytävällä on muitakin rooleja kokonaisarkkitehtuurissa, on yksi sen tärkeä rooli toimia dataan pääsyn mahdollistajana. IIC:n teollisen internetin referenssimallin mukaisesti yhdyskäytävän ei tarvitse olla yksittäinen sovellus tai laite, vaan se voi olla rooli, jonka useammat alijärjestelmät yhdessä toteuttavat. Esimerkiksi kenttäväyliä IP-liikenteeksi muuttava laite voi edelleen välittää tiedot sovellukselle, joka kommunikoi pilvisovelluksen kanssa.

Merkittävä haaste dataan pääsylle ovat myös tietoturvasyistä tiukasti kontrolloidut ja eristetyt tehdasverkkoympäristöt. Pilvisovellukset toimivat määritelmällisesti julkisen internetin välityksellä, joten suljettuja tehdasympäristöjä joudutaan lähtökohtaisesti aina avaamaan, jotta niistä kerättyjä tietoja voidaan integroida pilveen. Tämä voi herättää vastustusta tehdasverkon hallinnossa, sillä tehdasympäristöön avatut tietoliikenneväylät voidaan helposti ja perustellusti tunnistaa uudeksi tunkeutumisväyläksi tehdasverkkoon. Koska avauksia joudutaan välttämättä tekemään haluttujen integraatioiden toteuttamiseksi, tulisi ne suunnitella siten että potentiaalinen hyökkäysrajapinta pysyisi mahdollisimman kapeana. Esimerkiksi case-arkkitehtuureissa 2 ja 3 hyödynnetty OPC UA-tunnelointi on esimerkki tästä: sen sijaan että koko automaatioverkon palomuurein toteutettu eristys jouduttaisiin purkamaan OPC DA-protokollan rajoitteiden vuoksi, voidaan modernimmalla OPC UA-protokollalla liikennettä tunneloida salattua väylää pitkin eristetyn verkkoympäristön ulkopuolelle vain yhden palomuuriin avattavan portin lävitse. Tämä minimoi hyökkäysrajapintaa eristettyyn verkkoon, mutta ei heikennä itse dataintegraation toimivuutta. Eri tyyppisten tunnelointiratkaisujen merkitys lieneekin huomattava kokonaisjärjestelmäarkkitehtuurissa eri osaverkkojen yhdistävänä ja samalla tietoturvallisena ratkaisumallina.

Teknisten haasteiden lisäksi myös ympäröivät organisaatiot ja niiden harjoittama liiketoiminta voivat tulla vastaan teollisen internetin integraatioiden suunnittelussa. Esimerkiksi paikallinen automaatiojärjestelmän toimittaja saattaa nähdä pilviympäristöön kehitettävän sovelluksen ja järjestelmän tuottaman datan siirron pilviympäristöön sen itsensä ulottumattomiin uhkana omalle liiketoiminnalleen. Siinä missä suljettu automaatiojärjestelmä on mahdollisesti sen toimittajan hallussa, ja sen dataa voidaan käyttää vain toimittajan sallimilla tavoilla, pilveen vietyä dataa voidaan tarjota mille tahansa sovellukselle. Tämä luo kilpailua eri sovellusratkaisuiden välille.

Integraatio pilveen herättää siis kysymyksiä siitä kuka omistaa automaatiojärjestelmän tuottaman datan: sen toimittaja vai tuotantolaitoksen omistava yritys? Tuotantolaitoksen omistajan kannalta datan avoimuus ja sen mahdollistamat eri tyyppiset sovellusratkaisut lienevät hyvä asia, sillä niiden myötä kullekin sovellustarpeelle voidaan valita tarpeeseen parhaiten soveltuva ratkaisu. Sovelluksia kehittäville yrityksille teollisen datan avautuminen luo merkittäviä uusia markkinoita. Suljettuja ympäristöjä hallinnoiva, ja niihin omia sovellusratkaisujaan kehittävä toimija voi kuitenkin avautumisen myötä kokea asemansa uhatuksi.

Liiketoiminnallisten syiden lisäksi datan viennissä pilveen voidaan nähdä ongelmana luotavien integraatioiden tietoturvallisuus, mahdollisesti arkaluontoisen tiedon varastointi ulkomaalaisille palvelimille oman IT-hallinnon ulkopuolelle, sekä ylipäätään pilvimallin mielekkyys: saavutetaanko pilveen kehitettävistä sovelluksista tarpeeksi hyötyä, jotta integraatioita kannattaa rakentaa?

5.3.2 Arkkitehtuurimalli

Tässä kappaleessa käsiteltyjä teollisen internetin integraatioita voidaan tarkastella sekä pysty- että vaakasuuntaisina. Pystysuuntainen integraatio kuvaa yksittäisen sovelluksen integraatioväylää, joka sisältää tiedonkeruun tehdastasolla, tiedonsiirron tehdas- ja pilvitasojen välillä, sekä pilvitasolla tapahtuvan tiedon käsittelyn ja sovelluslogiikan. Vaakasuunnassa voidaan tarkastella integraatiokokonaisuutta laajemmin, esimerkiksi yrityksen eri tuotantojärjestelmien ja -laitosten, sekä dataa hyödyntävien sovellusten ja eri pilviympäristöjen tasolla, jolloin saadaan kokonaiskuva rinnakkaisista integraatioväylistä.

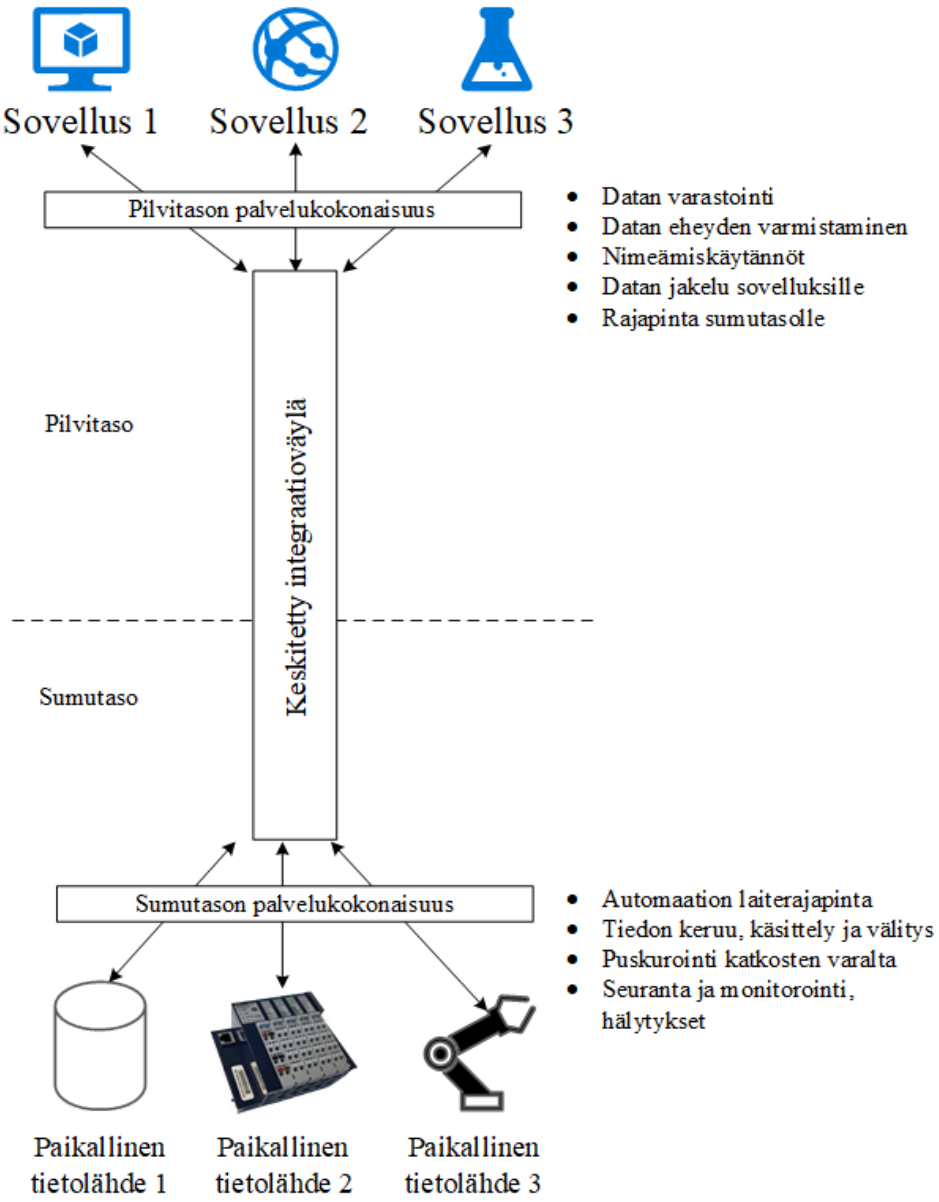
Pystysuuntaisen integraation haasteita ovat pilvisovelluksen vaatimusmäärittelyn ja automaatiojärjestelmän tarjoamien rajapintojen asettamat tekniset haasteet ja rajoitteet. Esimerkiksi edellisessä luvussa kuvatut raakadatan saatavuushaasteet ovat pitkälti pystysuuntaisen integraation ongelmia. Myös pilvisovelluksen vaatima dataformaatti ja taajuus asettavat omat reunaehtonsa integraatiototeutukselle. Pystysuuntainen integraatio asettaa siis reunaehdot yksittäisen datalähteen ja pilvisovelluksen toiminnalle.

Vaakasuunnassa haasteena on siiloutumisen, eri rinnakkaisten lähes samaa tehtävää varten rakennettujen integraatioratkaisuiden syntymisen välttäminen. Siiloutunut järjestelmäarkkitehtuuri aiheuttaa ylimääräistä kuormitusta integraatioväylää ympäröiviin järjestelmiin ja verkkoihin, sillä samoja väylän toiminnan kannalta tarpeellisia toiminnallisuuksia joudutaan toteuttamaan useampia rinnakkaisesti. Siiloutuminen vaikuttaa myös kokonaisarkkitehtuurin kustannuksiin esimerkiksi päällekkäisten ohjelmistolisenssien sekä heikomman ylläpidettävyyden aiheuttamien lisätyötuntien myötä. Siiloutuneessa järjestelmäarkkitehtuurissa myös yksittäisten tietoturvaongelmien havaitseminen on hankalampaa, ja mahdolliset koventamiset joudutaan tekemään useaan rinnakkaiseen kohteeseen. Malliarkkitehtuurin yhdeksi tavoitteeksi otetaan siis siiloutumisen minimointi ja mahdollisesti sen välttäminen kokonaan.

Aiemmissa kappaleissa integraatioita on tarkasteltu myös sen hajautuksen näkökulmasta. IoT lähtee perusperiaatteeltaan liikkeelle laitteiden ja tiedonkeruun hajauttamisesta, esimerkiksi maantieteellisesti tai arkkitehtuurin kannalta. Tehdasjärjestelmä, josta kerättävän tiedon varaan pilvisovellusta ollaan rakentamassa, sijaitsee tyypillisimmin kuitenkin verrattain pienellä maantieteellisellä alueella, ja sen dataa voidaan käsitellä yksittäisestä verkkosijainnista käsin. Koska työn lähtökohtana on selvittää jo olemassa olevien tuotantojärjestelmien integraatioita pilviympäristöön, ei voida tehdä oletusta, että kaikki laitteet

kykenisivät yhdistämään suoraan internetiin, tai edes TCP/IP-pohjaisiin verkkoihin. Kuten edellisessä luvussa todettiin, joko erityisen yhdyskäytäväohjelmiston, tai useiden yhdessä yhdyskäytävän roolin toteuttavien ohjelmistojen avulla saadaan merkittäviä etuja reunatason yhdistettävyyteen ja hallittavuuteen. Näin ollen reunatason yhdyskäytävä nostetaan yhdeksi tässä hahmoteltavan mallin olennaiseksi osaksi.

Periaatteellinen arkkitehtuurimallin hahmotelma esitetään kuvassa 25, joka esittää kolmen eri reunatason tietolähteen ja kolmen eri pilvisovelluksen kytkennän yhteiseen integraatiototeutukseen.



Kuva 25: Arkkitehtuurimalli

Malli on kuitenkin suunniteltu siten, että sekä datalähteitä että dataa kuluttavia sovelluksia voisi järjestelmässä olla huomattavasti enemmänkin. Malli voidaan jakaa karkeasti kolmeen osioon: sumu- ja pilvitasojen palvelukokonaisuuksiin, sekä näitä yhdistävään integraatioväylään.

Järjestelmän reuna-alueella sijaitseva sumutason palvelukokonaisuus voidaan määritellä yhdistelemällä työn aikana ilmenneitä reunatasolle liittyviä haasteita reunatason yhdyskäytävän toimintoihin. Yhdyskäytävä mahdollistaa tarpeenmukaisten protokollamuunnosten suorittamisen, jotta tehdasjärjestelmien dataa ylipäätään voidaan välittää pilviympäristöön, ja siihen voidaan yhdistää myös monipuolisempia raakadatan käsittelytoimintoja, jotka vastaavat tunnistettuihin haasteisiin. Sumutasolle määritellään arkkitehtuurimallissa seuraavat palvelut:

- Liitettävyyden ja tiedonsiirron mahdollistaminen teollisuus- ja IT-protokollien välillä. Case-arkkitehtuurien perusteella tyypillisimmin automaatiojärjestelmän tarjoama rajapinta on joko OPC-pohjainen, tai valmistajakohtainen logiikkaprotokolla kuten Siemens TCP/IP. Näin esimerkiksi kenttäväylät jäävät rajapinnan taakse, ja voidaan hyötyä korkeamman tason protokollan eduista.
- Datan käsittely. Esimerkiksi datan formaattimuunnokset, koosteiden kuten keskiarvojen laskenta, sekä raakadatasta mahdollisesti muodostettavien tunnuslukujen laskenta.
- Tiedonsiirron puskurointia katkeavien verkkoyhteyksien varalle mikä varmistaa, että pilveen kerättävä datamassa voidaan pitää eheänä tehtaan ja pilven välisen verkkoyhteyden palauduttua.
- Sumutason palveluiden tilan seuranta ja monitorointi. Mahdollisuus saada ilmoituksia ja hälytyksiä vikatilanteista.

Sumutason palvelukokonaisuus voi olla yksittäinen sovellus, tai useampi sovellus, jotka yhdessä toteuttavat määritellyt toiminnallisuudet. Sovellusten yhdistävät tekijät voivatkin olla vain niiden kyky kommunikoida keskenään ja niiden sijainti järjestelmän reunatasolla. Huomattava on myös, että kaikkien pilvisovellusten ei tarvitse hyödyntää kaikkia palvelukokonaisuuden tarjoamia palveluita, mutta jotta se todella olisi yleiskäyttöinen ja hyödyllinen, sen tulisi pystyä toteuttamaan listatut toiminnallisuudet. Tietoturvan varmistamiseksi ja hallitsemiseksi pääsy sumutason palveluihin tulisi rajata vain järjestelmän tekniselle ylläpidolle.

Tehtaan ja pilven välinen datalinkki muodostetaan mallissa keskitetysti hyödyntäen joko VPN-tunnelia tai pilvitarjoajan tarjoamaa dedikoitua väylää, sillä edellisessä luvussa kuvatun mukaisesti keskitetyllä ratkaisulla on teollisuuskäytössä etuja etenkin tietoturvan suhteen. Tunneloitu yhteys voidaan määritellä selkeästi vain kahden päätepisteen välille, jolloin sen monitorointi on yksinkertaista. Tunnelointi tarjoaa myös kaikelle liikennöinnille ylimääräisen suojauskerroksen käytetystä protokollasta huolimatta. Liikennöinti tunnelin sisällä voi olla tekstimuotoista viestinvälitystä esimerkiksi MQTT-protokollalla

case-kohteen 2 tapaan, tai se voi hyödyntää OPC tietomallia ja sen UA-protokollaa kuten case-kohteessa 3.

Myös arkkitehtuurin kolmannen osion, eli pilveen rakennettavan toteutuksen lähtökohtana on olla hyödyllinen mahdollisimman monipuolisille sovelluksille. Tämän vuoksi myös arkkitehtuurin pilvitaso kuvataan palvelukokonaisuutena, joka pyrkii vastaamaan useisiin eri datan käsittelyn ja mallintamisen haasteisiin. Todennäköisesti käytännön toteutuksessa myös pilvitason kokonaisuus on useamman eri palvelun ja PaaS-komponentin yhdessä toteuttama. Pilvitason palvelukokonaisuuden tehtäviksi järjestelmäarkkitehtuurissa määritellään:

- Raakadatan varastointi keskitetysti useista eri datalähteistä. Varastoinnin tulisi olla tallennustilan kannalta kustannustehokkaassa formaatissa.
- Datan eheydestä huolehtiminen yhdessä sumutason vastaavan palvelun kanssa.
- Nimeämiskäytäntöjen ylläpito: datapisteiden ja muiden muuttujien nimeäminen siten, että niistä voidaan tunnistaa miltä tehtaalta, linjalta ja koneelta ne ovat peräisin.
- Datan tarjoaminen sovelluksille standardirajapinnoin ja helposti käytettävässä muodossa. Datan mallin ylläpitäminen, ja saapuvan datan muuntaminen tarvittaessa tähän malliin.
- Pääsyn mahdollistaminen sekä raakadataan että käsiteltyyn dataan tarvittavalla tarkkuudella ja avoimella tiedonsiirtoprotokollalla. Mahdollisuus parametrisoida kyselyitä esimerkiksi tietylle aikavälille.
- Mahdollistaa viestintä myös reunatason suuntaan. Tämä vaatii tuen myös reunatason palveluilta, jotta tieto voidaan välittää aina automaatiojärjestelmälle asti.

Pilvitason palvelukokonaisuus ei siis sisällä vielä varsinaisten pilvisovellusten komponentteja, vaan se pyrkii kuvaamaan niiden kehittämistä helpottavan ja niiden toiminnan mahdollistavan palveluinfrastruktuurin. Myöskään pilvitason palvelukokonaisuuden ei tarvitse olla muiden kuin teknisen ylläpidon käytettävissä, mutta sen sisältämälle datalle tulisi voida määritellä eritasoisia pääsyoikeuksia palvelukokonaisuuden rajapinnoissa.

5.3.3 Ratkaisun arviointia

Edellisessä kappaleessa esitetyn arkkitehtuurihahmotelman lähtökohtana oli muodostaa yleiskäyttöinen arkkitehtuurimalli, joka toimisi periaatteellisena viitekehyksenä teollisen internetin tiedonsiirron integraatioiden suunnittelussa. Tässä kappaleessa käydään läpi, miten mallia vastaavan järjestelmäkokonaisuuden hankintaa voidaan perustella, ja mistä tekijöistä sen takaisinmaksun voidaan katsoa muodostuvan. Tämä on tärkeää järjestelmän arvioinnin kannalta, sillä kuten jo työn johdantokappaleessa esitettiin, pilvipohjaisten sovellusten tarjoamien etujen ja mahdollisuuksien on ylitettävä ne haasteet ja koetut riskit

mitä niiden vaatimien integraatioiden toteuttaminen vaatii. Mallilla pyritään myös vastaamaan kappaleessa 1.1 esitettyyn työn päätutkimuskysymykseen, eli teollisten järjestelmien ja pilvipohjaisten sovellusten keskinäisten integraatioiden hyvien suunnitteluperiaatteiden löytämiseen.

Yleiskäyttöisyys otettiin esitetyn arkkitehtuurimallin lähtökohdaksi, sillä sen katsottiin olevan ensisijainen tapa ehkäistä eri järjestelmien keskinäistä siiloutumista. Siiloutumisella taas puolestaan on vaikutuksia kokonaisuuden kustannustehokkuuteen, ylläpidettävyyteen ja tietoturvaan. Näin ollen mallia ei sidota tiettyihin tekniikoihin tai ympäristöihin, vaan sen eri osat kuvataan enemmänkin vastuina ja palveluina kuin kiinteinä sovelluksina.

Ratkaisun myötä samoja datalähteitä käyttävien pilvipohjaisten sovellusten kehitystyö ja integraatioiden rakentaminen suoraviivaistuu, sillä infrastruktuuri datan viemiseksi pilveen ja sen jakeluun eri sovelluksille tarvitsee toteuttaa vain kerran. Yksinkertaisimmillaan uuden sovelluksen liittäminen järjestelmään voi tarkoittaa vain sopivien kyselyiden kirjoittamista integraation pilviympäristössä tarjoamiin rajapintoihin, jotka ovat jo valmiiksi testattuja ja määriteltyjä. Näin ollen myöskään pilviympäristössä toimivien sovelluskehittäjien ei tarvitse tuntea automaatiojärjestelmän tiedonkeruuta tai muuta reunatason toiminnallisuutta voidakseen kehittää kerättyyn dataan perustuvaa sovellusta. Toisaalta ratkaisun myötä myöskään tehdasjärjestelmien ei tarvitse tukea moderneja IT-protokollia tai OPC UA:ta jotta niiden dataa voidaan hyödyntää pilvipohjaisesti, kunhan ne tarjoavat sopivan rajapinnan, johon voidaan kytkeytyä sumutasolta käsin.

Datan varastointi keskitetysti pilviympäristöön standardirajapintojen- ja protokollien taakse mahdollistaa sen jakelun yhdestä sijainnista ja vaadittavassa muodossa. Näin dataa voidaan jakaa eri tyyppisille yrityksille, esimerkiksi analytiikkaa, koneoppimista ja raportointia varten. Ylipäätään ratkaisu mahdollistaa datan tarjoamisen automaatiotoimittajasta riippumatta. Koska ratkaisun datan varastointi ei ota kantaa reunatason tiedonkeruuseen, voidaan useiden eri laitosten ja tehtaiden dataa kerätä yhteen sijaintiin. Tämä mahdollistaa raportoinnin ja yksiköiden vertailun esimerkiksi tehdas-, laitos- tai konetasolla. Raportteja voidaan myös jakaa yrityksen sidosryhmille ja muille toimijoille, sillä ne eivät sijaitse kiinteästi millään laitoksella.

Jotta arkkitehtuurimalli todella olisi yleiskäyttöinen, tulisi sen reuna- ja sumuosuuden olla toteutettavissa eri tyyppisissä tehdasympäristöissä. Lisäksi työn näkökulmana on jo olemassa olevien laitteiden hyödyntäminen teollisen internetin sovelluksissa, joten sumutason toteutuksella tulee olla laaja tuki useille eri automaatiolaitteille ja -protokollille. Mallissa tämä on toteutettu reunan yhdyskäytävän avulla. Tällä mahdollistetaan ratkaisun käyttö eri ympäristöissä, ja minimoidaan tarve tehdä muutoksia tai investointeja olemassa oleviin tuotantojärjestelmiin. Laajalla teknologiatuella vältetään myös toimittajariippuvaisten ratkaisujen syntyminen.

Vastaavasti myöskään järjestelmän pilvitaso ei voi olla riippuvainen yksittäisestä pilvitoimittajasta, sillä nopeasti kehittyvien pilvialustojen välille voi ajan myötä kehittyä merkittäviä eroja esimerkiksi hinnoittelun tai ominaisuuksien suhteen. Yksittäiseen toimittajaan sitoutuminen voi aiheuttaa myös kysymyksiä liittyen sen tarjoamien palveluiden ja niihin talletetun datan säilyvyyteen. Näin ollen toteutuksen pilveen tehtävän toteutuksen tulee olla siirrettävissä eri alustojen välillä. Tämä käytännössä rajaa SaaS-sovelluskomponentit pois toteutuksesta, sillä niiden ominaisuudet vaihtelevat toimittajittain, eivätkä niiden konfiguraatiot ja tietosisällöt ole tyypillisesti siirrettävissä suoraan kilpaileville tuotteille. Mikään ei kuitenkaan estä hyödyntämästä integraatiopalvelun pilveen tuottamaa dataa SaaS-sovelluksessa, mutta integraatioiden osaksi ne eivät sovellu. Eri pilvitarjoajien PaaS-tarjonta vastannee ominaisuuksiltaan toisiaan, mutta vastaavasti toteutuksen sitominen vain yhden toimittajan tarjontaan rajoittaa siirtoa toisille alustoille, sillä myöskään PaaS-komponentit eivät ole identtisiä eri alustojen välillä. Kaikista varminta ratkaisun siirrettävyyden suhteen onkin siis käyttää IaaS-pohjaisia resursseja, jolloin esimerkiksi tietyn käyttöjärjestelmän päällä toimivat integraatiosovellukset voidaan ajaa minkä tahansa pilvitoimittajan tarjoamassa IaaS-virtuaalikoneessa. Myös esimerkiksi ratkaisun mahdollisesti vaatimat tunneloinnit ja muut verkkokonfiguraatiot voidaan rakentaa pilvitarjoajan IaaS-virtuaaliverkkojen varaan. IaaS- ja PaaS-resursseja voidaan tietyssä määrin hyödyntää myös ristiin: esimerkiksi PaaS-tasoinen tietokanta voi toimia IaaS-alustalle rakennettavan sovelluksen tietovarastona, jolloin saadaan siirrettyä sen ylläpitovastuuta pilvitarjoajalle.

Muodostetun arkkitehtuurimallin tiedonsiirto reuna- ja alustatasojen välillä perustuu keskitettyyn tiedonsiirtoväylään, joka voi olla esimerkiksi VPN-tunneli tai pilvitarjoajan dedikoitua reititystä käyttävä väylä. Keskitetty ratkaisu tuo etuja tietoturvamielessä, sillä väylä voidaan alistaa samalle monitoroinnille kuin mitä yritys tekee muullekin sisäiselle verkkoliikenteelleen. Väylä on myös itsessään salattu, ja sen sisällä voidaan lisäksi käyttää pelkästään salattuja protokollia, jolloin ratkaisun tietoturva muodostuu ”sipulinkuorimaisesti” useista kerroksista. Arkkitehtuurimielessä ratkaisun tarjoama siiloutumisen minimointi edistää myös sen tietoturvaa, sillä mitä vähemmän siiloutunut ja pirstaloitunut järjestelmä on, sitä vähemmän se tarjoaa potentiaalista hyökkäysrajapintaa ja tietoturvan kannalta huonosti konfiguroituja komponentteja. Keskitettyyn ratkaisuun on myös helpompi tehdä tietoturvatarkastelua, kuten koventamista, ja sen asetusten monitorointi on helpompaa.

Hahmoteltu arkkitehtuurimalli on luonteeltaan melko kokonaisvaltainen, sillä se pyrkii huomioimaan useita eri haasteita ja mahdollisia toteutusympäristöjä. Tämän vuoksi täysin sitä vastaavan järjestelmän pystyttäminen voi olla yliampuvaa pienen pilvisovelluksen tai pilottihankkeen pohjaksi. Pienemmässä hankkeessa voikin olla kannattavaa karsia sumu- ja pilvitasojen palvelukokonaisuuksia vastaamaan paremmin sovelluksen suoria järjestelmävaatimuksia. Malli täyttänee kuitenkin tavoitteensa, eli sitä voidaan käyttää eri tyyppisten integraatiototeutusten suunnittelun lähtökohtana.

Tulevaisuudessa OPC UA saanee merkittävän roolin etenkin reunatason kommunikaation yleisprotokollana, sekä mahdollisesti myös pilviympäristön sisäisessä teollisen datan käsittelyssä. Esimerkiksi Microsoftin Azureen kehittämä Connected Factory-konsepti [45] mahdollistaa OPC UA:n hyödyntämisen pilven ja reunatason välisessä tiedonsiirrossa, ja pilvisovellusten kytkemisen suoraan palvelun avulla luotavaan OPC UA linkkiin. Microsoft on myös osallistunut OPC UA:n kehitystyöhön ja tuottanut kirjastoja, joiden avulla OPC UA:ta voidaan integroida Azure-sovelluksiin. Moderni OPC UA siis hyvin todennäköisesti tulee syrjäyttämään perinteisen OPC DA:n sekä valmistajakohtaisia protokollia reunatason tietoliikenteessä, ja se voi myös yksinkertaistaa linkin rakentamista tehtaan ja pilven välillä sen tietoliikenneominaisuuksien, kuten tunneloinnin ansiosta. Se ei kuitenkaan pysty itsessään vastaamaan kaikkiin malliarkkitehtuurin sumutasolle määriteltyihin ongelmiin, kuten esimerkiksi tiedon puskurointiin. Näin ollen yhdyskäytävään ja sitä ympäröiviin palveluihin perustuva malli pysynee käyttökelpoisena, vaikka OPC UA ja muut modernit protokollat tulevaisuudessa yksinkertaistavatkin integraatioiden rakentamista tehtaan ja pilven välille.

6. YHTEENVETO

Tässä diplomityössä selvitettiin jo olemassa olevien teollisten järjestelmien tiedonkeruun integrointia osaksi pilvipohjaisia sovelluksia. Työ jakaantui karkeasti kolmeen vaiheeseen, joista ensimmäisessä luotiin yleiskatsaus aiheeseen kirjallisuuslähteiden pohjalta, toisessa aihetta konkretisoitiin case-kohteilla, ja kolmannessa kerätystä tiedosta hahmoteltiin vastauksia tutkimusongelmaan.

Työn ensimmäisessä vaiheessa eli kappaleissa 2 ja 3 taustoitettiin aihetta kirjallisuuslähteiden pohjalta ja saatiin käsitys mistä osista jo-olemassa olevaa automaatiojärjestelmää hyödyntävä teollisen internetin sovellus koostuu. Kappaleessa 2 käytiin läpi automaatiojärjestelmien perinteistä rakennetta sekä teollisen internetin malleja ja sitä, miten ne pyrkivät murtamaan perinteisen automaation rakennetta. Erityisesti IIC:n teollisen internetin referenssimalli toimi viitekehyksenä myös työn myöhemmissä vaiheissa. Kappaleessa 3 tehtiin katsaus teollisen internetin järjestelmän käsittelemän datan luonteeseen ja keskeisimpiin tiedonsiirron integraatiotekniikoihin, kuten järjestelmän eri protokolliin sekä verkkoratkaisuihin. Lisäksi käytiin lyhyesti läpi kaupalliset ohjelmistotuotteet, jotka toimivat seuraavan osion case-kohteiden rakenneosina.

Toisessa vaiheessa eli kappaleessa 4 esiteltiin kolme erityyppistä tapaa toteuttaa integraatio pilvipohjaisen sovelluksen ja sen tuotantolaitoksella sijaitsevan datalähteen välille käyttäen asiakasprojekteina toteutettuja arkkitehtuureja case-kohteina. Järjestelmille yhteinen piirre oli, että asiakasyritysten ajattelumalli on hyvin sovelluslähtöistä ja toteutusten lähtökohtana on tyypillisesti jonkin käyttötapauksen toteuttaminen pilviympäristössä, ei niinkään siirtyminen kertaheitolla koko tuotannon digitalisoivaan IIoT-järjestelmään. Case-kohteista voidaan myös tehdä johtopäätös, ettei tällä hetkellä yksittäisillä toimittajilla ole tarjolla ratkaisuja koko järjestelmäratkaisun toimittamiseen: pilvipalveluiden toimittajilla on tarjota runsaasti laskentatehoa ja talletustilaa sekä eri alustoja ja sovelluksia tiedon käsittelyyn, mutta ei juurikaan täysin valmiita työkaluja tiedon siirtoon tehtaan tietojärjestelmistä niiden omalle alustalle. Vastaavasti teollisen kentän toimittajilla on valmiudet toimia erilaisten teollisten tietojärjestelmien kanssa ja käsitellä niiden tietoa, mutta ei täysin valmiita keinoja datan viemiseksi pilveen ja käsittelyyn pilvialustoilla. Näin ollen avoimien rajapintojen ja eri toimittajien yhteistyön merkitys korostuu pilvipohjaisen järjestelmän kokonaisarkkitehtuurin suunnittelussa.

Kappaleessa 5 eli työn kolmannessa vaiheessa tarkasteltiin integraatioratkaisujen arviointiperusteita ja näkökulmia, ja pyrittiin muodostamaan käsitys hyvistä suunnitteluperiaatteista integroitaessa olemassa olevia teollisuusjärjestelmiä osaksi teollisen internetin sovelluksia. Tuloksena saatiin ensisijaisesti IaaS- ja PaaS-komponentteja hyödyntävä mal-

liarkkitehtuuri, joka tarjoaa viitekehyksen ja infrastruktuurin eri tyyppisten teollisuusjärjestelmien ja pilvipohjaisten sovellusten välisiin tiedonsiirron integraatioihin. Mallissa määriteltiin sumu- ja pilvitasoille palvelukokonaisuudet, joiden täyttämisellä arveltiin parhaiten vastattavan eri tyyppisiin integraatioratkaisuun kohdistuviin haasteisiin. Arkkitehtuurin tavoitteina olivat ensisijaisesti siiloutumisen minimointi ja yleiskäyttöisyyden mahdollistaminen. Näiden näkökulmien katsottiin vaikuttavan myös esimerkiksi toteutuksen tietoturvaan ja laajennettavuuteen. Mallin arvioitiin vastaavaan työlle asetettuun tutkimusongelmaan, eli tarjoavan näkökulmia ja hyviä suunnitteluperiaatteita olemassa olevien automaatiojärjestelmien ja pilvipohjaisten sovellusten välisiin integraatioihin, ja se soveltuu hyvin eri tyyppisten integraatiototeutusten suunnittelun lähtökohdaksi ja ohjenuoraksi.

Työssä käsitelty ja esiin tuotu alan nykytilanne saattaa jäädä teollisen internetin sovellusten aikajanalla eräänlaiseksi välivaiheeksi, jossa tehdastason sisältävät vielä paljon eri legacy-tekniikoita, eikä esimerkiksi OPC UA ole vielä kaikkialla täydellisesti tuettu standardi. Tehtaan lattiatasolta alkavan ja pilveen päättyvän tiedonsiirtopolun sisältämien teknologioiden ja menetelmien yhtenäistäminen yhdeksi väyläksi ei ehkä ole vielä realismia kaikkialla, mutta juurikin OPC UA:n ja pilvialustojen kehittymisen vuoksi se voi tulevaisuudessa olla. Työssä tunnistetut suunnitteluperiaatteet kuten yleiskäyttöisyyteen ja laajennettavuuteen panostaminen integraatioiden toteuttamisessa säilynevät silti relevantteina jatkossakin, vaikka tekniset toteutukset tehtaan ja pilven välillä jossain määrin yksinkertaistuisivatkin ja tarve erillisille yhdyskäytäville protokollamuunnoksineen vähenisikin.

LÄHTEET

[1] Gartner, ”Hype Cycle for Cloud Computing, 2018,” Gartner, 31 Heinäkuu 2018. [Online]. Available: https://www.gartner.com/doc/3884671/hype-cycle-cloud-computing. [Haettu 9 Joulukuu 2018].

[2] P. C. Evans ja M. Annunziata, ”Industrial Internet: Pushing the Boundaries of Minds and Machines,” 2012. [Online]. Available: https://www.ge.com/docs/chapters/Industrial_Internet.pdf. [Haettu 11 Toukokuu 2019].

[3] A. Gilchrist, Industry 4.0 : The Industrial Internet of Things, Apress L. P., 2016.

[4] E. Sisinni, A. Saifullah, S. Han, U. Jennehag ja M. Gidlund, ”Industrial Internet of Things: Challenges, Opportunities, and Directions,” *IEEE Transactions on Industrial Informatics,* nro 14, pp. 4724 - 4734, 2018.

[5] Marketvisio, ”Valtaosa yritysjohtajista viittaa kintaalla miljardibisnekselle,” Marketvisio, 2014. [Online]. Available: https://yle.fi/uutiset/3-7261532. [Haettu 17 4 2018].

[6] W. Steiner ja S. Poledna, ”Fog computing as enabler for the Industrial Internet of Things,” *Elektrotechnik & Informationstechnik,* pp. 310-314, 2016.

[7] Aalto-yliopisto, ”Automaatio 1: Toiminnot osa 1,” 2017. [Online]. Available: https://mycourses.aalto.fi/pluginfile.php/505787/mod_resource/content/3/02_Toiminnot_OSA_1_2017.pdf. [Haettu 12 4 2018].

[8] M. Bajer, ”Dataflow in modern industrial automation systems. Theory and practice.,” ABB Corporate Research, Krakow, 2014.

[9] J. Kletti, Manufacturing Execution System - MES, Berlin: Springer, 2007.

[10] National Institute of Standards and Technology, ”The NIST Definition of Cloud Computing,” 9 2011. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf. [Haettu 28 3 2018].

[11] G. S. Er ja P. Pal, ”Cloud Computing Risks and Benefits,” *International Journal of Advanced Research in Computer Science,* pp. 256-259, 2017.

[12] G. Tucker ja C. Li, ”Cloud Computing Risks,” tekijä: *Proceedings on the International Conference on Internet Computing (ICOMP)*, Athens, 2012.

[13] M. Kavis, Architecting The Cloud, New Jersey: John Wiley & Sons, 2014.

[14] F. v. Bladel, ”IAAS, PAAS & SAAS, hoe zit dat ook alweer?,” ValueBlue, [Online]. Available: https://www.valueblue.nl/cloud-services-iaas-paas-saas-hoe-zit-ook-alweer/. [Haettu 28 3 2018].

[15] F. Bonomi, R. Milito, J. Zhu ja S. Addepalli, ”Fog Computing and Its Role in the Internet of Things,” tekijä: *MCC'12 Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, Helsinki, 2012.

[16] C. Matt, ”Fog Computing - Complementing Cloud Computing to Facilitate Industry 4.0,” *Business & Information Systems Engineering,* pp. 1-5, 2018.

[17] D. Linthicum, ”Edge computing vs. fog computing: Definitions and enterprise uses,” Cisco, [Online]. Available: https://www.cisco.com/c/en/us/solutions/enterprise-networks/edge-computing.html. [Haettu 11 Toukokuu 2019].

[18] Industrial Internet Consortium, ”INDUSTRIAL INTERNET REFERENCE ARCHITECTURE v 1.8,” 2017. [Online]. Available: https://www.iiconsortium.org/IIRA.htm. [Haettu 10 Toukokuu 2018].

[19] Industrial Internet Consortium, ”Introduction to Edge Computing in IIoT,” 2018. [Online]. Available: https://www.iiconsortium.org/pdf/Introduction_to_Edge_Computing_in_IIoT_2018-06-18.pdf. [Haettu 11 Toukokuu 2019].

[20] Q. Hardy, ”Consortium Wants Standards for ‘Internet of Things’,” The New York Times, 27 Maaliskuu 2014. [Online]. Available: https://bits.blogs.nytimes.com/2014/03/27/consortium-wants-standards-for-internet-of-things. [Haettu 9 Toukokuu 2018].

[21] J. Juhanko, M. Jurvansuu, T. Ahlqvist, H. Ailisto, P. Alahuhta, J. M. Collin, T. Heikkilä, H. Kortelainen, M. Mäntylä, T. Seppälä, M. Sallinen, M. Simons ja A. Tuominen, ”Suomalainen teollinen internet – haasteesta mahdollisuudeksi,” Elinkeinoelämän tutkimuslaitos, Helsinki, 2015.

[22] J. Seppälä, ”Automaation elinkaari ja tietoturva,” 16 10 2013. [Online]. Available: https://www.automaatioseura.fi/site/assets/files/1431/seppala_jari_automaation_elinkaari_tty_sas_asaf_16_10_2013.pdf. [Haettu 23 Syyskuu 2018].

[23] Fieldbus inc., ”IEC61158 Technology Comparison - State of the Bus,” 2008. [Online]. Available: http://www.fieldbusinc.com/downloads/fieldbus_comparison.pdf. [Haettu 23 Toukokuu 2018].

[24] Modbus Organization, ”Modbus Application Protocol V1.1b3,” 2012. [Online]. Available: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf. [Haettu 30 Toukokuu 2018].

[25] OPC Foundation, ”What is OPC?,” OPC Foundation, 2018. [Online]. Available: https://opcfoundation.org/about/what-is-opc/. [Haettu 30 Toukokuu 2018].

[26] W. Mahnke, S.-H. Leitner ja M. Damm, OPC Unified Architecture, Berlin: Springer, 2009.

[27] S. M.H. ja B. J., ”A Survey on OPC and OPC-UA - About the standard, developments and investigations,” tekijä: *Information, Communication and Automation Technologies (ICAT)*, Sarajevo, 2013.

[28] P. Hunkar, ”OPC UA vs OPC Classic,” DSInteroperability, Hudson.

[29] Kepware, ”Secure & Reliable OPC Tunneling,” Kepware, [Online]. Available: https://www.kepware.com/en-us/opc-tunnel/. [Haettu 9 Helmikuu 2019].

[30] F. Doglio, Pro REST API Development with Node.js, La Paz, Canelones: Uruguay, 2015.

[31] G. C. Hillar, MQTT Essentials - A Lightweight IoT Protocol, Packt Publishing Ltd., 2017.

[32] B. Adryan ja T. Konigseder, The Technical Foundations of IoT, Artech House, 2017.

[33] OPC Foundation, ”OPC Foundation announces OPC UA PubSub release as important extension of OPC UA communication platform,” 27 03 2018. [Online]. Available: https://opcfoundation.org/news/press-releases/opc-foundation-announces-opc-ua-pubsub-release-important-extension-opc-ua-communication-platform/. [Haettu 12 Toukokuu 2019].

[34] J. Pfrommer, ”Open Source OPC UA PubSub over TSN for Realtime Industrial Communication,” tekijä: *Emerging Technologies in Factory Automation*, Torino, 2018.

[35] A. Butterfield ja G. E. Ngondi, A Dictionary of Computer Science (7 ed.), Oxford: Oxford University Press, 2016.

[36] A. Faul, N. Jazdi ja M. Weyrich, ”Approach to interconnect existing industrial automation systems with the Industrial Internet,” tekijä: *IEEE 21st International Conference on Emerging Technologies and Factory* , Berlin, 2016.

[37] F. Klaffenbach, J.-H. Damaschke ja M. Oliver, Implementing Azure Solutions, Birmingham: Packt Publishing, 2017.

[38] Microsoft, ”Understand Azure global infrastructure,” [Online]. Available: https://azure.microsoft.com/en-us/global-infrastructure/regions/. [Haettu 23 Huhtikuu 2018].

[39] Microsoft, ”Azure products,” [Online]. Available: https://azure.microsoft.com/en-us/services/. [Haettu 4 Huhtikuu 2018].

[40] Microsoft, ”Azure IoT Reference Architecture,” 2016. [Online]. Available: http://download.microsoft.com/download/A/4/D/A4DAD253-BC21-41D3-B9D9-87D2AE6F0719/Microsoft_Azure_IoT_Reference_Architecture.pdf. [Haettu 26 Huhtikuu 2018].

[41] Kepware, ”KEPServerEX - Product Overview,” [Online]. Available: https://www.kepware.com/en-us/products/kepserverex/. [Haettu 15 Toukokuu 2019].

[42] Kepware, ”IoT Gateway - Product Overview,” [Online]. Available: https://www.kepware.com/en-us/products/kepserverex/advanced-plug-ins/iot-gateway/. [Haettu 10 3 2019].

[43] Microsoft, ”Real-time streaming in Power BI,” Microsoft, 02 05 2018. [Online]. Available: https://docs.microsoft.com/en-us/power-bi/service-real-time-streaming. [Haettu 22 Toukouu 2018].

[44] Microsoft, ”What is Azure Stream Analytics?,” 12 07 2018. [Online]. Available: https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction. [Haettu 12 Toukokuu 2019].

[45] Microsoft, ”Try a cloud-based solution to manage my industrial IoT devices,” Microsoft, [Online]. Available: https://docs.microsoft.com/en-us/azure/iot-accelerators/quickstart-connected-factory-deploy. [Haettu 26 Tammikuu 2019].

LIITE A: RAAKADATAVIESTI JSON JA CSV MUOTOISENA

JSON:


```
{
  "timestamp": 1502954301198,
  "values": [
    {
      "id": "Simulation Examples.Functions.Ramp1",
      "v": 43,
      "t": 1502954285762
    },
    {
      "id": "Simulation Examples.Functions.Sine2",
      "v": 4.898e-15,
      "t": 1502954285762
    },
    {
      "id": "Simulation Examples.Functions.Ramp1",
      "v": 47,
      "t": 1502954300760
    },
    {
      "id": "Simulation Examples.Functions.Sine2",
      "v": 39.9994,
      "t": 1502954300760
    },
    {
      "id": "Simulation Examples.Functions.Sine2",
      "v": 39.9994,
      "t": 1502954300760
    }]
}
```


CSV:


```
Servertimestamp,id,v,t
1502954301198,"Simulation Examples.Functions.Ramp1",43,1502954285762
1502954301198,"Simulation Examples.Functions.Sine2",4.898e-15,1502954285762
1502954301198,"Simulation Examples.Functions.Ramp1",47,1502954300760
1502954301198,"Simulation Examples.Functions.Sine2",39.9994,1502954300760
1502954301198,"Simulation Examples.Functions.Sine2"39.9994,1502954300760
```