Sari Järveläinen

Väyläprotokolla ohjelmointi

Metropolia Ammattikorkeakoulu

Insinööri (YAMK)

Älykäs Teollisuus

Insinöörityö

1.6.2021

Tiivistelmä

Tekijä:	Sari Järveläinen
Otsikko:	Väyläprotokolla ohjelmointi
Sivumäärä:	51 sivua + 6 liitettä
Aika:	1.6.2023
Tutkinto:	Insinööri (YAMK)
Tutkinto-ohjelma:	Älykäs Teollisuus
Ohjaajat:	Yliopettaja Jarno Varteva Lehtori Tuomo Heikkinen

Väyläprotokolla ohjelmointi opinnäytetyö sisältää OPC historian alkuvaiheista OPC säätiön perustamiseen. OPC:n vanhempien osien määrittelyistä osat Data Access, Alarm and Event, Historical Data ja XML – DA on kirjoitettu omien otsikoiden alle. Vanhempi OPC versio tunnetaan nykyisin nimellä OPC Classic. OPC johdetaan sanoista Open Platform Communications ja sitä käytetään erityisesti SCADA- ja MES- järjestelmien sovelluksissa suljetuissa paikallisverkoissa rajoittuen Windows-tietokoneisiin. OPC UA käsittää historian lisäksi osia, joita jo OPC Classic sisältää ja ne on sisällytetty OPC UA rakenteeseen. Informaatiomalli kuvaa tiedonsiirron arkkitehtuuria ja sen toimintaa. OPC säätiö kehittää ja hallinnoi OPC UA:ta ja onkin jatkokehittänyt OPC UA 10000 version, johon on sisällytetty aiempia osia, osa on uusia ja osaa on muutettu. Kehitys on ollut varsin voimakasta vuosina 2021-2022 uusien versioiden julkaisussa.

Teollisuusautomaatio ja IoT osuus ovat katsaus molempien historiaan, nykytilanteeseen ja tulevaisuuden kehityskohteisiin. Automaation kehitys teollisen vallankumouksen jälkeen on ollut huimaa ja nyt on menossa teollinen vallankumous neljä, ja puhutaan teollisuus 4.0:stä ja pyritään kyberfyysiseen järjestelmään. Mobiilisti ohjattavat laitteet ja järjestelmät ovat tulleet edullisten perustamiskustannustensa takia vahvaksi vaihtoehdoksi kalliille räätälöidyille teollisuusratkaisuille. Automaation tietoturvan tarve on korostunut viimeisen vuoden aikana huomattavasti kyberhyökkäysuhan kasvaessa Ukrainaan kohdistuvan hyökkäyssodan takia.

Opinnäytetyön ohjelman kirjoituksen aikana käytettiin Sysmac Studiota PLC-yhteydellä, jotta voidaan testata yhteyden toimivuutta ja ohjelman toteutusta. Ohjelmakieleksi valikoitui yrityksen DA-Teamin valinnan perusteella C#, joka on yleisesti käytetty ohjelmointikieli väyläprotokolla ohjelmoinnissa. Ohjelma on kirjoitettu Visual Studio ohjelma-alustaa käyttäen, ja molemmat ovat Microsoftin ohjelmointiin kehittämiä työvälineitä. Ohjelma on opinnäytetyön liitteenä ja sillä voi muodostaa yhteyden PLC kanssa, mutta muuttujan arvoja sillä ei pysty muuttamaan.

Avainsanat: IoT, OPC, OPC UA, Teollisuusautomaatio

Abstract

Author:	Sari Järveläinen
Title:	Bus protocol programming
Number of Pages:	51 pages + 6 appendices
Date:	1 June 2023
Degree:	Bachelor of Engineering
Degree Programme:	Smart Indrusty
Supervisors:	Jarno Varteva, Principal Lecturer Tuomo Heikkinen, Lecturer

---

Bus Protocol programming in this thesis contain old parts begin of the historical times of OPC to establish the OPC Foundation. Oldest parts of the OPC specifications as the Data Access, the Alarm and Event, the Historical Data and XML- DA have included in theis own headers. The Old version of OPC is current called OPC Classic. OPC is coming words Open Platform Communications. OPC UA contains parts that have been already in the OPC Classic and these are now the parts of its structure. Information Model describes structure of the implementation transmission of the information and operations. OPC Foundation improves and manages OPC UA and it has made improvements to the OPC 10000 version, that includes parts of the old versions and some new parts and some of the parts have got improvements. The improvements have been quite strong in years 2021-2022 releases.

Parts of automation of the industry and IoT have look for historical event and current situation to the parts of the future improvements. Progress of the Automation Industry after the First Industrial Revolution has been huge and current The Fourth Industrial Revolution and its called Industry 4.0 and its going on to Cyber Physical Systems. Mobilization has came a potent options for equipments and systems connection and control without expensive industry system just for industry use. Cybercafe security for Automation has came to more preside because of the war of aggression against to the Ukraine.

Writing of the Thesis program the Sysmac Studio was used to connection to the PLC, that we could tested the connection how it has been worked and how program has been made. The Program language C# was a idea from the DA-Team Company, and C# is common used programming language in the Field Protocol programming. Program has been written by using a platform for the Visual Studio, of which both are made by the Microsoft for programming working equipments. Program is in a attachment in the end of the thesis.

Keywords: IoT, OPC, OPC UA, Automation

Sisällys

1 Johdanto....1

2 DA-Team....1

3 OPC....1

3.1 Historia....1
3.2 Määrittely....4
3.2.1 Data Access....5
3.2.2 Alarm and Event....11
3.2.3 Historical Data Access....13
3.2.4 OPC XML- DA....15

4 OPC UA....17

4.1 Historia....17
4.2 OPC UA määrittelyt ja standardit....19
4.2.1 OPC Batch....21
4.2.2 OPC Data eXchange....23
4.2.3 OPC Complex Data....26
4.2.4 OPC Command Execution....26
4.2.5 OPC Security....27
4.2.6 Standardi....28
4.3 Informaatiomalli....29
4.4 Tavoitteet....30

5 OPC Säätiö....30

6 Teollisuusautomaatio....33

6.1 Historia....33
6.2 Nykytilanne....35
6.3 Tulevaisuus....37

6.4 Standardit automaation tietoturvalle ja kyberturvallisuus.....38

7 Teollinen Internet.....41

7.1 Kehitys.....41
7.2 Käyttö teollisuudessa.....42
7.3 Tulevaisuuden käyttökohteet.....44

8 Sysmac Studio.....44

8.1 Virtuaaliympäristö.....44
8.2 PLC-yhteys.....46

9 Väyläprotokolla ohjelmointi.....47

9.1 Yleisesti.....47
9.2 C# (käytetty ohjelmointikieli).....47
9.3 Visual Studio.....48

10 Ohjelmaesittely.....48

10.1 Ohjelman rakennekuvaus.....48

11 Yhteenveto.....50

Lähteet.....5

Liitteet
Liite 1: Ohjelmaesittely

Lyhenteet

ARM: *Automous Mobile Robort.* Itsenäisesti toimiva mobiilirobotti

AU: *Certification Authotity.* Sertificaatin. Allekirjoittaja julkisen avaimen salausjärjestelmässä

CLI: *Common Language Infrastucture.* .NET välikieli

CLR: *Cerification Revocation List.* Sertifikaatin uudelleenlatauslista julkisen avaimen salausjärjestelmässä

CLR: *Common Language Runtime.* Ohjelman kääntämistä varten oleva kieli .NET ympäristössä

CLS: *Common Language Specification.* Kuvaa käyttökelpoisen koodin ja ajonaikaisen ympäristön, mikä sallii korkeampitasoisten kielen käytön eri alustoilla

CNC: *Computer Numerical Control.* Tietokoneella ohjattu työstökone

COM: *Component Object Model.* Microsoftin ohjelmistokomponenttimalli

CTS: *Common Type System.* Asettaa datatyypin ja operaatiot, mitkä jaetaan kaikkien CTS-kääntäjien kesken ohjelmointikielissä

DCOM: *Distributed Component Object Model.* Etäkutsuproseduuri

DDE: *Dynamic Data Exchange.* Tiedonsiirtomekanismi Windows järjestelmässä

HMI: *Human-machine-interface.* Automaatiojärjestelmän käyttöliittymä

IEC: *International Electronical Comission.* Kansainvälinen sähköalan standardijärjestö

IoT: *Internet of Things.* Esineiden Internet

ISO: *International Organization for Standardization.* Kansainvälinen standardointijärjestö

JIT: *Just In Time Complier.* Ajonaikainen kääntäminen

LoRa: *Long-Range Radio Frecuency Network.* Langaton radioverkko, jota käytetään IoT-laitteiden yhteyksillä

M2M: *Machine-to-Machine.* Koneelta-koneelle yhteysjärjestelmä

MES: *Manufacturing Excution System.* Järjestelmä tuotannonseurantaan

MSIL: *Microsoft Intermediate Language.* Kieli kääntämistä varten .NET ympäristössä

NC: *Numerical Control.* Ohjelmoitava työstökone

.NET *.NET Framework* .Net Fremework on Microsoftin kehittämä ohjelmistokomponenttikirjasto

NIST: *National Institute of Standards and Technology.* Yhdysvaltalainen standardijärjestö

OLE: *Object Linking & Embedding.* Microsoftin kehittämä teknologia, mikä salliiupottamisen ja linkittämisen dokumenteille ja muille olioille.

OPC: *Open Platform Communication.* Teollisuuden automaatiosovelluksissa käytettäviä tiedonsiirtostandardeja

OPC A & E: *Open Platform Communicatioa Alarm and Event Specification.* Määrittelee hälytysten ja tapahtumien tyypin vietien tiedonvaihdolle

OPC DA: *Open Platform Communication Data Access Specificatio.* Asiakaspalvelin standardien ryhmä, mikä tarjoaa reaaliaikaiset yhteydet

OPC DX: *Open Platform Communication Data eXchange.* Määrittelee tiedonsiirron yhteydet tiedonvaihdolle

OPC FX: *Open Platform Communication Field eXchange.* Määrittely datan vaihdolle ohjaimelta ohjaimelle ja ohjaimelta laitteelle sekä laitteelta laitteelle

OPC HDA: *Open Platform Communication Historical Data Access Specification.* Määrittelee kyselymetodit ja analytiikan aikaleimatulle historiadatalle

OPC UA: *Open Platform Communication Unified Architecture.* Tiedonsiirtoa varten luotu rajapinta arkkitehtuuri automaation ohjaukselle

PKI: *Public-Key.* Julkisen avaimen salausjärjestelmä

PLC: *Programmable Logic Controlle.* Ohjelmoitava logiikkalaite

PubSub: *Publish-Subcribe.* Menetelmä datan ja tapahtumien ilmoitukselle

PUMA: *Programmble Univer Machine for Assembly.* Ohjelmoitava automaatiolaite asennusta varten

RA: *Registration Authority.* Rekisteröi sertifikaatin julkisen avaimen salausjärjestelmässä

RFID: *Radio Frequency Identificator.* Radiotaajuinen etätunnistus

SCADA: *Supervisory Control And Data Acqustion.* Käytönohjaus- ja valvontajärjestelmä

TCP/IP: *Transmission Control Protocol/Internet Protocol.* Tiedonsiirtoprotokolla

UDP/IP: *User Datagram Protocol/ Internet Protocol.* Yhteydetön tiedonsiirtoprotokolla

UML: *The unified modeling language.* Yleiseen käyttöön kehitetty mallinnuskieli, mikä tarjoaa standardin tavan järjestelmän visualisointiin

UTC: *Universal Time Coordinated.* Koordinoitu yleisaika

VB: *Visual Basic.* Ohjelmointikieli

XML: *Extensible Marup Language.* Merkintäkieli standardi

1 Johdanto

Opinnäytetyön tarkoituksena on tehdä väyläprotokolla ohjelma, jolla saa muodostettua yhteyden ohjelmasta PLC:lle asti ja muuttaa muuttujien arvoja. Teoriaosuudessa tutustutaan OPC Classic- ja OPC UA- rajapintoihin, joilla on mahdollista luoda yhteysrajapinta automaatiolaitteiden ja väylien sekä tuotantoa ohjaavien laitteiden ja järjestelmien välillä. Työn aikana käytössä oli alkuvaiheessa PLC-laite, jonka avulla yhteyttä muodostetaann Sysmac Studion avulla Visual Studion ohjelmistoalustalta tehtävästä ohjelmasta ja sen näytöllä tulevasta lomakkeesta yritetään muuttaa muuttujien arvoja.

Työn tarkoituksena on saada väyläprotokolla ohjelmointia varten versio, jota pystyisi tarjoamaan yrityksille automaation kehitystä varten ja rakentamaan ohjaussysteemin, jolla saadaan yhteys koko yrityksen kattavaksi, niin tuotannon ohjaukseen kuin automaation hallintaan asti, jolloin DA-Team voisi tarjota tätä suunnittelun lisänä automaatioprojekteihinsa, joissa tehdään automaatiosuunnittelu yleensä tuotantolinjakohtaisesti tai jopa koko tehtaan automatisoimiseksi. Väyläprotokolla ohjelma on yksi ohjelmoinnin yritetyimmistä ohjelmista ohjelmoinnin harrastajien ja miksei ammattilaistenkin kommenttien ja kysymysten määrän perusteella Internetin keskustelu- ja ohjelmoinnin kysy- ja vastaa palstoilla.

2 DA-TEAM

Muurlassa DA-Team suunnittelee ja valmistaa automaatiolaitteita, kuten robottisoluja, testausautomaatiota, kuljettimia, manipulaattoreita ja räätälöityjä ratkaisuja. Alkujaan Salolainen yritys, mikä toimii myös palveluliiketoimintamallilla mekaniikka ja automaatiosuunnittelussa, PLC- ohjelmoinnissa Omron-laitekannalle ja simuloinnissa Rockwell Demo3:lla. Da-Team on perustettu vuonna 2009, toimitusjohtajana on Tommi Jokinen. Liikevaihto vuonna 2021 oli 626 tuhatta euroa, liikevoiton ollessa 50 tuhatta euroa. Työntekijöitä oli kaksi. Liikevaihdon nousu vuonna 2021 oli suuri, 611,4%, liikevoittoprosentin ollessa 8,3%. Liikevoitto oli 50 000 euroa. (Taloustiedot: DA-Team)

3 OPC

3.1 Historia

OPC on standardi, jonka Suomen Automaatioseura määrittelee näin: ”OPC:n *idea onkin tarjota standardirajapinta, jonka yli sovellukset voivat tarjota mittaus- ja ohjaustietoa muille sovelluksille valmistajista ja erilaisista teollisuusprotokollista riippumatta*”, OPC säätiö on vastuussa standardin hallinnoinnista ja kehittämisestä. OPC standardi on sarja määrittelyjä, joita on kehittänyt teollisuustoimittajat, loppukäyttäjät ja ohjelmistokehittäjät. Nämä määrittelyt määrittävät rajapinnan asiakkaalle ja palvelimille, kuten myös palvelimelta palvelimelle, mukaan lukien pääsyn reaaliaikaiseen dataan, hälytysten monitorointiin, ja tapahtumiin, historia tietoihin ja muihin sovelluksiin. Standardin ensimmäinen julkaisun tarkoituksena 1996 oli yhdistää PLC:n erityis-protokollat kuten Modbus, Profibus jne. yleiseen standardoituun rajapintaan sallien HMI/SCADA systeemien toiminnan rajapinnassa välittäjänä, mikä voisi kääntää yleisen OPC lue/kirjoita pyynnön laitekohtaiseksi pyynnöksi ja päinvastoin. (OPC Foundation)

Joulukuussa 1995 ensimmäinen versio OPC määrittelyistä, toinen version määrittelyistä seurasi maaliskuussa 1996 ja aloitusseminaari pidettiin Dallasissa, Texasissa huhtikuussa 1996. Lontoossa Englannissa 1996 kesäkuussa ja Japanissa elokuussa 1996 antamaan kiinnostuneille kehittäjille nopean esityksen ehdotetusta standardista. OPC 1.0 julkaistiin elokuussa 29. 1996. Oikea versio 1.0A ”of the OPC Data Acess Specification julkaistiin 1997. Nykyään OPC Classic nimityksellä tunnettava OPC johdetaan sanoista Open Platform Communications ja sitä käytetään erityisesti SCADA- ja MES-järjestelmien sovelluksissa suljetuissa paikallisverkoissa rajoittuen Windows-tietokoneisiin. (OPCConnect.com: History of OPC, Suomen Automaatioseura: OPC-toimikunta)

Alkujaan OPC standardi oli rajattu Windows käyttöjärjestelmille. Windows 3.0 esiteltiin 1990, mikä mahdollisti valtavirtaa edustavan tietojenkäsittelyalustan, joka ajoi useita sovelluksia samanaikaisesti, ja mikä parasta, Windows 3.0 tarjosi standardimekanismin sovelluksille vaihtaa tietoja keskenään. Tämä mekanismi oli Dynamic Data Exchange (DDE), eikä mennyt kauaakaan, kun käyttäjät saivat käyttöönsä yleisiä sovelluksia kuten Microsoft Excelin. Ensimmäisten Human Machine Interface (HMI) ja Supervisory Control and Data Acquisition (SCADA) ohjelmiin perustuvilla tietokone teknologialla vuosina 1989-1991, DDE:tä käytettiin ensimmäisenä rajapintana ohjelma-ajureille prosessien raja-alueelle. Pian tämän jälkeen kehitettiin tuoteriippumaton DDE palvelin.

Tietokoneen liityntäkortit varustettiin DDE ohjelmarajapinnalla ja näin ohjelma pystyi toimimaan DDE asiakasrajapinnan kanssa. (OPCConnect.com: History of OPC, Iwanits & Lange 2006:1)

DDE oli kuitenkin riittämätön, mikä johti siihen, että HMI ohjelmien valmistajat määrittelivät DDE laajennuksia kasvattaakseen suorituskykyä, jolloin tuloksena oli valmistajakohtaisia DDE rajapintoja ja yleistoimivuus rajapinnoissa eri sovelluksille oli kaukana. DDE korvattiin tehokkaammalla Object Linking and Embedding (OLE) teknologialla. OLE for Prosess Control (OPC), nimitystä alettiin käyttämään 1995, Component Object Model (COM) ja Distributed COM (DCOM) ovat Microsoftin teknologiaa, joihin perustuu OPC ja ne ovat Windows-käyttöjärjestelmässä käytettäviä tekniikoita sovelluksille tietojenvaihtoa varten. Automaatiotoimittajat Fisher-Rosemount, Intellution, Opto 22 ja Rockwell Software kuuluivat Task Force ryhmään, joka kehitti OPC standardia. (OPC Foundation: OPC History, Suomen Automaatioseura: OPC-toimikunta)

OLE 2.0 julkaistiin 1992 ja DCOM kehitettiin korvaajaksi OLE teknologialle, sen käytön mahdollisti Windows NT, mikä otettiin nopeasti käyttöön teollisuudessa, mahdollistaen HMI, SCADA ja DCS systeemit. DCOM:ssa asiakasvoi käyttää useita palvelimia samanaikaisesti ja palvelin voi tuottaa toimintoja usealle asiakkaalle samanaikaisesti. DCOM määrittelee standardi mekaniikan palvelimen ja asiakkaan välillä. Koska DCOM ja palomuurien turvallisuus mekanismit sulkivat toisensa pois, niin DCOM:a ei voinut käyttää suoraan yli Internetin ja siksi myös tästä syystä Microsoft kehitti uuden Internet yhteensopivan .NET konseptin (Iwanits & Lange 2006: 2-3), josta ensimmäinen 1.0 versio julkaistiin 2002 (Wikipedia-artikkeli: .NET)

Kaikille ohjelmointikielille tarkoitettu standardoitu ajonaikainen ympäristö Common Language Infrastucture (CLI) on .NET ydin. Tämä määrittely sisältää yleisen standardin oliopohjaisille ohjelmointikielille – Common Language Specification (CLS) ja Common Type System (CTS). Periaate on sama kuin Javassa: CLI ei tuota prosessorille määriteltyä koodia, vaan jonkinasteista välittäjäkoodia, Microsoft Intermediate Language (MSIL), joka ajetaan ajonaikaisessa ympäristössä. Common Language Runtime (CLR) ja siirretään kääntäjälle, Just In Time Compiler (JIT), joka luo prosessorikohtaisesti määritellyn koodin ajon aikana. Ennen prosessorikohtaisen koodin luontia MSIL koodin tarkistaa CLR, ja luokkien ajon aikana koodi hyväksytään. CLR myös kontrolloi pääsyä ulkopuolisiin resursseihin kuten tiedostoihin ja verkkolinkkeihin ja vapauttaa resursseja,

jotka eivät ole enää käytössä. Konsepti .NET sallii yhteistoiminnan välittämättä ohjelmointikielestä, esimerkiksi jos luokat on määritelty C#-kielellä ne voi johtaa VB:llä ja .NET tarjoaa laajan toimivan luokka kirjaston ja vahvasti tuetun ohjelmisto kehityksen (Iwanits & Lange 2006: 3).

OPC Classicin etuja laitevalmistajille on tuotteen yhteentoimivuus kaikissa OPC-järjestelmissä, eikä tarvitse jokaiselle laitteelle tehdä erikseen ajuria. Lisäksi ei tarvitse tuntea jokaisen järjestelmän erityisvaatimuksia ja uuden laitesukupolven tulessa markkinoille tarvitsee päivittää vain OPC palvelin, eli tarvitaan vain yhden tuotteen tuki. Ohjelmistotoimittajien ei tarvitse käyttää aikaa eri ajureiden ohjelmointiin ja tuotteita voi käyttää kaikkien laitteiden kanssa. Standardirajapinta mahdollistaa laajan tuotevalikoiman järjestelmäsuunnittelijoille, mitä laitteita ja ohjelmistoja valittavissa projekteihin käyttöönottoajan pysyessä lyhyenä. Loppukäyttäjälle OPC tarjoaa lisää joustavuutta, valinnanvapautta erilaisten tuotteiden saatavuuden kasvaessa, mikä pienentää myös kustannuksia kasvattamalla samalla laatua ja käyttömukavuutta. Lisäksi hyväksytty standardi lisää investoinnin suojaa ja mahdollistaa eri laitevalmistajien tuotteiden yhdistämisen. (Iwanits & Lange 2006: 9)

OPC Classicin nopea kasvu vuodesta 1997 muutamasta OPC tuotteesta useisiin tuhansiin vuoteen 2005 mennessä, osoittaa tämän teknologia hyvää vastaanottoa ja valtavaa menestystä (Iwanits & Lange 2006: 5) ja se on laajasti käytössä erityisesti SCADA- ja MES- järjestelmissä Windows-tietokoneisiin ja suljettuihin paikallisverkkoihin rajoittuen. OPC Classicin jälkeen OPC säätiö on kehittänyt OPC UA eli OPC Unified Architecture – teknologiaa korvaamaan OPC Classicin vuodesta 2005 lähtien ja sen toivotaan tuovan 2020-luvulla standardointiin koneiden ja laitteiden toiminnallisuuksia, jolloin asenna ja liitä mahdollistaisi luotettavan ja nopean käyttöönoton. Tätä Plug & Produce toiminnan kehitystä tukee monet järjestöt, kuten NAMUR, OPAF (Open Process Automation Forum), Platform Industrie 4.0, PLCopen, umati (Universal Machine Tool Interface) (Suomen Automaatioseura: OPC-toimikunta).

3.2 Määrittely

Suomen Automaatioseuran mukaan: ”*Tunnetuimmat OPC Classic- määrittelyt ovat DA (Data Access), A&E (Alarms and Events) ja HDA (Historical Data Access). Näistä DA on tarkoitettu tosiaikaisen prosessidatan siirtoon prosessilaitteista ja ohjausjärjestelmistä. A&E on tarkoitettu hälytys- ja tapahtumatietojen välittämiseen ja HDA puoles-*

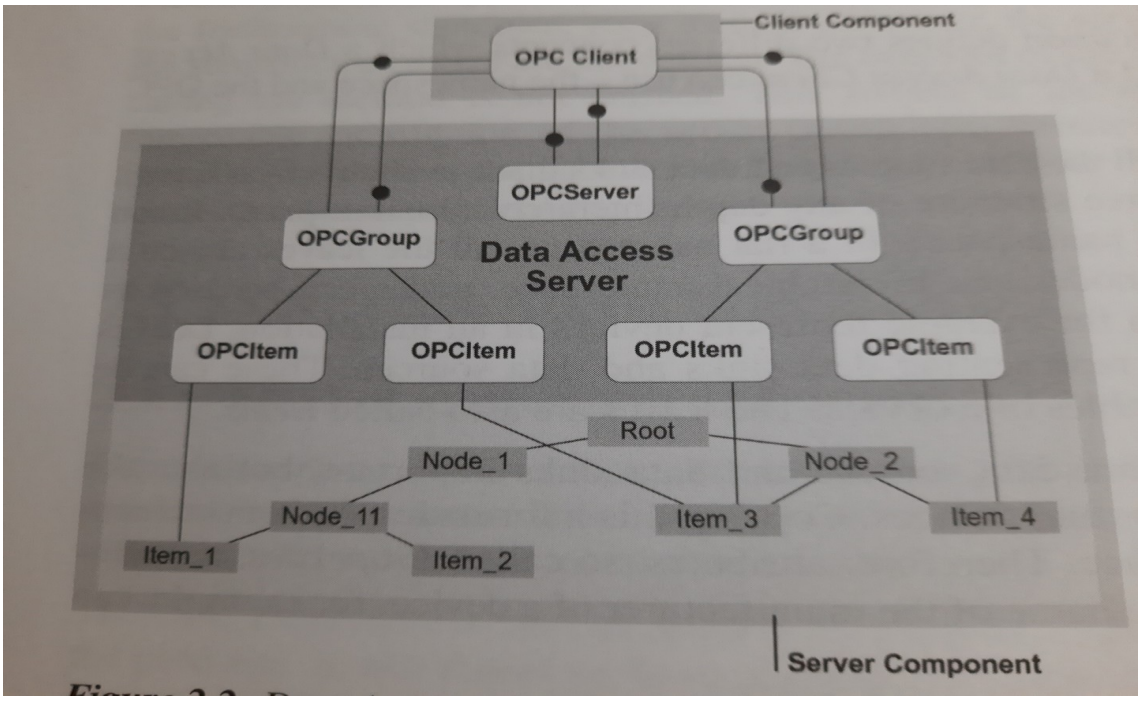
taan on tarkoitettu historiatietojen siirtoon."(Suomen Automaatioseura: OPC-toimikunta). DA määrittelyt luotiin versiossa 2.0 lokakuussa 1998. Tammikuussa 1999 A&E version 1.01 määritteli kuinka ilmoitetaan A&E palvelimelle yllättävistä tapahtumista prosessissa. Historiatietojen siirtoa, HDA, määrittely aloitettiin jo vuonna 1997 ja se valmistui syyskuussa 2000. Määrittelyt ja toteutukset tietoturvalle OPC komponenteille on ollut käytössä syyskuusta 2000 (Iwanits & Lange 2006: 5).

3.2.1 Data Access

DA Specification on vanhin OPC määrittelyistä. Alkujaan se määritteli rajapinnan asiakkaan ja palvelimen välillä mahdollistaen pääsyn prosessidataan. Datalähteet voi paikantaa I/O – korteille ja DA palvelin tarjoaa yhden tai useamman yhteyden. Tietolähteenä voi olla esimerkiksi lämpötila- antureita, jotka on kytketty suoraan tietokoneeseen ja ne voi olla myös sijoitettu ohjaimiin. DA asiakas voi olla kytkettynä useampaan DA palvelimeen kenttäväyliin sarjaliittimellä samanaikaisesti. DA asiakas voi olla yksinkertaisimmillaan Excel -taulukko tai monimutkainen VB kuvaus (komponenteista/laitteista) isommasta ohjelmasta HMI - tai SCADA -järjestelmästä (Iwanits & Lange 2006: 25).

DA palvelin on yksinkertaisimmillaan ohjelma, joka esimerkiksi tarjoaa pääsyn PLC:n rekistereihin sarjaportin kautta tai monen muuttujan monimutkainen ohjelma eri laitteiden laajoilla tiedonvälitystavoilla tai PC perusteinen ohjaus, joka voi olla osa isompaa ohjelmistoa, missä se voi saada dataa näiltä ohjelmilta. Näihin toteutusvaihtoehtoihin ei ole asetettu rajoja standardissa. OPC:ssa määritellään kaksi eri käsitettä DA palvelimille työkaluiksi, joita DA voi käyttää eli nimiavaruuden ja OPC olio hierarkian. Nimiavaruus sisältää kaikki datalähteet ja varastot, jotka ovat palvelimelle saatavilla. Nimiavaruus voi olla puumalli, millä tahansa syvyydellä eli hierarkkinen nimiavaruus, mutta se voi olla myös litteä, jolloin puumallin lehdet ovat yhdessä tasossa. Puurakenteessa ei ole solmuja, vaan solmuina voi olla laitteet rakennelmassa. Lehdet ovat saatavilla solmuille, edustaen datavarastoja ja – lähteitä, mitkä olla asetuspisteitä ja laitteiden mitattuja muuttujia. Lehtiä on myös kutsuttu Item – sanalla eli osioiksi. Sovelluksissa, eivät ainoastaan datalähteet ja datavarastot ole tärkeitä, vaan myös näiden laite- ja muuttujatiedot. Lehtien/osioiden tietojen kopiointi aiheettomasti laajentaa nimiavaruutta siksi attribuutit eli ominaisuudet kohdistetaan lehtiin ja solmuihin ja esimerkiksi laitevalmistajan nimi voidaan kuvata ominaisuudeksi (Iwanits & Lange 2006: 25).

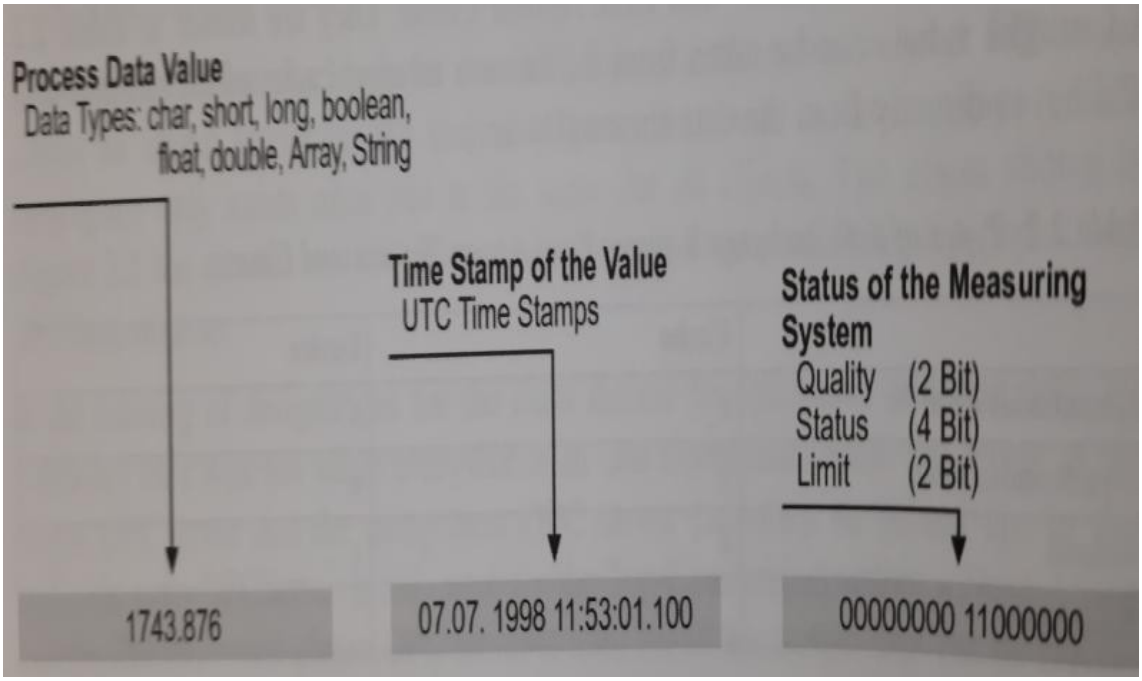
DA asiakas voi luoda useampia OPC olioita DA palvelimelle määrittelemään kuvaa prosessista. OPC palvelin -olio on ylemmän tason olio OPC olio hierarkiassa. OPC-GroupObject, myöhemmin mainittu ryhmäoliona, on seuraavan tason muoto ja viimeisellä tasolla on OPCItemObject, myöhemmin mainittu osio-oliona, mikä kuvaa lehtiä tai ominaisuuksia nimiavaruudessa, arvoja ja parametreja, joista DA asiakas on kiinnostunut. Ryhmäoliota käytetään järjestelemään osio-olioita, mikä voidaan tehdä loogisesti laittamalla kaikki muuttujat yhteen prosessilohkoon tai dynaamisesti samasta tai samanaikaisesta aikaikkunasta käytetyt prosessimuuttujat samaan lohkoon käyttäjän valinnan mukaan. Kaikkia ryhmäolioita hallinnoi OPCServer eli palvelinolio. Oliohierarkioita voi olla saatavilla OPC palvelimen laitteella, jos useammalla DA asiakkaalla on pääsy näihin laitteisiin. Jokaisella asiakkaalla on olemassa tietty hierarkia, mikä tarkasti vastaa sen vaatimuksia. (Iwanits & Lange 2006: 25-26)



Kuva 1: DA palvelin - nimiavaruus ja oliohierarkia (Iwanitz & Lange 2006: 26)

Kuvassa 1 oliohierarkia ja nimiavaruus DA palvelimella, jossa vain kaksi oliota hierarkiassa ovat todellisia DCOM olioita-palvelinoloilla ja ryhmäoliolla, ja vain näille kahdelle on määritelty rajapinta, metodi ja parametrit. Todellisissa sovelluksissa useita arvoja voidaan lukea ja kirjoittaa samanaikaisesti, ja on tehokkaampaa olla pääsy useampaan osio- olioon yhdellä yhteysyrityksellä, mikä on tehty metodilla ryhmäolion rajapinnassa.

Tästä syystä osio-oliolla ei ole yhtään asiakasrajapintaa ja niiden toteutus voidaan pitää yksinkertaisena. Tämä ei poissulje luku ja kirjoitus pääsyä yksittäiseltä osio-olioilta. Automaation rajapinta on määritelty kaikille kolmelle olioryhmälle. Data voidaan kolmella tapaa siirtää DA palvelimelle eli arvo on saatavilla tietokoneella sovelluksessa tai arvo on kutsuttu ulkoiselta laitteelta esim. kenttälaitteelta tai arvo on lähetetty ulkoiselta laitteelta kutsumatta. Arvo voidaan ottaa palvelimen sisäisestä välimuistista tai suoraan laitteelta. Käytettäessä synkronista tai asynkronista kutsua asiakas voi mukautua optimaalisesti tilanteeseen. Synkronoidussa kutsussa asiakas kutsuu metodia ja odottaa paluuarvoa, kun arvot saatavilla tietokoneella. (Iwanitz & Lange 2006: 26)



Kuva 2: Datamalli DA määrittelyssä (Iwanitz & Lange 2006: 28)

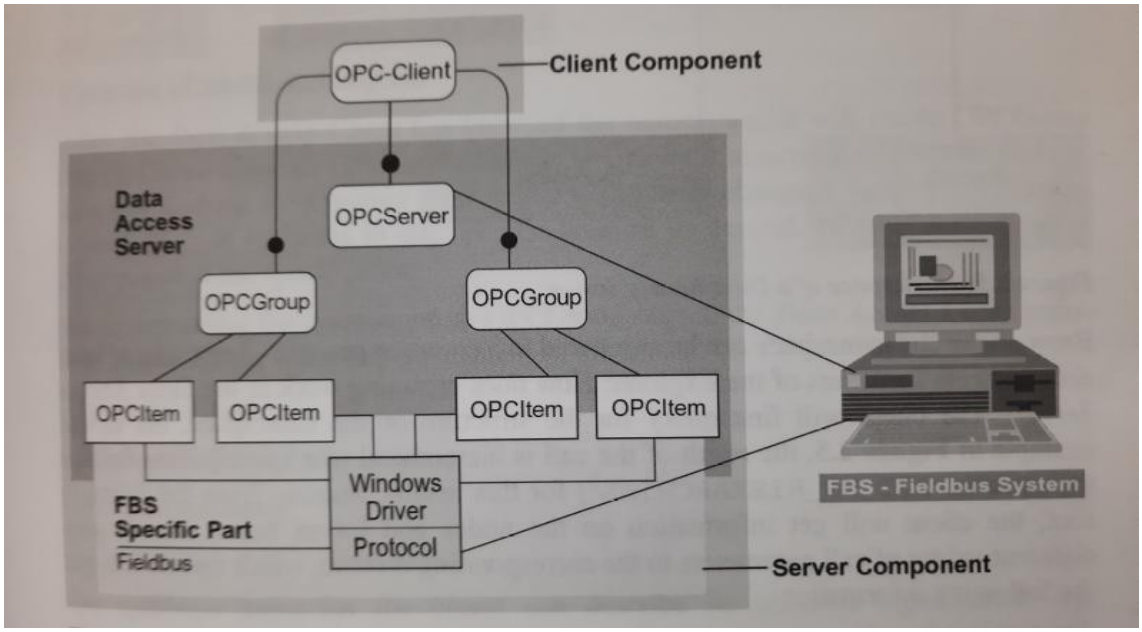
Palvelin tarjoaa avoimen pääsyn dataan eri sovellusalueilla ja asiakas mahdollistaa sen käytön muille sovelluksille. OPC käyttää DCOM VARIANT datatyyppiä, jossa informaatio on DCOM VARIANT:ssa. Palvelin ja asiakas ovat vastuussa DCOM datatyypin vaihtamisesta muuksi datatyypiksi ja päinvastoin. Aikaleima, Time Stamp, on käytössä kun arvolle annetaan lukuaika, joka on 8 bittiä pitkä ja ilmaisee aikaa 01.01.1601 ja aikaleima tulee 100ns välein. Aikaleiman esitystapana on Universal Time Coordinated (UTC) Windowsille. Prosessista saatava arvo, jolle aikaleima annetaan voi olla tyyppiä char,

short, long, boolean, float, double, Array, String ja tila-arvo muodostuu 8 bitistä. Tila-arvosta 2 bittiä on laadulle, 4 bittiä kuvaa tilaa ja 2 bittiä raja-arvoja. Tila bitit ovat hyvä (Good), huono (Bad) ja epävarma (Uncertain) eli yhteys on, mutta mitattu arvo ei ole määriteltyjen raja-arvojen sisällä, lisämerkintänä voi olla viimeisin käypä arvo (Last usable Value). Kaksi viimeistä bittiä voi ilmoittaa anturiviasta ja arvo voi olla merkitty mahdottomaksi. (Iwanits & Lange 2006: 28)

Palvelimen käynnistyksen yhteydessä DA asiakas pääsee OPC palvelin olioon ja asiakas käy läpi nimiavaruutta ja saa tietoja sen rakenteesta. Pääavaimen avulla luodaan osio-olioita. OP_NS_HIERARCHIAL kysely aloittaa virtuaalijuuresta root ja asiakas saa tietoja solmuista sekä päästään eri kutsuparametrien arvoihin ja niitä vastaaviin metodeihin, joilla saadaan seuraavat tiedot: solmut nykyisten solmujen alla (ROOM-11) tai vain lehdet nykyisen solmun alla (Temp, Moistness), tai kaikkiin alla oleviin solmuihin, kun paluuarvona saa avaimet solmuihin ja lehtiin litteässä nimiavaruudessa. Paluuarvoa voi rajata suodattamille, kuten vain lehdet joilla tietty VARIANT eli muuttujatyyppi, lehdet joihin vain lukuoikeus tai lehdet, jotka sisältävät vastaavan merkin. Hierarkkisessa nimiavaruudessa asiakaan on mahdollisuus liikkua läpi koko nimiavaruuden saadakseen tietoja liittyen uuteen solmuun. Metodit tukee tätä liikkumista solmusta solmuun ja kun palvelin tietää missä pisteessä asiakas kutsuu toista funktiota, voi palvelin palauttaa vastaavan arvon. Nimiavaruuden silmäilyn jälkeen DA asiakas luo ryhmäolion ja osio-oliot, joita käytetään rakenteessa. Luodessaan ryhmäolion, DA asiakas lähettää arvot seuraaviin parametreihin: olion nimi (symbolic name), lukutiheys (Requessted-UpdateRate), automaattisesti asiakkaille luettavat arvot (PercentDeadband), aktiivinen tila (Active-State). Jokaisella kutsulla asiakas voi luoda vain yhden ryhmäolion ja saa seuraavat parametrit: yksilöllisen polun nimiavaruuteen (fully qualifield ItemId), aktiivinen tila (ActiveState), kysytty tietotyyppi millä se haluaa arvon lukea tai kirjoittaa (RequstedDataType), osio-olioiden luontia varten tarvittavat tiedot (ClientHandle). Metodikutsun suorituksen jälkeen palvelin palauttaa parametrit oman oletukseni mukaan datatyypeistä (CanonicialDataType) ja osio- olioiden luonneista (ServerHandle). (Iwanits & Lang 20 2006: 30-33)

Kuva 3 kuvaa DA-palvelinrakennetta, jossa voidaan erottaa sovellusriippuvainen ja -riip-pumaton osa. Sovellusriippumatonta osaa ovat oliot OPC palvelin ja OPC ryhmäolio ja osa OPC osio-olioista, joiden sovellusriippuvainen osa kommunikoi Windows ajureiden kautta kenttäväylän kanssa. Prosessiarvoissa on paljon tietoja, kuten laitteiden nimet, jotka on esitelty solmuissa, kalibrointitietoja ja muuta jotka ovat kiinnostavia tietoja. Jokaisella ominaisuudella on yksi piste solmu tai lehti nimiavaruudessa ja jokai-

sella ominaisuudella on numeerinen tunnus (PropetyID), datatyyppi ja kuvaus englannin kielellä.



Kuva 3: DA-palvelimen rakenne (Iwanitz & Lange 2006: 28)

Arvoalueet ominaisuuksille (PropertyID) on jaettu ala-arvoalueisiin:

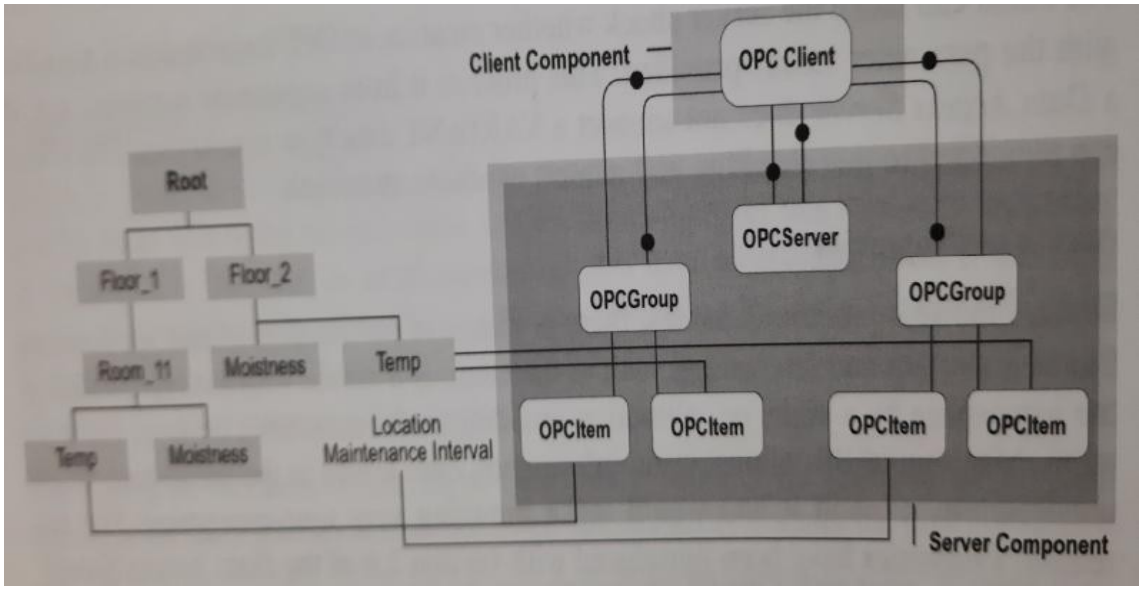
- 1-99 OPC- kohtaiset ominaisuudet, joilla on vaikutusta OPC määrittelyssä ja ovat kaikkien osio-olioiden käytössä, esimerkiksi tietotyyppi, pääsyoikeudet.

- 100-4999 suositeltavat ominaisuudet. Määrittelee ominaisuuksia, mitkä ovat ominaisuuksia ylemmälle arvoalueen ominaisuudelle. Kunhan ne on OPC määrittelyn mukaisia. Nämä ominaisuuden on tuettu palvelimella ja esimerkiksi 101 arvo kuvaa yksityiskohtaisemmin prosessimuuttujaa kuten ikkunan lämpötila huoneessa 11 (Temperature at window in Room 11).

- 5000-65535 toimittajakohtaisia ominaisuuksia, joihin voi lisätä valmistajan, käyttäjän tai muiden tietoja palvelimelle, esimerkiksi kalibrointitietoja (Iwanits & Lange 2006: 33-34).

Vasemmalla kuvassa 4 on osa nimiavaruudesta ja oikealla OPC oliot DA palvelimella. Root sisältää kolme solmua ja 4 lehteä, mitkä esittävät prosessin muuttujia. Yhdellä lehdistä on kaksi ominaisuutta ”Location” ja ”Maintanence interval”. Kaksi ryhmäoliota on luotu DA palvelimelle, joilla on kaksi osio-oliota, jotka ovat kohdistettu nimiavaruu-

den Item osiin ja siten prosessimuuttujiin. Täysin vastaava osio numero (fully qualified ItemId) voisi olla: ”Floor-1.Room_11. Temp”, joten ryhmäolio hallinnoi osio-oliota, joka sisältää lämpötila-arvon. Toinen ryhmäolio sisältää lämpötila-arvon ja tiedon kunnossapitovälistä. OPC asiakkailla on pääsy OPC olioihin palvelimella rajapinnan metodeilla. Tämä on esitetty kuvassa viivoilla, joissa mustat ympyrät. Kutsun takaisin vastauksen rajapinta on merkitty asiakasta lähemmillä mustilla ympyröillä. (Iwanits & Lange 2006: 34)

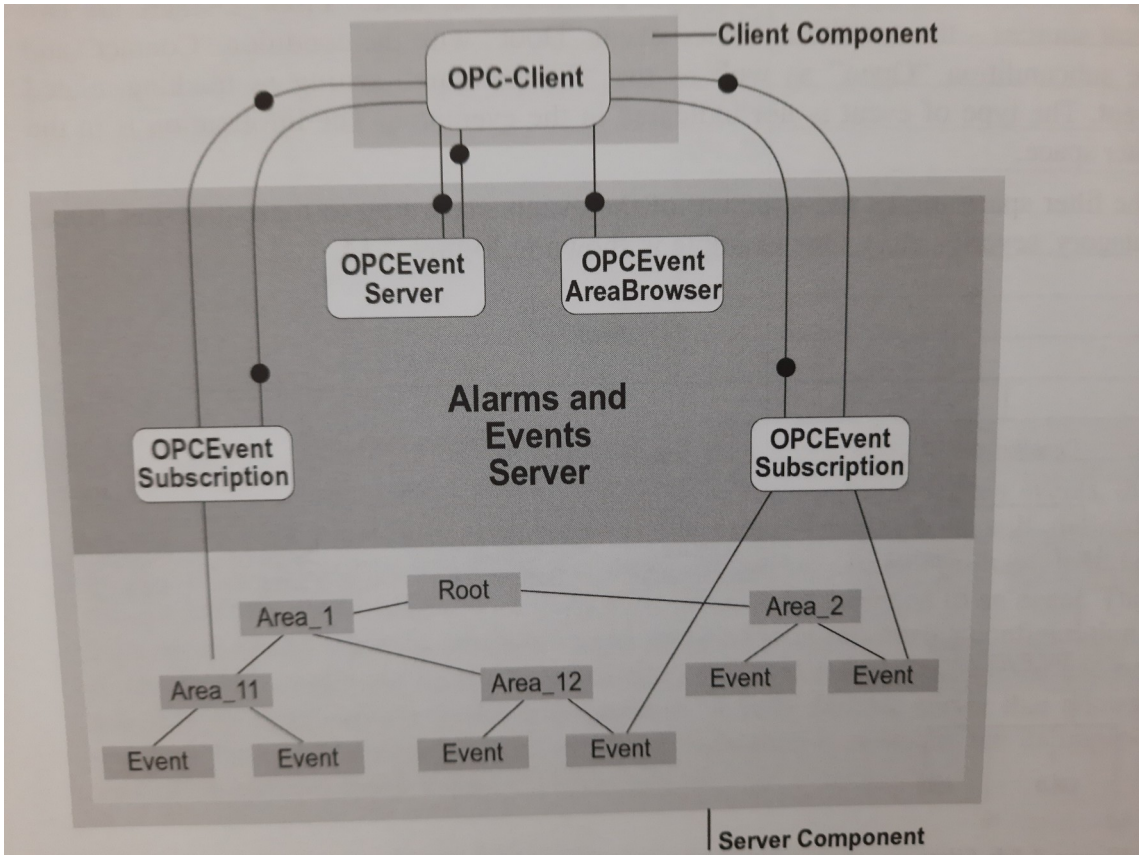


Kuva 4: Nimiavaruus ja oliohierarkia (Iwanitz & Lange 2006: 34)

3.2.2 Alarm and Event

OPC A & E määrittelee rajapinnan asiakkaan ja palvelimen välillä, mitä käytetään monitoroinnissa, lähetyksissä, tapahtumissa ja hälytyksissä. A & E palvelin voi tallentaa ja analysoida arvoja eri datalähteestä ja selvittää missä tapahtuma on esiintynyt. DA palvelin mahdollistaa jatkuvan tietovirran. Automaattinen arvojen siirto voidaan mukauttaa suhteelliseen arvojen muutokseen +/-1 yksikköä, mikä mahdollisuus esiintyy vain analogisten arvojen yhteydessä ja jos arvolla ei ole oliorakennetta. A & E palvelin ei lähetä itse arvoja asiakkaille vaan tiedon siitä, että jotain on tapahtunut, esimerkiksi lämpötila on ylittänyt raja-arvon. Tämä määritelmä on tehty, koska palvelin keskittyy laitteiden ja sovellusten muuttujien monitorointiin. Määrittely ei kuvaa kuinka tämä tulee tehdä. On mahdollista määrittää kriteerit kaikille muuttujille, niin analogisille, binäärisille ja struktu-

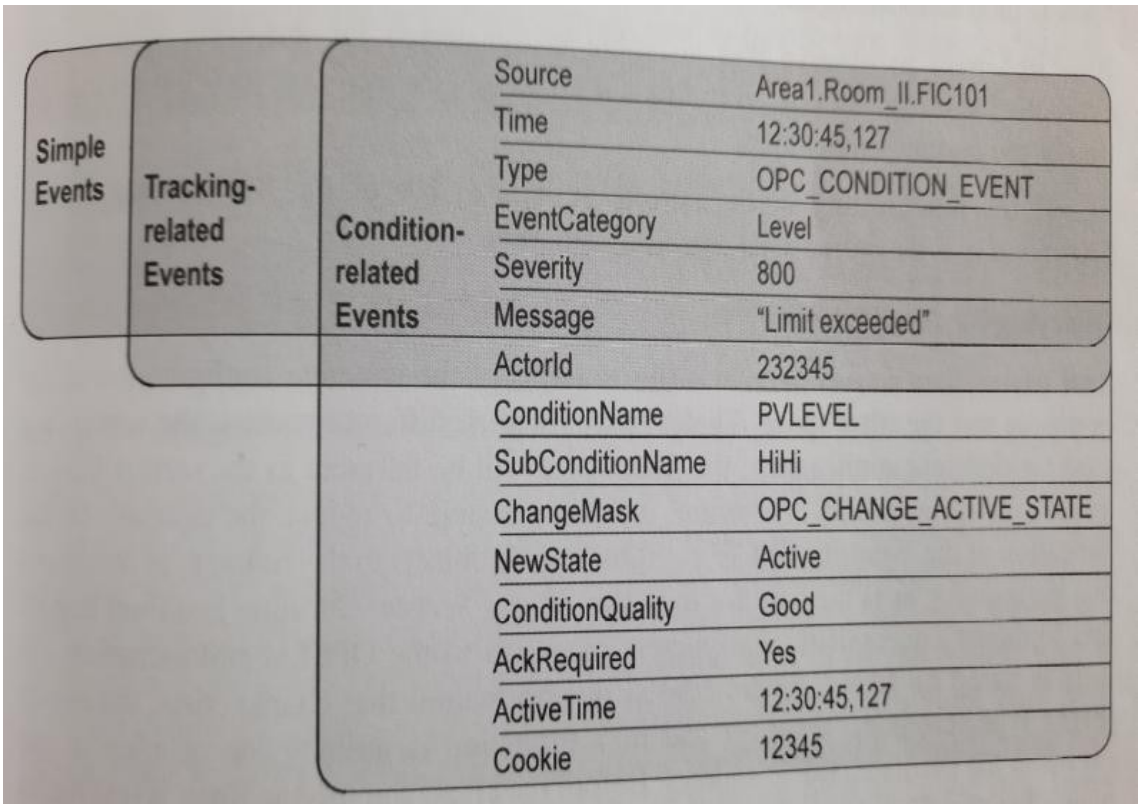
roiduille tietotyypeille, yksinkertaiselle datalle tai muuttujiin kiinnitettyihin arvoihin kuten lämpötilaan ja aikaan. (Iwanits & Lange 2006: 54)



Kuva 5: A & E palvelin - oliohierarkia- ja tapahtuma-alue (Iwanitz & Lange 2006: 58)

A & E asiakas voidaan kuvata yksittäisenä osana joka saa tietoa suoraan laitteilta ja sovelluksilta tai DA palvelimelta, ne ovat yhteydessä DA asiakkaan toimintaan. A & E asiakas voi olla yksinkertainen Excell-taulukko tai monimutkainen ohjelma osana ohjaussysteemiä. A & E määrittely tukee sitä, että asiakkailla voi olla pääsy useammalle palvelimelle ja yhdellä palvelimella voi olla monta asiakasta. Tapahtuma eli Event on havaittu ongelma, kuten laitevika, raja-arvon ylitys, operaattorin väliintulo. Useita tapahtumia voidaan kohdentaa niihin liitettyihin muuttujiin, esimerkiksi ylitettyyn lämpötilaarvoon voidaan liittää neljä raja-arvoa. Tämä tarkoittaa, että neljä tapahtumaa voi esiintyä ja jokainen niistä ilmoittaa, että ylempi tai alempi raja-arvo on ylitetty. A & E määrittely määrittää käsitteen olosuhteille ja aliolosuhteille, mikä liittyy siihen ja olosuhde on voi-

massa vain sen esiintymisen ajan. Kuvassa 6 on ominaisuuksia tapahtumille eri tapahtumatyypeissä. (Iwanits & Lange 2006: 54-59)

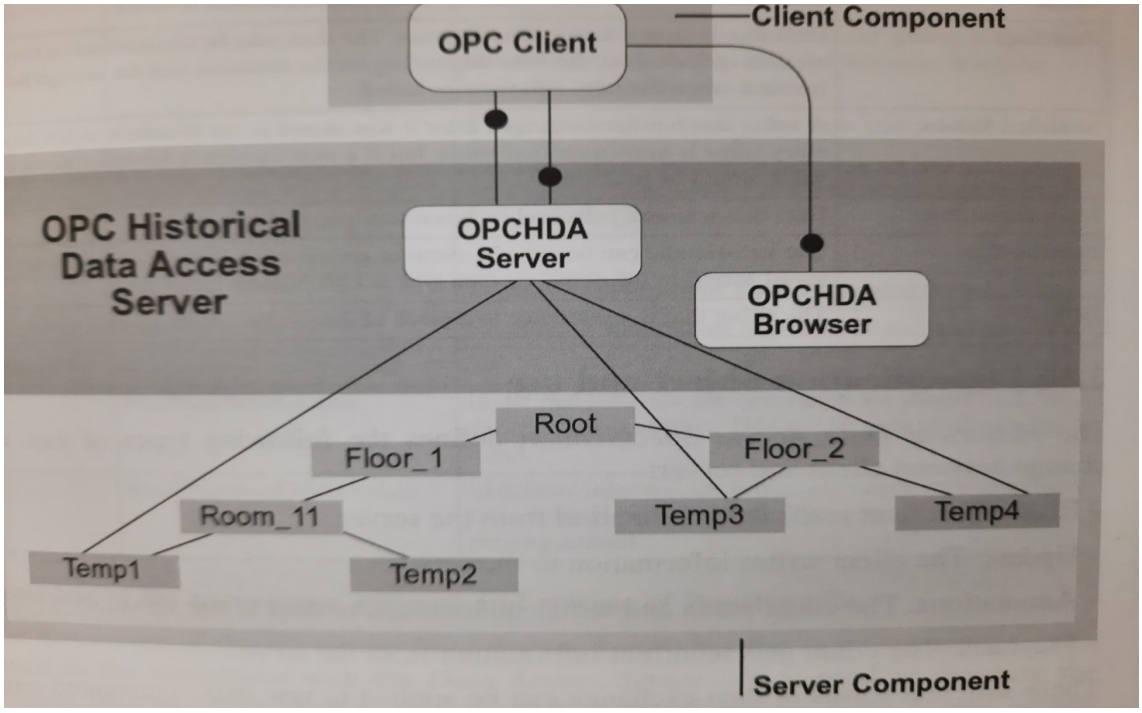


Kuva 6: Ominaisuuksia tapahtuma, Event, ilmoitukselle ei tapahtumatyypeissä (Iwanitz & Lange 2006: 59)

3.2.3 Historical Data Access

OPC HDA Specification mahdollistaa asiakkaille pääsyn historiatietoihin, mikä voi olla tietovarastoihin varastoitua raakadataa tai asiakkaan pyynnöstä järjesteltyä raakadataa, jota voidaan ylikirjoittaa, kommentoida tai siihen voidaan lisätä uutta dataa. OPC HDA mahdollistaa kaksi palvelintyyppiä, yksinkertaisen datapalvelimen, joka rajapinnalla toteuttaa vain rajapinnan ja pääsyn raakadataan tai monimutkaisen datapakkauksen ja analyyttisen datan käsittelyyn soveltuvan palvelimen, joka mahdollistaa data-analytiikan kuten keskiarvot, minimi- ja maksimiarvot, datan eheyttämisen, - lisäämisen ja historiantietojen lukemisen datamuutoksista. Palvelimen tietolähdettä ei ole määritelty, se voi olla vaikka datavarasto. Määrittelee kaksi rakennetta, nimiavaruuden ja oliohierarkian, joita HDA palvelin ja asiakas voi käyttää. Nimiavaruus sisältää kaiken historiadatan, johon palvelin tarjoaa pääsyn. Määrittely edellyttää vain olemassa olevaa hierark-

kista nimiavaruutta. DA palvelimen nimiavaruus on verrattavissa HDA palvelimen määrittelyn nimiavaruuteen (Iwanits & Lange 2006: 65-66).

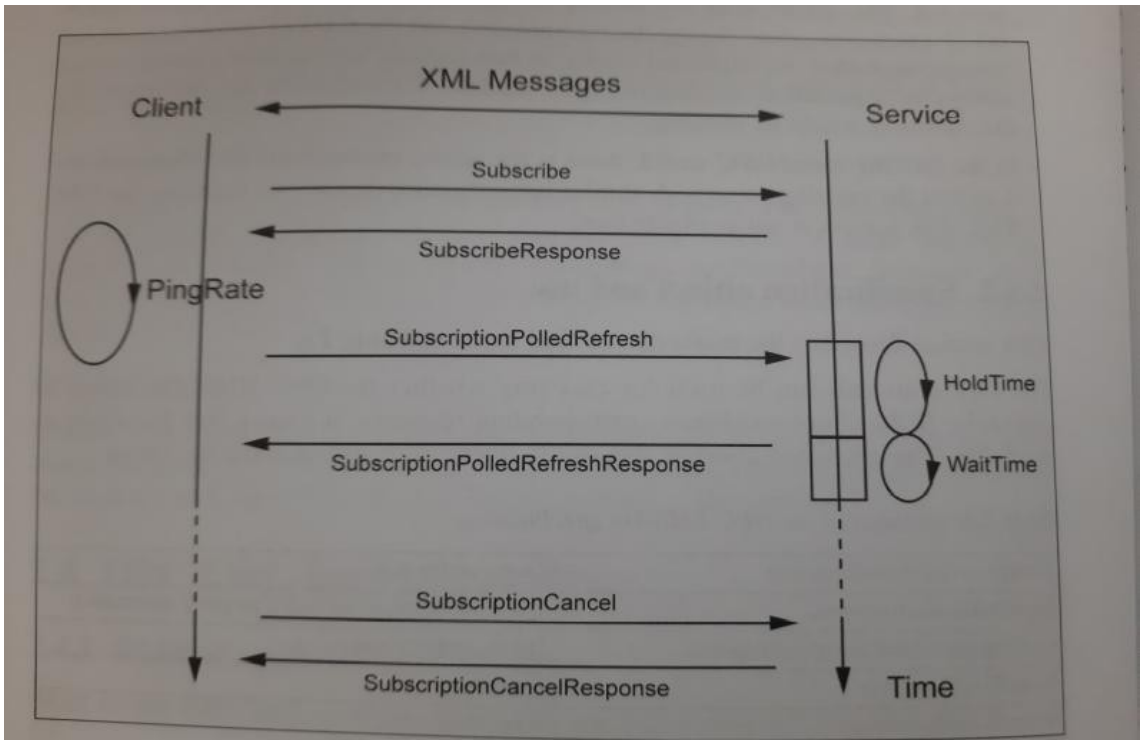


Kuva 7: HDA palvelin - nimiavaruus ja oliohierarkia (Iwanitz & Lange 2006: 67)

HDA asiakas voi luoda eri OPC olioita HDA palvelimelle. OPC HDA palvelin olio on ylimmän tason olio, joka tarjoaa täyden toimivuuden lukea, kirjoittaa tai muuttaa dataa ja niiden ominaisuuksia sekä silmäillä, läpikäyden nimiavaruutta. HDA palvelimella ei ole ryhmäolioon ja osio-olioihin verrattavia olioita, vaan asiakas antaa suoraan osoitteen data osiolle käsitelläkseen oliota luomatta oliota tähän tarkoitukseen palvelimelle. DA palvelin sallii pysyvän pääsyn prosessimuuttujiin, mikä rakenne on muotoiltu eri näkökulmasta ja pysyvä lisäys ja poisto osio-oliosta saman osion nimiavaruutta on aika poikkeavaa. Prosessimuuttujia on saatavilla DA palvelimelle 100-1000 tai vähemmän ja OPC asiakas haluaa lukea ja kirjoittaa vain yhden arvon prosessimuuttujaan toisin kuin HDA palvelimella prosessimuuttujia on tarjolla 1000-10000 ja asiakas ei kuitenkaan ole lukemassa näitä muuttujia jatkuvasti, mutta luultavasti päivittäin. Toisaalta asiakas ei ole kiinnostunut vain yhdestä ainoasta arvosta, vaan useammasta arvosta tai kootusta arvoryhmästä esimerkiksi keskiarvosta. Suuri joustavuus määriteltäessä yhdistelmiä on tullut tunnusomaiseksi piirteeksi HDA:ssa. (Iwanits & Lange 2006: 66)

3.2.4 OPC XML- DA

Automaation hallintaan ja prosessin tilaohjaukseen oli enenevässä määrässä alettu käyttää Intranettiä ja Internetiä, niin OPC Classiciin lisättiin 2000 luvun alussa OPC XML- DA määrittely. Nimiavaruus OPC XML- DA palvelimella sisältää kaikki muuttujat, joille Clientilla eli asiakkaalla on luku tai kirjoitusoikeus. Suuri määrä metodeja on OPC DCOM nimiavaruudessa tietojen läpikäynnissä yhdistetty yhteen metodiin. Yksi pyyntö asiakkaalta, jotta tiedot löytyisi, niin palvelin vastaa BrowseResponse vastauksella. Asiakas voi määritellä suodattimia, maksimi määrän osioita, ja tietojen silmäily tietojen löytämiseksi alkaa ja jatkuu toistuvilla pyynnöillä ja vastauksilla. GetProperties pyynnöllä, asiakas voi tiedustella arvoja ominaisuuksille. BrowseResponse asiakas saa tietoja olemassa olevista ominaisuuksista. Nimiavaruuden silmäiltyään asiakas voi lukea ja kirjoittaa arvoja ja se pääsee ItemIdentifier kautta suoraan vaadittuun muuttujaan ja sen arvoon. Palvelin lähettää arvon ja palauttaa tuloksen asiakkaalle, mistä riippuen asiakas voi kirjoittaa aikaleiman ja laatutiedon kuin myös tulevan arvon uudelleen. (Iwanits & Lange 2006: 44-46)



Kuva 8: Datan automaattinen siirto OPC XML- DA määrittelyssä (Iwanitz & Lange 2006: 46)

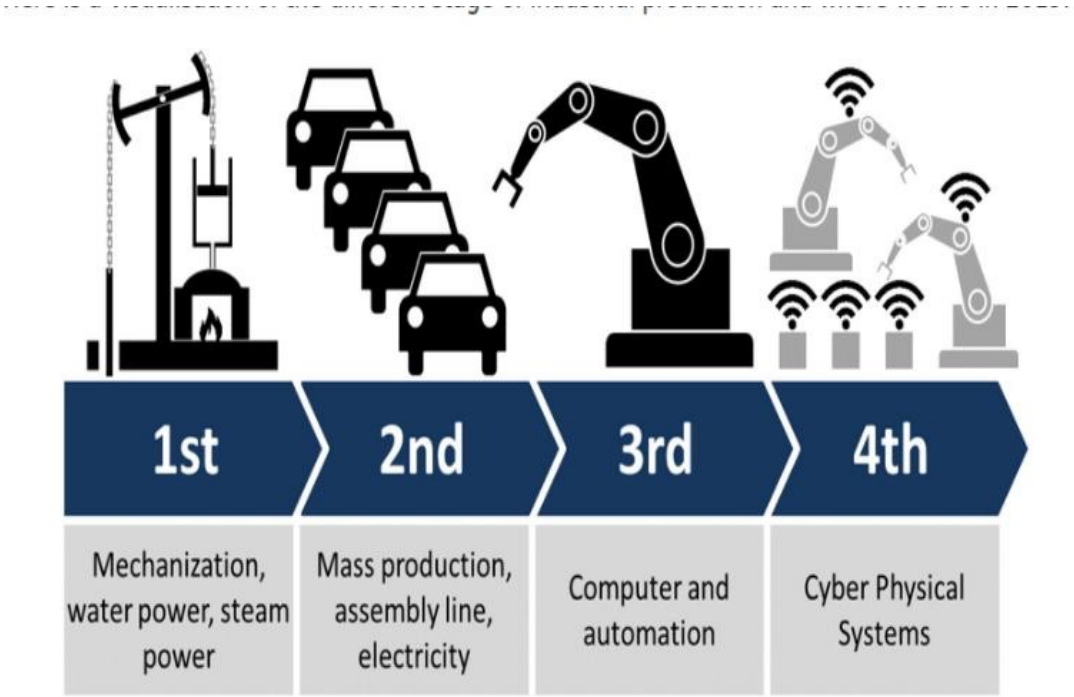
EnablingBuffering tarkoittaa, että asiakas ei voi tiedustella arvoja niin usein kuin RequestSamplingRaw sallii, vaikka palvelin saa arvoja useammin kuin asiakas tiedus-

telee niitä. Jos EnablingBuffering on TRUE, palvelin säilöö arvon ja lähettää sen vastauksena seuraavaan pyyntöön. Kun palvelin palauttaa SubricptionResponse vastauksen, se sisältää takaisin kutsupyynnön käsittelyn sekä kuinka usein arvoja pyydetään. Yhteyden katkaisu tapahtuu pyynnöllä SubscriptionCancelRequest ja palvelin palauttaa vastauksena SubscriptionCancelResponse.Jos aika ei muutu HoldTime määrittelyn aikana, niin arvon muutosta odotetaan WaitTime määrittelyyn ajan ja arvon muutos lähetetään heti. Jos WaitTime aika kuluu ilman muutosta, lähetetään tyhjä vastataus. Datan lähetys XML:llä ei ole yhtä tehokasta kuin bittidata lähetys DCOM:lla, siksi tulee olla tiedon määrän mukainen lähetystapa valittavissa. OPC DCOM DA määrittelee takaisin soiton, joita ei välttämättä ole määritelty internetpalveluissa, jolloin palvelin ja asiakas eivät toteuta oliomallia ja takaisin soiton loppupää puuttuu. Koska HTTP on tiedonsiirtoprotokolla, sillä ei välttämättä ole viimeisintä tietoa asiakkaasta. Osa HTTP laajennuksista säilyy tähän tarvittavan datan, mutta nämä laajennukset eivät ole välttämättä tuettuja. GetStatusCall pyynnöllä tarkistetaan saatavilla oleva OPC XML- DA saatavuus. (Iwanits & Lange 2006: 46-47)

4 OPC UA

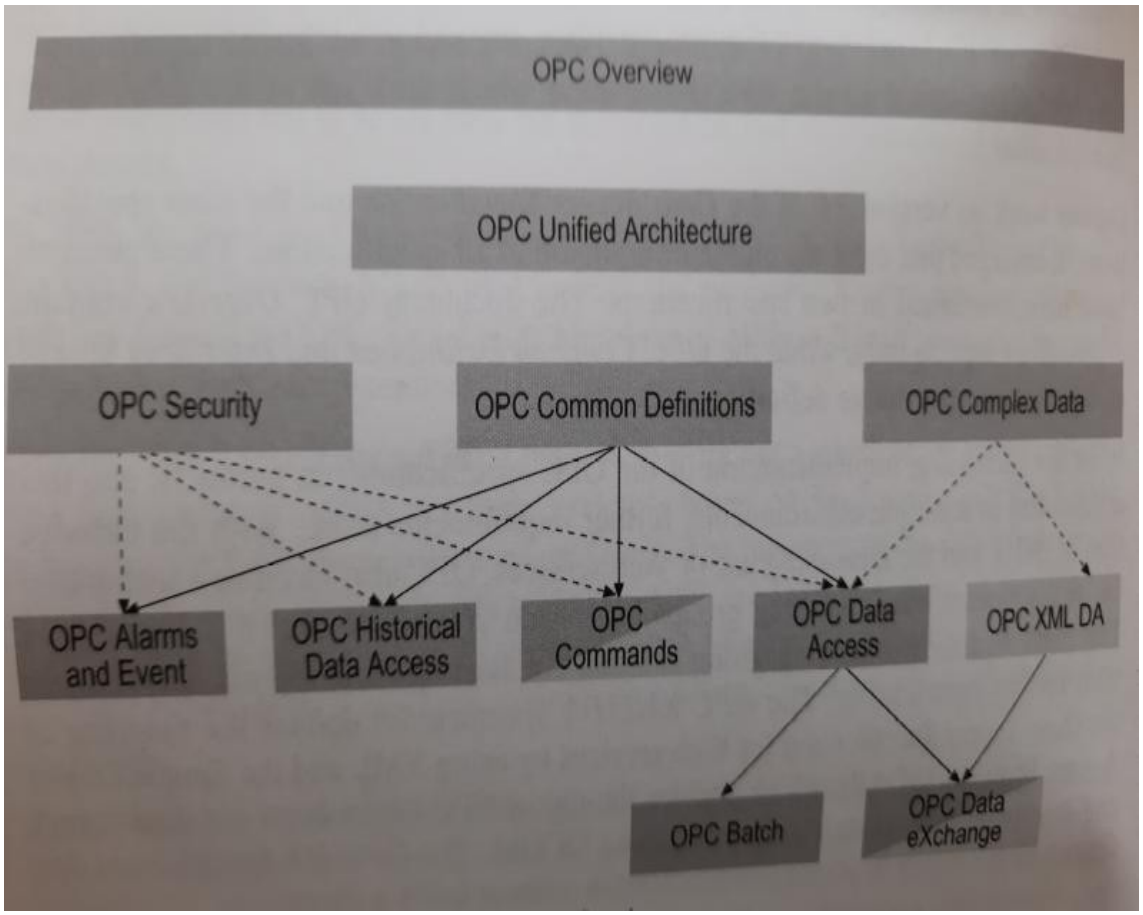
4.1 Historia

OPC Classic on ensimmäisenä väyläprotokolla standardina OPC UA:n historian kehitystä ja se on Windows järjestelmiin rajoittuva, sisäverkossa toimiva protokolla ja teknisen kehityksen myötä Internetin yli toimivat osat kehitettiin OPC UA:ssa toimivimmiksi kokonaisuuksiksi. Vuonna 2003 OPC Foundation julkaisi 13 osaa käsittävän OPC UA:n, joka vastaa paremmin teollisuuden haasteisiin ja on laajalti käytössä, nyt kun jo teollisuus 4.0 vallankumous on meneillään. Teollisuus 4.0 on Jyri Tulosen esityksen mukaan alkanut vuonna 2012 Saksasta ja hän kirjoittaa, että ”*Teollisuus 4.0 on teknologia, jolla voidaan fyysistä tuotantoa ja sitä ohjaavaa, verkon yli siirtyvää dataa ja IT-tekniikka yhdistää toisiinsa. Toisin sanoen, luodaan kyberfyysinen järjestelmä*. Vastaavasti täysin digitaalisessa järjestelmässä tuote valmistetaan asiakkaan pyynnöstä asiakkaan tilauksen mukaisesti täysin automaattisessa tehtaassa yksilöllisesti sarjatuotannon tapaan ilman ihmistyövoimaa. (Iwanits & Lange: 5, Tolonen 2018: Teollisuus 4.0)



Kuva 9: Teollisen valankumouksen vaiheet (Tolonen 2018: Teollisuus 4.0)

Vuonna 2005 OPC UA rakenne oli kuvan 10 mukainen. Kaikille OPC UA:n osille olevat yhteiset määrittelyt on yhdistetty OPC Overview- osaan. Tietoturva on OPC Security, OPC Common Definision sisältää yleisen rajapinnan ja yleiset määrittelyt. OPC Complex Data määrittelee mahdollisuudet esittää kompleksisen datan rakennetta ja pääsyä tähän dataan. OPC Classikiin jo kuuluneet OPC Alams and Event (AE), OPC Historical Data Access (HDA) , OPC Data Access (DA) ja OPC XML DA, joiden lisäksi OPC Commands eli määrittelyt rajapinnan ylittäville komentokäskyille ja ohjeistus niiden toteuttamiselle. OPC Batch, joka perustuu OPC Data Access määrittelyyn ja laajentaa sitä määritellen rajapinnalle pääsyn ja eräajon vaatimukset. OPC Data eXchange määrittelee yhteydenpidon palvelimen ja palvelinprosessien välillä. Itse OPC Unified Architecture määrittelee alustariippumattoman pääsyn dataan DA, AE ja HDA palvelimille web palveluja käyttäen. Kaikki muut osat olivat julkaisussa 1.0 (Release 1.0), ainoastaan OPC Batch on julkaisusta 2.0 (Release 2.0) ja OPC UA osa oli Draft eli luonnosteltavana edelleen. (Iwanits & Lange 2006: 6-7)



Kuva 10: OPC UA rakennekuvaus (Iwanitz & Lange 2006: 6)

4.2 OPC UA määrittelyt ja standardit

OPC UA on kasvanut kolmestatoista osasta 24 osaan kuva 11 (Damm 2022: 6). Uusimmat osat ovat tulleet vuonna 2021-2022 ja ne oheisessa kuvassa 12 vihreällä värillä merkitty. Osa osista muutettu ja merkitty vaaleansinisellä ja osa on olemassa olevia osia, merkitty tumman sinisellä, ja joidenkin osien nimi on muuttunut käyttämäni OPC oppaan sisällöstä toiseksi kuten Alarm ja Event on Alarm Conditions, joten en uudelleen kirjoittanut OPC UA osia, mitkä esitelty OPC Classik:in yhteydessä eli A & E, HDA, DA. Nyt OPC UA on esitelty Suomen Automaatioseuran OPC päivillä OPC 10000 UA:na, joka vastaa paremmin IIoT aikaa (Damm 2022: 13).

OPC FOUNDATION

OPC Number	Title	1.03	1.04	1.05.00	1.05.01	1.05.02
10000-1	Part 1: Overview and Concepts	✓	✓			✓
10000-2	Part 2: Security	✓	✓			✓
10000-3	Part 3: Address Space Model	✓	✓	✓	✓	✓
10000-4	Part 4: Services	✓	✓	✓		
10000-5	Part 5: Information Model	✓	✓	✓	✓	✓
10000-6	Part 6: Mappings	✓	✓	✓	✓	
10000-7	Part7: Profiles	✓	✓			✓
10000-8	Part 8: DataAccess	✓	✓	✓		
10000-9	Part 9: Alarms and Conditions	✓	✓		✓	✓
10000-10	Part 10: Programs	✓	✓	✓		
10000-11	Part 11: Historical Access	✓	✓			
10000-12	Part 12: Discovery and Global Services	✓	✓			✓
10000-13	Part 13: Aggregates	✓	✓			✓
10000-14	Part 14: PubSub		✓		✓	✓
10000-15	Part 15: Safety		✓	✓		✓
10000-16	Part 16: State Machines			✓		✓
10000-17	Part 17: Alias Names		✓	✓		
10000-18	Part 18: Role-Based Security			✓	✓	✓
10000-19	Part 19: Dictionary References			✓		✓
10000-20	Part 20: File Transfer			✓	✓	✓
10000-21	Part 21: Device Onboarding					✓
10000-22	Part 22: Base Network Model				✓	✓
10000-23	Part 23: Common ReferenceTypes				✓	✓
10000-24	Part 24: Scheduler					✓

Kuva 11: OPC UA osat (Damm 2022: 6)

OPC UA 1.05 julkaisun tilanne (Damm 2022: 6)

1.05.00 release completed (10/2021)

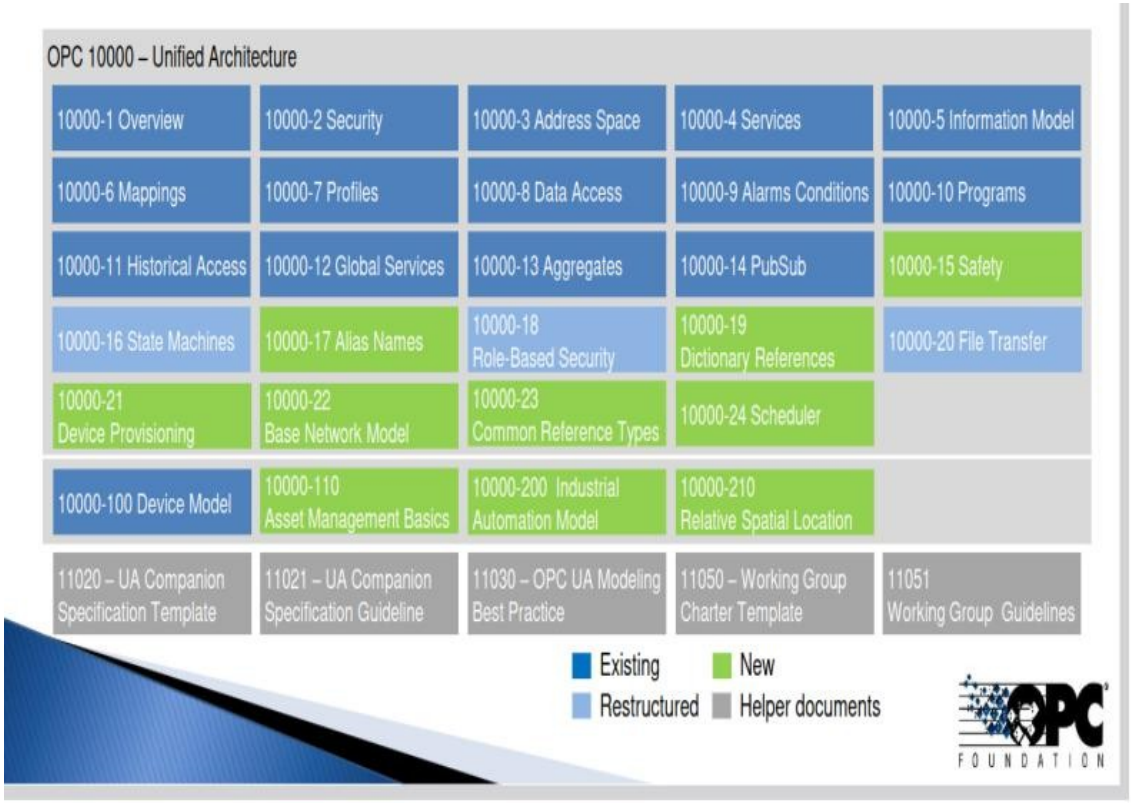
- Jakaa 10000-5 osiin; 10000-16 State Machines, 10000-18 Role-Based Security, 10000-20 File Transfer

1.05.01 julkaisu valmistui (03/2022)

- Uudet 10000-22 Base Network Model, 10000-23 Common Reference-Types features 10000-14 (PubSub) FLC/FX:lle ja QoS/TSN:lle

1.05.02 julkaisu valmistui (10/2022)

- Uuden 1.05.02 julkaisun kanssa kaikki osat saavat 1.05 version, paitsi osa 11 Historical Access (tulee versioon 1.05.03), 10000-21 Device Onboarding, 10000-24 Scheduler ja useita selvennyksiä ja parannuksia 10000-12 (Discovery and Global Services)



Kuva 12: OPC UA 10000 osien uudistuminen (Damm 2022: 13)

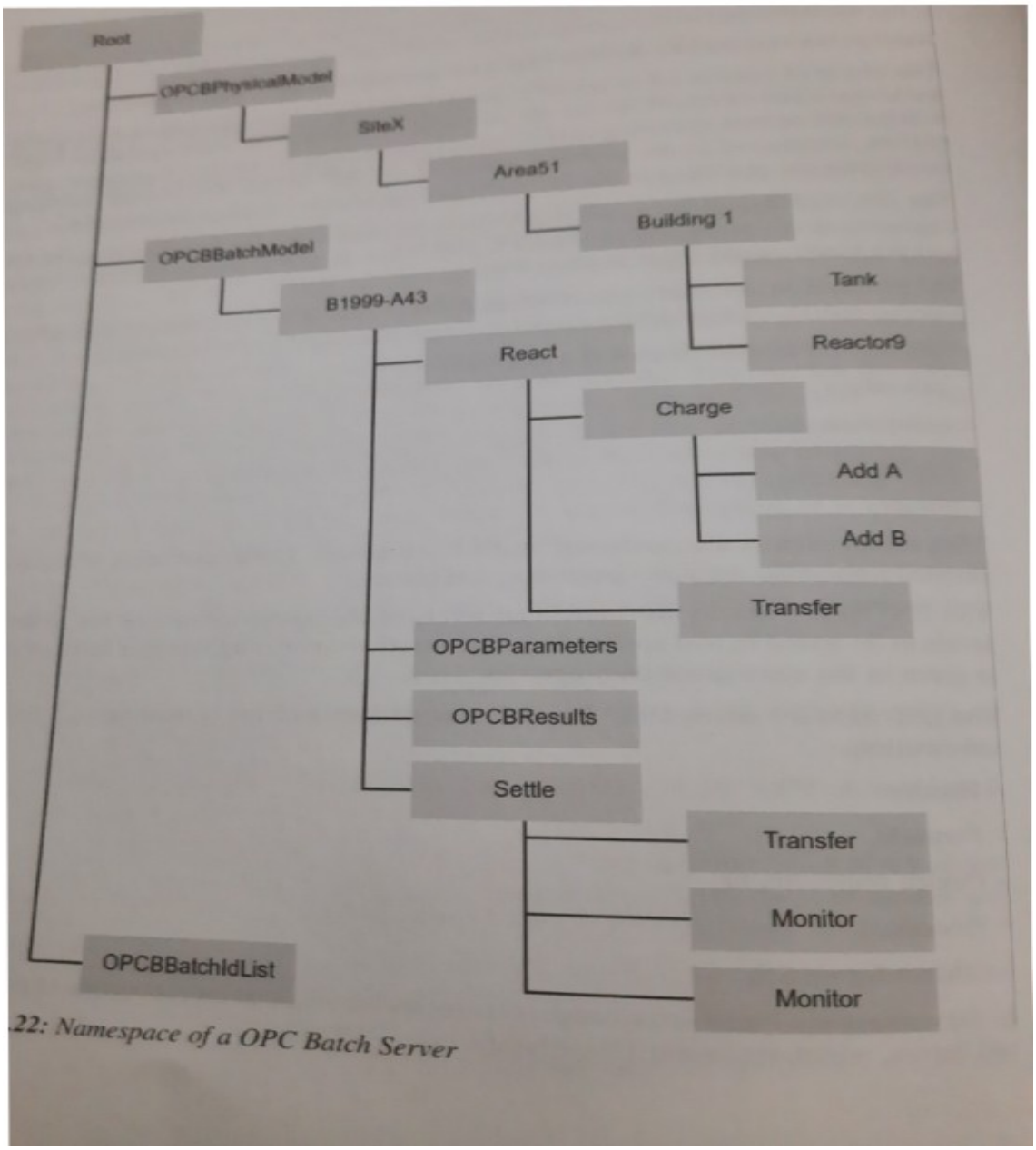
4.2.1 OPC Batch

OPC Batch käsitellään IEC standardeissa nykyisin kuvan 11 osissa (Parts) 1, 2, 16, 17, 18, 21, 22, 23 ja 24. Batch tarkoittaa eräajoa ja on tietokenttä sovelluksille ja malleille

ei kuvannut erillisenä osana täysin uutta rajapintaa asiakkaalle ja palvelimelle. Se määrittelee lisäykset DA:lle (Data Acceess) prosessin eräajoon, mikä sisältää eri ohjeita kuvaukselle tuotteiden valmistuksessa ja datan tiedot muuttuvat, kun tuote valmistuu ja ohjedata on lähetetty sekä raportti datasta saatu. Eräajon mukainen standardi oli jo IEC 61521-1, joka kuvaa eräajoprosessin toiminnan sisältäen raportin luonnin, prosessin- ja laitteen ohjauksen, sekvenssin ja sen raportoinnin luonnin. OPC Batch ei kuvaile ratkaisua eräajon ongelmiin, mutta kuvaa mahdollisuuksia ratkaisuihin eri valmistajille erilaisissa ympäristöissä. OPC Batch palvelin omaa seuraavat ominaisuudet:

- Tukee kaikkia pakollisia rajapintoja ja on vaihtoehto DA määrittelyn rajapinnalle
- Tukee rajapintoja, jotka on määritelty OPC Batch määrittelyssä
- Sillä on olemassa oleva määritelty nimiavaruus, rakenne ja pääavain, mitkä on jo määritelty standardissa IEC 61512-1 ja pystyy luomaan suuren määrän dataa, joka vaihdettavissa eräajossa, ja sen nimiavaruus voi olla hyvin laaja.
- Osien arvot nimiavaruudessa on luettavissa OPC Batch palvelimella, optiona on arvojen kirjoitus.

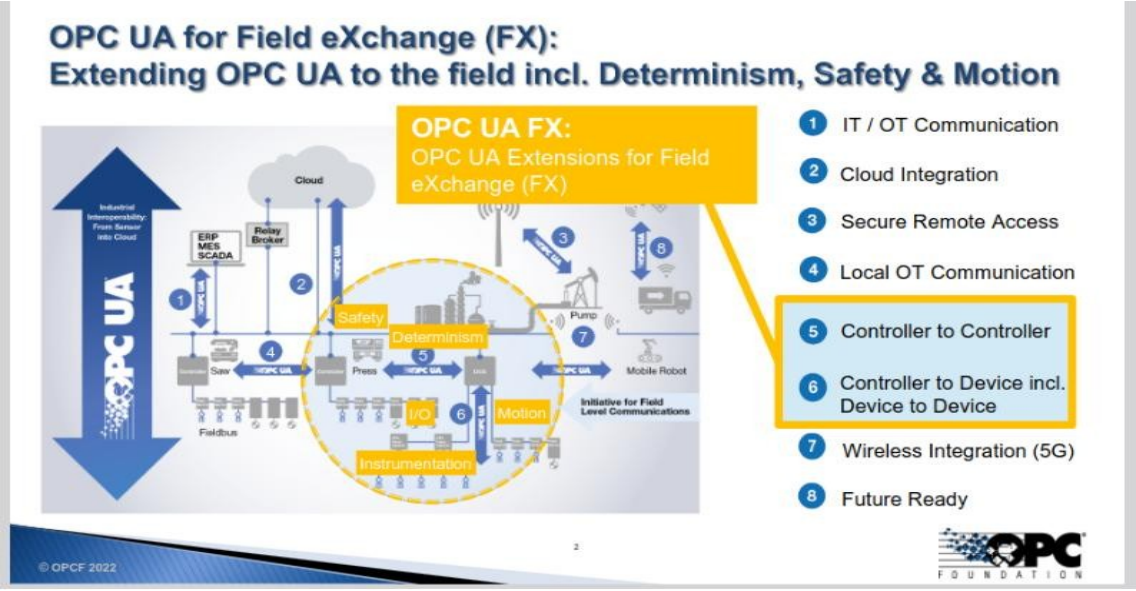
Standardi sisältää neljä tietoa eli laitteiden kapasiteetin, nykyisen toimintakunnon, historia- ja ohjetietojen sisällön. Eräajo (Batch) tehdään laiteella, josta on kuvattu laitekapasiteetti. Batch palvelin sisältää 5 tyyppistä dataa eli datalla on otsikko (Header), kaava tai malli (formula), laitevaatimukset, menettelytavan ja muut tiedot. Näistä kaikista tiedot on liitetty ominaisuuksiksi solmuihin ja lehtiin nimiavaruudessa, mikä vastaa menettelytapaa ohjainmoduuleissa. Esimerkkikuvana kuva 13, jossa nimiavaruus on OPC Batch palvelimella ja osa sen haaroista edustaa mallia IEC61512-1 ja ne on sijoitettu solmuihin OPCPhysicalModel ja OPCBatchModel. OPC Batch palvelin mahdollistaa pääsyn koko yritys prosessiin ”Site X”, jossa on ”Area51” saatavilla, missä ”Building_1”, johon on sijoitettu ”Tank” ja ”Reactor9”. Käytännön sovelluksissa kauemmat solmut voivat olla saataville laitteet ja toimintayksiköt ”Tank” ja ”Reaktor9”. Kaikki tiedot on sisällytetty OPCPhysicalModel haaraan. OPCBatchModel haara sisältää yhden haaran ”B1999-A43”, mikä liittyy kahteen menettelytapaan ”React” ja ”Settle”, jotka sisältävät operointivaiheen. Parametrit ja tulokset voivat olla olemassa jokaisella tasolla nimiavaruuden hierarkiassa OPCBatchModel haarassa. Parametreja käytetään lähettämään laitteilta tietoja operaatiosta ja tuloksena saadaan käytetyt, pyydetyt arvot laitteilta. (IEC: 61521, Iwanitz & Lange 2006: 74-77)



Kuva 13: Tehdasalueen puurakenne (Iwanitz & Lange 2006: 76)

4.4.2 OPC Data eXchange

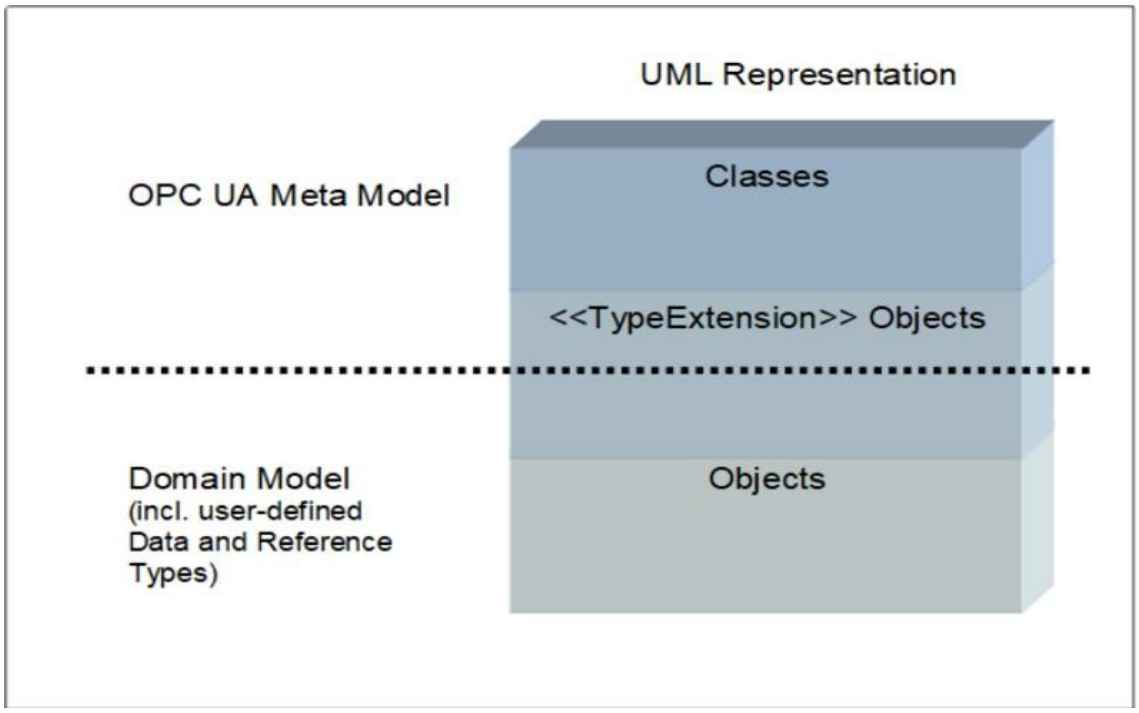
OPC DA:n (Data Access) palvelimen täytyy ajoittain vaihtaa dataa suoraan, esimerkiksi tietoja täytyy lähettää suoraan tuotantoalueelta toiselle, mikä vaatii asiakkaiden (Client) käyttöä tai mahdollisia siltoja, mitkä tekevät tietojenvaihdosta hidasta ja se ei sisälly kaikkiin käytettävissä oleviin ratkaisuihin. Lisäksi iso osa kenttäväylistä käyttää TCP/IP protokollaa Ethernetissä siirtämään dataa, niin asiaa on ratkaistu aiemmin OPC DX:llä, nyt OPC FX eli OPC Field eXchange. OPC DX sovelluksessa palvelin vastaanottaa dataa yhdeltä tai useammalta DA palvelimelta tai DX palvelimelta. DA asiakaspalvelut toteutuu myös DX palvelimessa ja tämän asiakkaan DX palvelin luo OPC ryhmän (OPC-Group) ja OPC osien (OPCItem). Lähtöpiste datalle on lähdeosa, ja piste, mihin data kirjoitetaan, on kohdeosa, ja tämä kokonaisuus on yhteys määrittelyssä. DX konfiguraatio sisältää kaikki yhteydet, jotka on määritelty asiakas konfiguraatiossa ja liitetty nimiavaruuteen DX palvelimella. Asiakasta monitoroidessa, monitoroi myös yhteyttä, mikä on tehty yhdessä DA:n kanssa. Määrittelyssä erotetaan ajonaikainen ja konfiguraatioosa yhteydelle. DX:n ajonaikainen horisontaalinen datanvaihto on myös UA FX:ssä, joka on nykyisin käytössä. (Iwanitz & Lange 2006: 49, OPC Foundation: UA FX)



Kuva 14: OPC UA FX sijoittuminen tehdasympäristössä (Lutz 2022: 2)

Nykyinen malli Field eXchange, FX, on oheisessa kuvassa 14 ja näemme kohdat, missä sitä voi hyödyntää tuotantoympäristössä. FX on ohjain ohjaimille ja laiteille, myös laitteilta laitteille yhteydellä. Kuvassa 90 näemme FX-pinon arkkitehtuurin, jossa FX-osat on merkitty keltaisella ja UPC UA:n osat sinisellä. Turvallisuus on määritelty FX:n

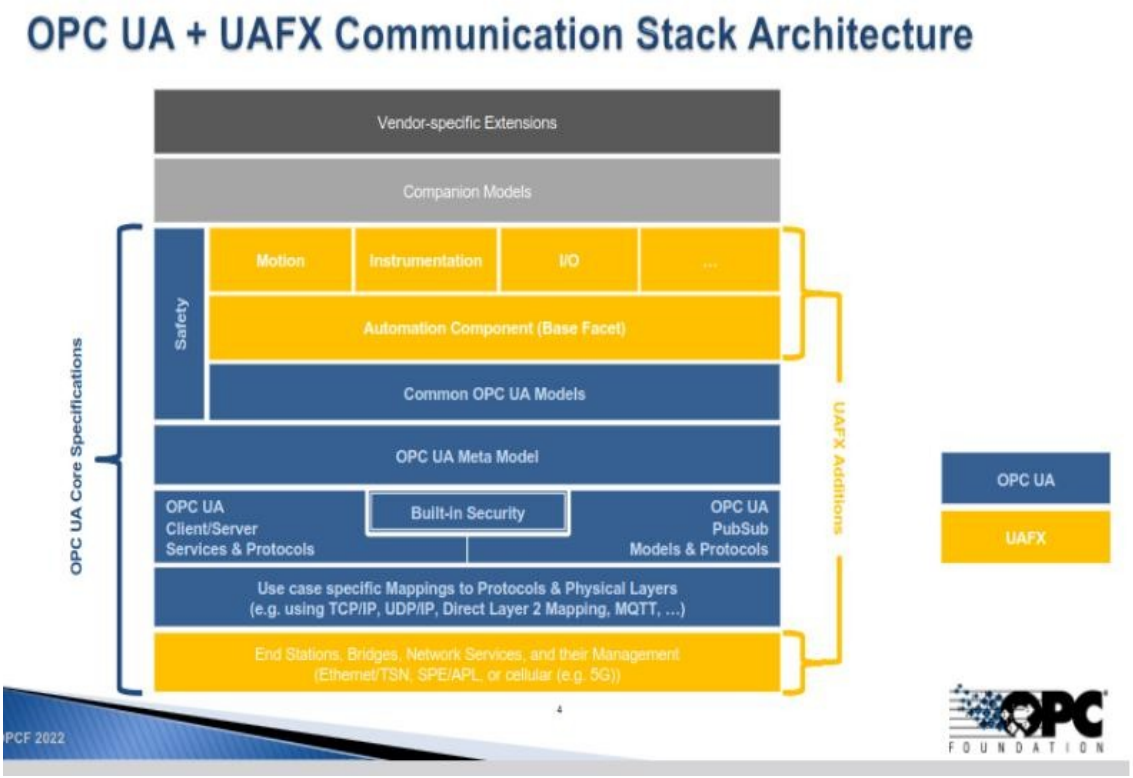
osille vasemmalla OPC UA:n osana ja se kattaa keltaisen osan FX käyntitiedot, instrumentointi, lähdöt ja tulot, jne, mitkä ovat automaatiokomponenttien yläpuolella, turvallisuuden jatkuen yhtenäisenä OPC UA normaaliin malliin asti. Tämän yläpuolella on kumppanimalli vaaleanharmaana ja ylimpänä toimittajan mukaiset erityislaajennukset tumman harmaalla. OPC UA Meta Modell on osan (the OPC UA Address Space Model) UML esityksen osa UML luokan ja UML olion väliltä merkittynä «Type Extension» olioksi, joka jää UML lukan ja UML olioiden väliin, siten että UML oliot edustavat DataTypes ja ReferenceTypes- tyyppiä ja Oliot jotka kuuluvat domain malliin voivat sisältää käyttäjämääritellyn ReferenceTypes ja DataTypes-tyypit, mitkä on myös merkitty «Type Extension»:lla. Domain Model sisältää ObjectTypes ja VariableTypes esittämään UML oloita. Kuva 91 (Lutz 2022: 3, OPC Foundation: Meta Model)



Kuva 15: OPC Meta Model ja UML:n Domain rajapinnan esitys (OPC Foundation: Meta Model

OPC MetaModel alapuolella on vasemmalla asiakkaat ja palvelimet, palvelut ja protokollat, oikealla OPC UA PubSub mallit ja protokolla. Tämän alapuolella käyttötapaukseen liittyvät protokollat, kuten TCP/IP, UDP/IP, suora liityntäkerros, MQTT jne. ja fyysiset kerrokset. Tämän alapuolella on FX kerros siltoineen, loppupisteineen, verkkopal-

veluineen, ja niiden hallinta. Esim. Ethernet/TSN, SPE/APL, solu, kuten 5G. Itse julkaisu UAFX määrittely sisältää neljä osaa kuvaamaan tiedonvaihtoprosessia ja konfiguroitua datan välitystä automaatiokomponenttien välillä käyttäen osia OPC UA asiakas/palvelin ja PubSub kompinaationa päästä-päähän yhteyttä ja perusdignostiikkaa. (OPC Foundation: UAFX).



Kuva 16: Pinorakenne OPC UA + UAFX arkkitehtuurissa (Lutz 2022: 3)

4.2.3 OPC Complex Data

DA-palvelin (Data Acces Server) tukee yksinkertaista ja kompleksista dataa. Lämpötila on esimerkki yksinkertaisesta datasta ja diagnostinen data on esimerkki kompleksisesta datasta. Molemmat datatyypit on voitu lukea OPC DCOM ja XML- DA:lla. Asiakkaan (Client) tulee ymmärtää yksilöllisiä elementtejä kompleksisesta datasta. Ymmärtää myös datan semaattisuuden, jolloin asiakaan tulee erottaa osat ja myös tietää datan tyyppi ja suhde toisiinsa. Datan tyypin kuvaus on määritelty OPC:n ulkopuolella esimerkiksi kenttäväylän dokumentaatiossa. OPC:n ovat liittyneet määrittelyyn kompleksisesta datasta. Kompleksinen data on määritelty nykyisin standardin IEC62541-3 ja käytetään myös standardin IEC61131-3 PLC toimittajille UA asiakkaina kanssa. Toiminnallisuuden toteutus ohjaimessa mahdollistaa käynnistää yhteyden yllä pitäminen minkä

tahansa saatavilla olevaan OPC UA palvelimeen ja ohjain voi vaihtaa kompleksista dataa rakenteessa horisontaalisesti muiden ohjaimien kanssa käyttäen kenttäväylää tai vertikaalisesti muiden laitteiden kanssa käyttäen OPC UA palvelimen palvelu arkkitehtuuria, kuten MES/ERP järjestelmää ensin keräämällä dataa määräyksistä tai kirjoittamalla uuden tuotanto määräyksen pilvipalveluun. (Iwanitz & Lange 2006: 50-51, OPC Foundation: IEC61131-3, IEC: IEC62541-3)

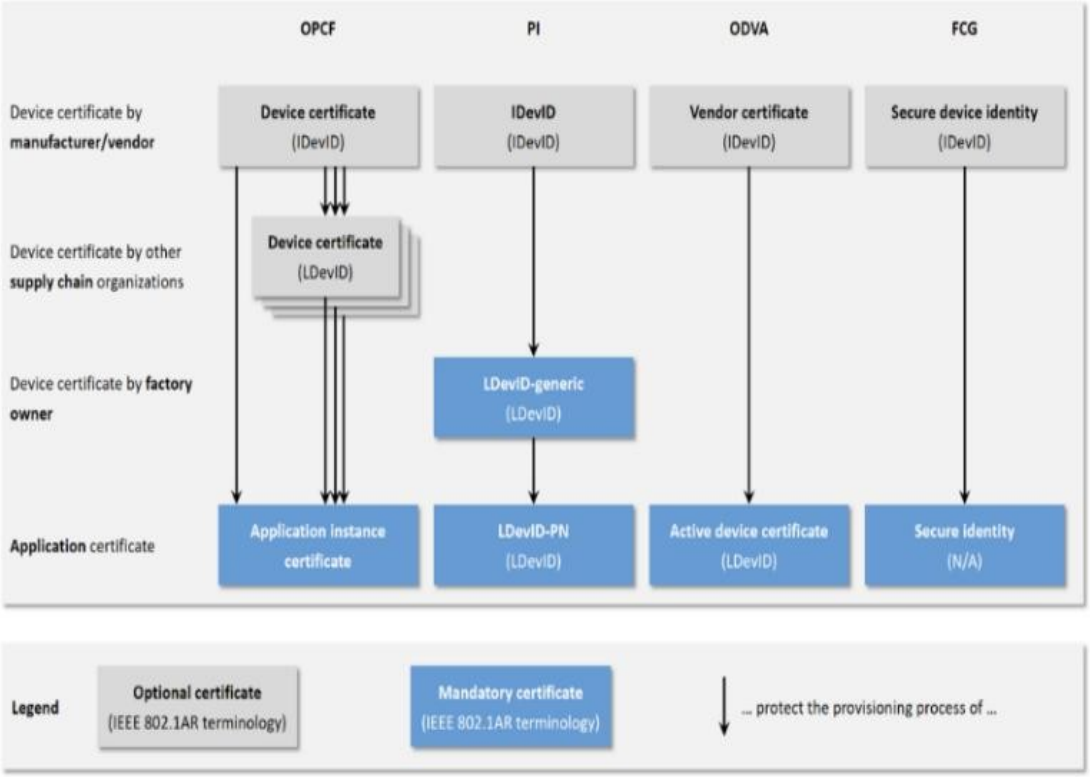
4.2.4 OPC Command Execution

Sovelluksille, ei ole ainoastaan arvojen lukeminen ja kirjoittaminen tärkeää, vaan myös käskyjen suorittaminen ja ne voivat olla periaatteessa liitetty osioihin (Items). Asiakas (Client) tulee kirjoittaa arvot tarkasti ja arvioida arvojen luku. Tämä lähestymistapa ei täysin vastaa käskyjen semaattisuutta ja käskyjä voidaan suorittaa rinnakkain sekä käyttää eri tulo parametreja. Käskyjen toteutus perustuu useimmiten laitteen tilaan, jonka muutokset ovat tärkeitä, ei ainoastaan yhteyksissä dataan pääsyyn, vaan hälytys ja historian datassa. OPC määrittelyt on sisällytetty OPC standardeihin ja käsky määrittely on ollut itsenäinen määrittely sekä COM rajapinnalle ja web palvelimille, mutta sisältyy nyt IEC TR 62453-42 osaan (IEC: standardi TR 62453-42) ja toteutunut yhdessä DA, A & E ja HDA kanssa. XML on käytetty kuvaamaan käskyjen ominaisuuksia. Kun asiakas saa haltuunsa tiedot saatavilla olevista käskyistä, mitä se voi ne suorittaa, joko synkronisesti tai asynkronisesti, niin asiakas voi pyytää ilmoituksen, jos tila muuttuu asynkronisessa käskyn suorituksessa yhteystilassa. (Iwanitz & Lange 2006: 51-51)

4.2.5 OPC Security

OPC UA turvallisuus (Security) perustuu teollisuuden Ethernetin turvallisuus konseptiin, jota muokkaavat säännöllisissä kokouksissa Ethernet Security Harmonization Group ja sen jäsenet OPC Fountation, ODVA Inc. Profibus & Profinet Internetional sekä FieldComm Group. Heidän päämääränsä on saada loppukäyttäjän turvallisuusprotokollasta helpommin käytettävä, monimutkaisten ratkaisujen sijasta. Varmistaakseen teollisuuden turvallisuuden joka tasolla käytetään kryptografisia avaimia ja julkinen avain ja yksityinen avain muodostavat avainparin, jolla yhteys muodostetaan. Julkinen (Public-key) on pakattu elektroniseen dokumenttii, serfifikaattiin, joka identifioi avainparin omistajan. Yksityinen avain on turvallisesti talletettu sovelluksen koneeseen ja siihen on vain omistajalla pääsy. (OPC Foundation: Security)

Julkinen avain toiminnot (Public-Key infrastucture, PKI) eli kommunikointi kumppanin kanssa. Sovellus tarvitsee sertifikaatin kumppanilta, turvallisuus protokollan mukaista kommunikaatiota varten, sisältäen mekanismin sertifikaatinvaihtoa varten ja se on turvallista, kun molemmat todentavat sertifikaatin saannin kumppaniltaan. Tätä toimintoa varten on kolme funktiota, sertifikaatin luontiin (Certification Authority, AU), sertifikaatin rekisteröintiin (Registration Authority, RA) ja sertifikaatin uudelleen latauslista (Certification Revocation List, CRL). AU vastaa sertifikaatin allekirjoituksesta, joka tehdään RA:n pyynnöstä ja allekirjoitetut sertifikaatit edustavat sijoitettu sertifikaatti hierarkiaan AU sertifikaateissa. RA on funktion, mikä sertifikaatti annetaan pyydettäessä ja turvallisuuspolitiikka voi olla automatisoitua, jolloin sertifikaatti pyynnön tulessa, sertifikaatti lähetetään turvallisuudesta vastaavan pääkäyttäjän kännykkään. AU ja RA voidaan sijoittaa samaan koneeseen, mutta ne ovat loogisesti erillään olevia funktioita.
(OPC Foundation: Security)



Kuva 17: Sertifikaatit OPC UA (OPC Foundation: Security)

Julkisella avaimella on jatkuva seurantavaatimus ja aikaraja saattaa tulla vastaan, jolloin sertifikaatti täytyy ”herättää” ja tätä tarkoitusta varten on CLR, mikä identifioi AU allekirjoittaman sertifikaatin, mikä ei enää ole validi ja se herätetään RA:n toimesta ja

CLR toimii sekä off- että online yhteydellä. Yrityksen AU voi tallettaa varmaan talteen yksityisen avaimen ja varmistaa, että sitä on vaikea vahingoittaa ja jos se vahingoittuu vai sitä yksityistä avainta koskeva linja vahingoittuu ja yrityksen AU voi herättää AU, jota käytetty tuotantolinjan suojaukseen. Luetettujen sertifikaattien listalle pääsee vain, jos AU on luotettujen listalla. Sertifikaattityyppejä on neljä, joista kolme on laitteille ja yksi sovelluksille. Sertifikaattien hallintaan on työkalu ja kuvassa 17 näkyy eri sertifikaattityypit ja mihin ne kohdistuvat. (OPC Foundation: Security)

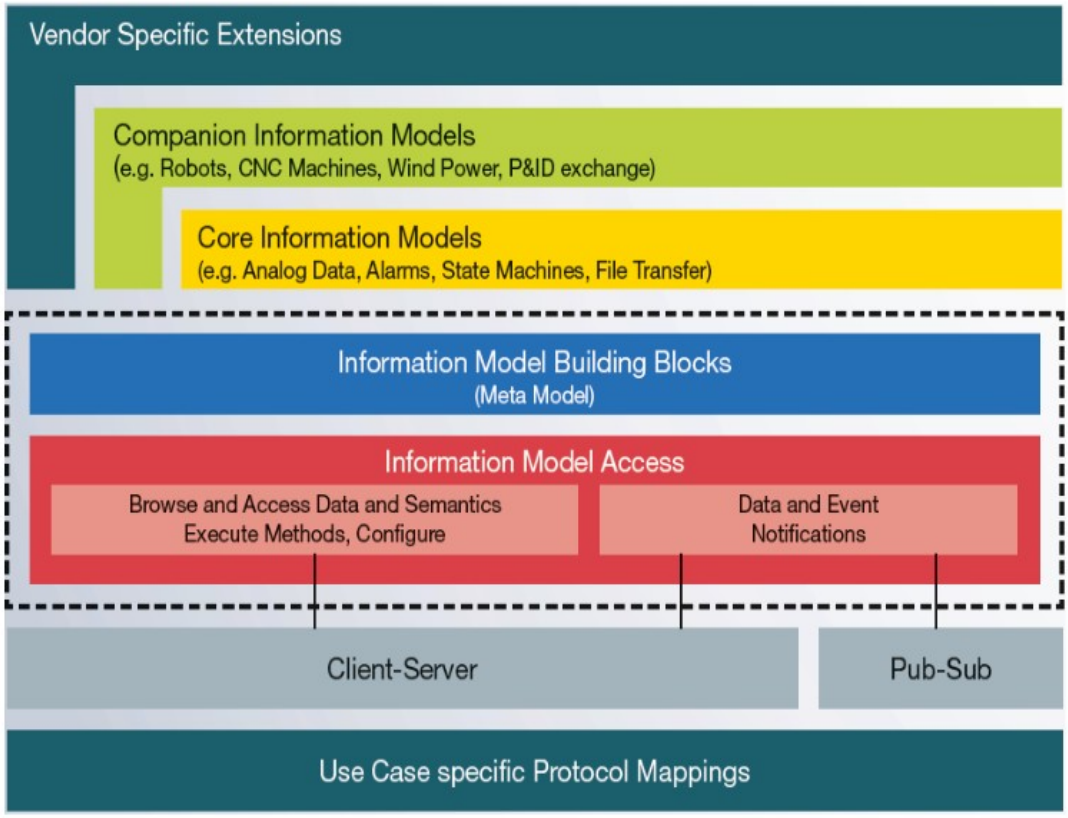
4.2.6 Standardi

Standardeista vastaa IEC (International Electronical Comission) organisaatio, mikä on perustettu kansainvälisen kongressin, joka pidettiin St. Louisissa 1904, ehdotuksesta sopimaan termistöstä ja luokituksesta sähköisille laitteille ja koneille. Perustava kokous oli 26.- 27.6.1906 Lontoossa, puheenjohtajana Alexander Siemens. IEC:n perusajatus on tehdä sähköstä ja sähköisistä teknologioista turvallisempia, tehokkaampia ja luotettavampia. IEC standardi OPC UA:lle on 62541 ja osia on ainakin 100 asti, sillä löysin listauksesta standardin IEC 62541-100:2015. Kuten kohdassa 4.2.2 olleesta kuvasta 12 nähdään, OPC UA on jälleen kehitysvaiheessa ja uudistuksia on tehty paljon vuosien 2021 ja 2022 aikana. (IEC: History, Standardi 62541)

4.3 Informaatiomalli

Informaatiomalli OPC UA:ssa kääntää kehysmallissa tulevan datan käyttökelpoiseen muotoon hankalimmissakin kompleksisissa useamman kerroksen rakenteissa ja se voi mallintaa datan ulottuen siihen täydellisellä olio pohjaisella kapasiteetilla. Kehykset (Frameworks) ovat perustavaa laatua oleva elementti OPC UA:ssa, ja niillä määritellään säännöt ja perusta rakentaa kuvaavat lohkot tarpeelliselle informaatiomallille OPC UA:ssa, jossa itsessään määritellään useita ydinmalleja eri teollisuuden aloille ja muut organisaatiot voivat rakentaa oman mallinsa sen yläpuolelle, jolloin siirtyy erityisesti heihin liittyvät datan erityiset piirteet OPC UA:n, mikä on kuvassa 18 esitettynä. OPC UA määrittelee tarpeellisia pääsy mekanismeja informaatiomallille, kuten Look-up mekanismin tiedon läpikäymiseksi paikantaakseen eri tahot ja niiden semaatiikan, luku- ja kirjoituksen operaatiot nykyiseen – ja historialliseen dataan, metodien suorituksen sekä ilmoitukset datasta ja tapahtumista. (OPC Foundation: OPC UA)

Asiakas/palvelin yhteydenpidossa on valittavana täysi valikoima pääsymalleja palvelimen kautta ja käytössä on palvelin orientoitunut arkkitehtuuri (Serviceoriented architecture, SOA), johon palvelin tarjoaa vastaanoton pyynnöille, prosessoiden ne ja lähettäen tuloksen takaisin vastauksena. PubSub (Publish-Subscribe) tarjoaa vaihtoehtoisen menetelmän datan ja tapahtumien ilmoitukselle. Kun asiakas-palvelin yhteyden pidossa jokaisella ilmoituksella on yksi asiakas taatulle toimitukselle, niin PubSub on optimoitu usealta-usealle konfiguraatiolle. PubSub kanssa OPC UA sovellus ei suoraan vaihda pyyntöä tai vastausta, vaan tosiasiassa tiedon julkaisija lähettää datan viestiorientoituneeseen väliohjelmistoon tietämättä mitä tai jos yhtään allekirjoittajaa siellä voi olla. Allekirjoittajat, PKI :ssa, ilmaisevat kiinnostuksensa tietyn tyyppiseen dataan ja prosessoi viestin ilman tarvetta tietää mistä se on alun perin. (OPC Foundation: OPC UA)



Kuva 18: OPC UA arkkitehtuurimalli informaation käsittelylle (OPC Foundation: OPC UA)

4.4 Tavoitteet

OPC UA tavoite on vastata nykyiseen teollisuuden toimintaympäristöön standardoituna rajapintana. Yhtenä tavoitteena oli oppia OPC Classicin ja OPC UA:n toiminta ja rakenne eri osineen. OPC UA 10000 sisältää jo niin monta osaa etten niitä saanut sisällytetyksi tähän opinnäytetyöhön. Uskon OPC UA:n olevan käyttökelpoinen ratkaisu ja saavuttaneensa tavoitteensa, joka on heidän perusajatus alun perin eli olla kaikkien käytettävissä oleva rajapinta, jonka kautta sovellukset tarjoavat muille sovelluksille mittaus- ja ohjaustietoa tuotannon käyttämästä protokollasta ja laitevalmistajista riippumatta.

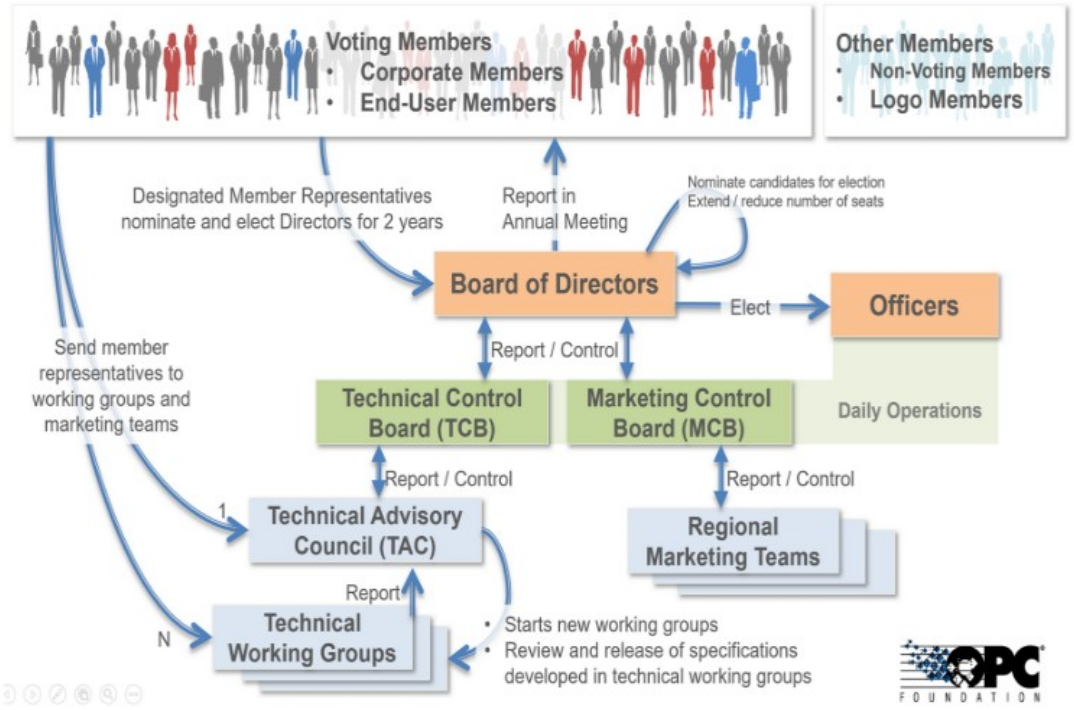
5 OPC Säätiö

OPC säätiö: Teollisuuden yhteentoimivuus standardi, perustettiin virallisesti 22.4.1996. OPC säätiön perusajatus on hallita maailmanlaajuista organisaatiota, missä käyttäjät, tuotetoimittajat ja yhteistyökumppanit, tekevät yhteistyötä kehittääkseen standardeja kaikille tuotetoimittajille ja alustoille, turvallisen ja luetettavan yhteensopivuuden saavuttamiseksi teollisuusautomaatiossa. Saavuttaakseen tämän päämäärän OPC säätiö luo ja ylläpitää määrittelyjä, varmistaa toiminnan ja ohjeiden määräysten sekä säädösten yhteensopivuuden OPC kanssa sertifiointi testauksella. OPC säätiö tekee yhteistyötä johtavien teollisuusstandardijärjestöjen kanssa (OPC Foundation: Mission Statement).

OPC säätiöllä on jäseniä pienistä järjestelmätoimittajista maailman suurimpiin automaatio- ja teollisuustoimittajiin asti. Säätiöllä on tällä hetkellä yli 920 jäsentä ja tuhansia OPC standardeja noudattavia tuotteita. Suomessa kymmenen yritystä on OPC säätiön jäsenenä ja näistä mainittakoon: Neste Engineering Solutions oy, Valmet Automation oy ja Wärtsilä Finland oy sekä oppilaitoksista Tampereen Yliopisto. Jäseneksi pääsee täyttämällä OPC säätiön sivuilla seitsemän vaiheen lomakkeen ja maksamalla jäsenmaksun. Logo jäsenet saavat käyttää OPC logoa, ja opiskelijat pääsevät jäseniksi ilmaiseksi (OPC Foundation). Suomen automaatioseuralla on OPC toimikunta, jonka ”*tavoitteena on edistää suomalaista automaatioalan opetusta, tutkimusta ja yritystoimintaa jakamalla tietoa OPC- järjestön toiminnasta ja spesifikaatioista, järjestämällä koulutusta ja tapahtumia sekä osallistumalla mahdollisuuksien mukaan OPC- spesifikaatioiden kommentointiin ja laadintaan.*” Suomen automaatioseura järjestää OPC Day Finland – ta-

pahtuman kerran vuodessa, edellisen kerran tapahtuma oli 29.11.2022. (Suomen Automaatioseura: Jaostot).

OPC säätiön tehtävää suorittava organisaatio kuuluu mukaan myös OPC:n käyttäjät, jotka muodostavat äänestäjät (Voting members), joita ovat yritysjäsenet ja loppukäyttäjäjäsenet. Muut jäsenet (Others Members) eivät voi äänestää tai lähettää jäseniä työryhmiin (Technical Advisory counsil) ja markkinointitiimiin (Regional Marketing Teams). Äänestäjät valitsevat johtajat kahdeksi vuodeksi johtokuntaan (Board of Directors), joka antaa vuosittaisen raportin äänestäjille ja nimeää ehdokkaita johtokuntaan ja ehdottaa valittavaa paikkamäärää. Johtokunta valitsee virkailijat (Officers), jotka vastaavat päivittäisestä toiminnasta. Tekninen valvonta (TCB) ja markkinointi valvonta (MCB) raportoi johtokunnalle ja valvoo oman osa-alueen toimintaa. Tekninen valvonta raportoi ja saa raportteja tekniseltä neuvonantajien neuvostolta (TAC), jolle tekninen työskentelyryhmä raportoi ja he aloittavat uusia työryhmiä, julkaisevat määrittelyjä ja kehittävät teknistä työryhmää. Äänestävät jäsenet lähettävät edustajia tekniseen neuvonantajaneuvostoon (TAC), teknisiin työryhmiin ja markkinointitiimiin. (OPC Foundation)



Kuva 19: Organisaatiokuva OPC säätiöstä (OPC Foundtation)

Teknisen valvonta (TCB) tehtävänä on keskustella ja neuvoa kaikissa ydintapauksissa ja kumppani määrittelyissä, avoimen lähteen säilytyksessä, kehitys sopimuksissa muun muassa. Se on myös ensimmäinen taho, mikä käsittelee uusiin tekniikoihin liittyviä asioita. Markkinointi valvonta (MCB) tehtävänä on keskustella, johtaa ja neuvoa kaikissa markkinointiin liittyvässä, kuten painopisteessä, toimintasuunnitelmassa, markkinoinnin suunnassa, miltä markkinointi näyttää, kaikenlaisessa vuorovaikutuksessa ja esilletuomisessa, kuten videoissa, Internetissä, lehdistötiedotteissa, sosiaalisessa mediassa muun muassa. Tekniikan ja markkinoinnin neuvoa antava neuvostot on muodostettu johtajista ja visionääreistä suurimmista automaatiotoimittajien ja ydin jäsenistä OPC säätiössä. Teknisen neuvoston jäsenet (TAC) neuvovat säätiön johtoa nykyisissä ja tulevaisuuden teknisissä asioissa ja tarjoavat ohjausta kaikkiin tekniikan aloihin. Erityisesti tehtävät sisältävät uuden tehtävän käyttöönoton ja lopullisen julkaisun hyväksymisen kaikilta teknisiltä työryhmiltä. Markkinoinnin neuvosto (MBC) neuvoo OPC säätiön johtoa nykyisissä ja tulevissa markkinoinnin mahdollisuuksissa ja yleisesti neuvoo säätiön markkinoinnin suunnassa. (OPC Foundation)

6 Teollisuusautomaatio

6.1 Historia

Automaation käsite on määritelty myös näin: ”*Automatisoinnin edellytyksenä on kyky mitata, säätää ja toteuttaa haluttu toiminnallisuus ihmisen ennakkoon osoittamien vaatimusten mukaisesti*” (Asp & Heinonkoski & Hyppönen 2008: 13).

Ihminen alkoi kirjoittaa lukuja ja laskea 5000 vuotta sitten. Matematiikka on automaatiotekniikan peruselementti mittaamisen sekä takaisinkytkennän kanssa. Ensimmäiset mekaaniset laskukoneet ja logaritmit, joita käytettiin navigoinnissa, kehitettiin 1600-luvulla. Logaritminkehittäjiä olivat englantilaiset John Napier (1550- 1617) ja Henrik Briggs (1561-1630) (Asp & Heinonkoski & Hyppönen 2008: 13, Wikipedia-artikkeli: John Napier, Henry Briggs). Teollinen vallankumous aiheutti automaation tarpeen ja ranskalainen keksijä Joseph Marie Jacquard (1752-1834), joka keksi kutomakoneen, jota ohjattiin reikäkorteilla erilaisten kuviokudosten tekemiseksi, mikä oli tietokoneiden ja tietokoneohjelmien kehitykselle merkittävää (Groover 2008: 86, Wikipedia- artikkeli: Joseph Marie jacquard). Logiikka kehittyi 1800-luvulla ja sen tekijöinä olivat August De Morgan (1806 - 1871) Englannista ja George Boole (1815- 1864) Irlannista (Asp &

Heinonkoski & Hyppönen 2008: 13, Wikipedia-artikkelit: August De Morgan, George Boole). Sähkön käytön katsotaan alkaneen1881 New York City:n sähkövoimageneraattoriaseman käytön myötä ja jo 1920-luvulla sähkö ohitti voimanlähteenä höyryn (Groover 2008, 42). Massatuotanto otettiin käyttöön 1920-luvulla Fordin T-mallin valmistuksessa, ja siinä otettiin käyttöön ensimmäisenä liukuhihnaan perustuva tuotantolinja, jossa autot kuljetettiin työpisteestä toiseen mekaanisesti jatkuvana virtana (Groover 2008: 86).

Yhdysvalloissa väestönlaskennassa otettiin avuksi vuonna 1890 reikäkorttikone, jonka kehitti Herman Holleriht (1860- 1929) (Wkipedia-artikkeli: Herman Holleriht), joka perusti yrityksen Tabulating Machines Company, mistä syntyi nykyinen IBM. Reikäkorttikoneessa oli kortinsyöttölaite ja reikäkorttilävistin, joka toimi näppäimistöllä. ENIAC, ensimmäinen tietokone, valmistui 1945 sisältäen 18 000 radioputkea, optimaalinen käyttöaika ilman, että jokin putki vikaantui, oli 20 tuntia, tehontarve 150 kW ja kellotaajuus 100 kHz. Vuonna 1948 keksittiin transistori ja ferriittirengasmuisti sekä levymuisti vuosina 1951 ja 1956 (Wkipedia-artikkeli: Ferriittimuisti). IBM 1620 tietokone laski desimaalijärjestelmällä, ennen tietokoneiden siirtymistä laskennassa binäärijärjestelmään. Cray1- kone oli tieteelliseen laskentaan tarkoitettu nk. supertietokone. 1970 – luvulla, säätö- ja ohjaustehtäviin käyttöön otetut, PDP-8- ja PDP-16 tietokoneet olivat käytössä useissa tehtaissa vielä 1990-luvulla. (Asp & Heinonkoski & Hyppönen 2008, 52 – 55, Groover 2008: 87)

Prosessiteollisuudessa automaation kehitystä vauhditti tietotekniikan kehittyminen ja 1950- 1960-luvulla otettiin käyttöön digitaalinen tietokone. Analogiset laitteet vaihdettiin digitaalisiin säästäen kustannuksissa ihmistyön muuttuessa valvomotyöskentelyksi. Texaco oli ensimmäinen, joka yritti käyttää öljynjalostamolla 1950-luvulla digitaaliohjausta. Imperial Chemical Industries tehtaalla Englannissa 1962, otettiin käyttöön digitaalinen ohjaus, jossa oli mitattavia muuttujia 224 kappaletta ja 129 venttiiliä. Tietokoneella ohjattu työstökone CNC oli MIT (Massachusetts Instutute of Technology) työskennelleiden John Parsons ja Frank Stulen kehittämä. Digitaalinen tietokone mahdollisti uudelleenohjelmoinnin, joka parani itse PLC:n keksimisen myötä 1970- luvun alussa. Samoihin aikoihin kehitettiin CNC (Computer Numerical Control)- ja NC (Numerical Control) -työstökoneita, jotka olivat ohjelmoitavissa. Omron PLC:stä tein seminaariesityksen, jossa esitettiin PLC:n historiaa ja kehitystä sekä siihen liittyviä kenttälaitteita. (Groover 2008: 114-115, 156)

Robotti sana on kuvannut keinotekoisia työntekijöitä, ”Rossum’s Universal Robots (R.U.R)” näytelmässä, jonka kirjoittivat veljekset, Josef ja Karel Éapek. Tsekinkielinen

”robota” tarkoittaa veroluonteisesti tehtyä työtä maanomistajalle. Robotin määritelmä: ” *Robotti on uudelleen ohjelmoitavissa oleva, monipuolinen, vähintään kolminivelinen mekaaninen laite, joka on suunniteltu liikuttamaan kappaleita, osia, työkaluja tai erikoislaitteita ohjelmoitavin liikkein monenlaisten tehtävien suorittamiseksi*”, tämä määritelmä on kansainvälisen robottiyhdistyksen (Asp & Heinikoski & Hyppönen 2008, 110). Vuonna 1938 ohjelmoitava ruiskumaalausmekanismi ja varsinainen teollisuusrobotti 1954 ja sen kaupallinen versio 1962, Unimate-robotti, avusti valurautakoneiden käsittelyssä kuumissa olosuhteissa. Teollisuusrobotin ensimmäinen patentti oli 1961 George Devolin nimissä. Ensimmäinen käsivarsirobotti PUMA (Programmble Universal Machine for Assembly) vuonna 1978 oli yleisrobotti. Vihivaunu keksittiin 1970-luvulla ja 1980-luvulta asti on ollut itsenäisesti toimivia robotteja siirtotehtävissä ja kaivoksissa. Kävelemään robotti opetettiin 1997 Japanissa. Liikkuakseen robotti tarvitsee monipuolisen aistinjärjestelmän, jotta se voi havainnoida ympäristöä ja omaa tilaansa sekä paikkaansa ympäristössä. Konenäön kehittyminen ja paikallistamisjärjestelmien parantuminen on tuonut itseohjautuvat autot ja muut kulkuvälineet, esimerkiksi ruokakuljetuksiin robotit Espoossa. (Asp & Heinonkoski & Hyppönen 2008, 109- 110)

6.2 Nykytilanne

Teollisuusautomaatio jaetaan kappaletavaratuotantoon ja prosessiautomaatioon. Prosessi pyritään hallitsemaan virtauskomponentteja, painetta, lämpötilaa, pinnakorkeutta tai aineen pitoisuutta. Prosessissa on hallittavana myös fysikaalisia, bioteknisiä tai kemiallisia ilmiöitä, jotka voivat luonnostaan olla vakaita (stabiili) tai epävakaita (epästabiili) reaktioita. Vakaassa prosessissa automaatiolla mitataan ja ohjataan prosessin aikana tapahtuvia muutoksia. Epävakaassa prosessissa automaatiolla avulla pidetään prosessin tila haluttuna ja hallitusti siirrytään seuraavaan tilaan käyttäen yleisesti takaisin kytkettyjä säätöpiirejä vertaamalla saatua mittaustietoa vertailusignaaliin. Prosessin toiminnan varmistamiseksi käytetään redundanssisuutta eli kahdentamista tai kolmikertaistamista, jolla estetään vikatilanteiden aiheuttamia pitkiä prosessin toiminnan katkoja. (Kippo & Tikka 2008: 12-13)

Kappaletavaratuotanto on historiallisesti mekanisaation jatke eli massatuotannossa käytetään apuenergiaa tai konevoimaa työvaiheiden suorittamisessa ihmistyön sijasta. Varsinaisesta massatuotannosta on siirrytty massaräätälöintiin, jolloin kuluttaja voi vali-

ta autolleen tai muulle tuotteelle yksilöllisiä ominaisuuksia,jotka asennetaan tuotteeseen jo tuotantolinjalla. Kappaletavara tuotantoa on yleensä automatisoitu pitkälle eri työvaiheissa numeerisesti ohjattujen koneiden ja teollisuusrobottien avulla. Kappaletavara tuotantoa on metalli-, sähkö- ja elektroniikkateollisuudessa, mutta myös vaatetus- ja puusepänteollisuudessa. Usein tuotantolinjan alkupää on jatkuvaa prosessituotantoa ja loppupäässä tuotantoa käsitellään kappaleita. (Asp & Heinonkoski & Hyppönen 2008: 100- 105)

Robotteja käytetään sekä prosessi, että kappaletavara-automaatiossa. Siirryttäessä tuotannossa automaation käyttöön monessa vanhassa tehtaassa aloitetaan automatisointi lisäämällä robotti tuotteiden pakkaukseen tai tekemään tuotantolinjalla toistuvia työvaiheita, jotka kuormittavat työntekijää tai altistavat kemikaaleille ja ammattitaudeille, kuten ruiskumaalaus tai hitsaus. Nykyään käytössä on myös cobotteja, ihmisen kanssa yhteistyössä toimivia robotteja, joita käytetään tuotantolinjalle ja ne ovat turvallisia, eivätkä tarvitse suojahäkkejä ympärilleen. Cobotteja käytetään muun muassa elektroniikkateollisuudessa ja kuluttajatuotteiden valmistuksessa erityisesti pienkomponenttien kokoonpanotehtävissä (Robotiikkatilastot: 33).

Robotteja oli noin 3,7 miljoona kappaletta vuoden 2022 lopussa ja keskimäärin teollisuusmaissa 10 000 työntekijää kohden on 141 teollisuusrobottia, ja 52% uusista roboteista, joita on 517 385 kappaletta, investoidaan Kiinaan. Suomessa investoitiin vuonna 2021 24% enemmän robotteja kuin edellisvuonna ja teollisuuteen tuli 532 uutta robottia, robottitiheyden ollessa 161 teollisuusrobottia 10 000 työntekijää kohden ja niiden kokonaismäärän ollessa vajaa 6000, joista 8% on cobotteja. Uudet tehtaat pyritään automatisoimaan mahdollisimman pitkälle pyrkien pitämään ihmistyön määrän mahdollisimman vähäisenä. Esimerkkinä tästä mainittakoon uusi, Salossa oleva, Salo Tech aurinkopaneelitehdas, jossa on jo täysin automaattinen tuotantolinja 275 W aurinkopaneeleille. (Robotiikkatilastot: 24-26)

6.3 Tulevaisuus

Mobiilirobotiikka on mainittu Automaatioväylän verkkojulkaisussa 2022/12 tulevaisuuden automaatioratkaisuna, minkä mahdollistaa itsenäisesti toimivat mobiilirobottiratkaisut, joiden ”avulla voidaan kasvattaa piensarjatuotantoa tekevän teollisuuden tuottavuutta kustannustehokkaasti ja valmista infrastruktuuria vain kevyesti päivittämällä”. Autonomous Mobile Robot (AMR) toimii vaakatasossa tapahtuvaan tavaroiden siirtoon in-

novatiivisena ratkaisuna. Mobiilirobottien alus-talla voidaan kuljettaa erilaisia kuormakoreja ja ne ovat liitettävissä joustavasti erilaisiin automaatiojärjestelmiin. Esimerkiksi Still ACH- ratkaisussa kuormaustason alle ajava ACH mobiilirobotti nostaa kuormaustason ja siinä olevat tavarat ja kuljettaa tavarat pisteestä A pisteeseen B. Kuorman kantavuus 1500 kg asti, täyttää standardin ISO-3691-4 turvallisuus vaatimukset. Vaihtoehtoisesti siirtää tavaroita uudelleenkäytettävien siirtoasemien tai räätälöityjen yksilöllisten kuormakorien tai – alustojen kuten monikerrollisten vaunujen avulla ja on etähallittavissa ohjausjärjestelmän kautta sekä liitettävissä standardirajapintojen kautta käytössä jo oleviin automaatiojärjestelmiin. (STILL ACH)

Kuva 20: Mobiilirobotti ACH kuljettaa tavarahyllyä (STILL ACH)

Teollisuusautomaatiossa on enenemässä määrin menossa kohti jatkuvaa tuotantoa, jossa keskeytykset on minimoitu ennakoivalla kunnossapidolla, jonka mahdollistaa käytössä oleva IoT teknologia antureineen ja sen tuottaman datan analysoinnin tulosten hyödyntäminen seisokkiaikojen ollessa ennalta sovittuja pitävine aikatauluineen. Mobiilirobotiikan lisääminen tuotantoon ei vaadi raskaita investointeja ja on sovitettavissa huomattavasti pienemmillä muutoksilla tai otettavissa suoraan käyttöön ilman tehdastilamuutoksia vaatimansa pienen käytävätilansa ansiosta. Teollisuusautomaatiossa tulevaisuuden näkyvät onkin ennakoitavuudessa tuotannon suhteen ja muutoksien ja uudistuksien investointien minimoinnissa ja tarvittavien ratkaisujen helppokäyttöisyydessä, turvallisuudessa ja nopeassa käyttöönotossa. (Automaatioväylä 2022/02: 12-16, Robotiikkatilasto: 29)

6.4 Standardit automaation tietoturvalle ja kyberturvallisuus

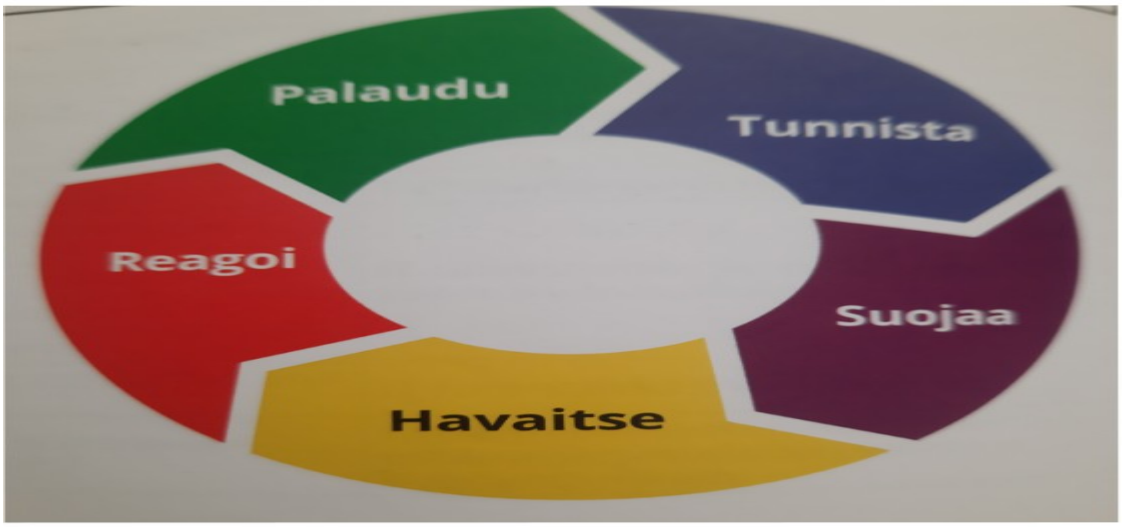
Kriittisen tuotannon turvaaminen vaatii myös automaation tietoturvan pitämistä hyvällä tasolla, niin automaation tieturvan ohjeistuksesta löytyy standardeja, joita esitelty kirjassa automaation tietoturva, jossa Pasi Ahonen on tehnyt kartan parhaista hyödynnettävis-sä olevista tietoturvastandardeista.

- IEC 62443- sarja (Security for Industrial automation and control system) erityisesti automaatiojärjestelmille
- ISO/IEC 27000- sarja [ISO27001]: Tietoturvallisuuden hallintajärjestelmä, joka on hallintastandardi yleiselle tietoturvallisuudelle
- NIST (National Institute of Standards and Technology), mikä sisältää:
    - NIST ”cypersecurity framework”
    - FIPS -standardit (Federal Information Prosessing Standards)
    - SP 800- sarja julkaisut (Special Publications)
    - IR- raportit (Internal Reports)

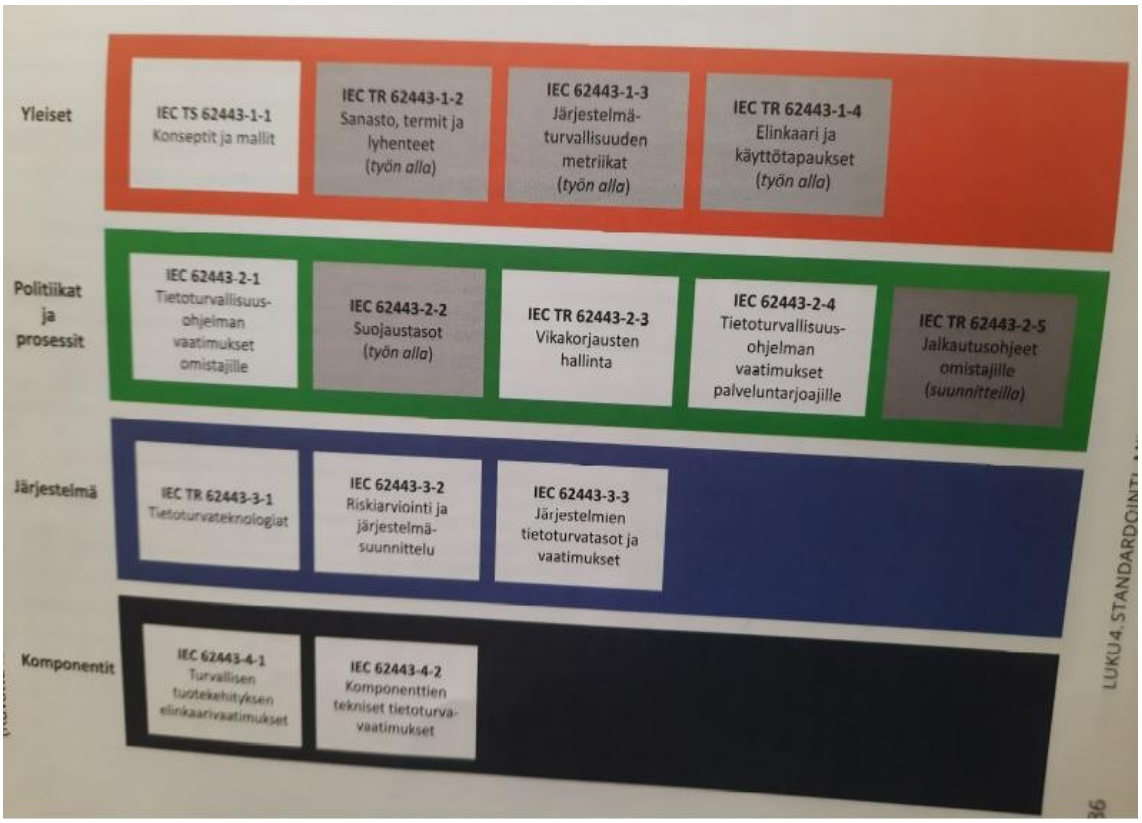
Yhdysvaltain kauppaministeriö pyrkii NIST-standardeilla parantamaan teollista kilpailukykyä ja automaation turvallisuutta. Nykyisessä maailmanpoliittisessa tilanteessa kyberturvallisuudesta on tullut entistäkin tärkeämpi osa automaation käytön suojausta. Kyberturvallisuusriskin hallintakeinot on kuvassa 21 eli tunnista, suojaa, havaitse, reagoi ja palaudu. Tietoturvamekanismit salaus ja tunnistautuminen tehdään kryptografisesti ohjelmistotasolla tai fyysisen laskentaelimen kautta, joka on erityisen luotettava. Näitä käyttää myös Yhdysvaltain liittovaltion hallinnolliset järjestelmät toiminnassaan. Kyberturvallisuuden ylläpidossa ”*konfiguraatiomuutoksia tehdään vain suunnitelluissa huoltoseisokeissa, joissa tavallisesti korjataan tai uusitaan kuluneita tai vanhentuneita*

järjestelmiä", mikä turvaa automaation käytön tuotantovaiheessa ja on ennakoivassa huollossa yksi huomioitava osa-alue. (mm. Ahonen 2021: 84-86, 90, 96)

Kyberturvallisuuden tasosta saadaan selvyys ulkoisella auditoinnilla tai kyberharjoittelulla, jolloin oma henkilöstö ja kumppaniyrityksen henkilöstö osallistuu keinotekoisiin kyberuhka- tai häiriö tilanteisiin, joita oikeasti voi tulla vastaan todellisessa käyttötilanteessa. Tärkeintä harjoittelussa on kyberhyökkäyksestä palautuminen, kuten myös oppiminen kuinka tunnistetaan kriittiset kyberuhat tuotannossa sekä saadaan tilannekuva tietoturvatietoisuudesta ja – osaamisesta. Toimintatavat, ohjeiden toimivuus ja häiriötilanteiden prosessien kehittäminen sekä varautumisen taso että yhteistoiminta kehittyy kyberuhkaharjoittelulla ja muutoksien tarve tulee esiin ja toiminnan parantaminen voi alkaa. "*Tyypillisesti simuloidussa johtamisharjoittelussa harjoitellaan ensisijaisesti yhteistä tilannekuvan ja – ymmärryksen muodostamista sekä sen pohjalta tapahtuvaa päätöksentekoa ja johtamista"* ja tavoitteet ovat kyberharjoituksen suunnittelun lähtökohta. Suunnittelun jälkeen, kun skenaario on kirjoitettu, mikä usein kohdistuu kriittisiin järjestelmiin ja niiden tietoihin ja toimintaan, tulee toteutusvaihe, minkä jälkeen on palaute – ja jatkotoimenpidevaihe. Palautteesta selviää, toteutuiko kyberharjoituksen tavoitteet ja mitä pitää muuttaa tai parantaa. (mm. Ahonen 2021: 197-209)



Kuva 21: Reagointimalli kyberuhkaan (mm. Ahonen 2021: 90)



Kuva 22: ISA- IEC standardit, tilanne 14.10.2020. (mm. Ahonen 2021: 86)

7 Teollinen Internet

7.1 Kehitys

Teollinen Internet on termi, jonka GE otti käyttöön vuonna 2013. Teollisen Internetin muodostavat esineiden Internet (IoT) kautta saatavan tiedon analysointi ja työn tehostaminen yhdistäminen liiketoiminnassa, joka mahdollistaa uudet liiketoimintamallit ja kilpailukykyiset asiakkaalle räätälöidyt palvelut (Internetopas). Bruce Sinclair kirjoittaa teoksessaan IoT Inc:how your company can use the internet of things to win in the outcome economy. Kuinka he myivät IoT teknologialla varustetun hiirenloukun ACME Pets yritykselle IoT:n alkuaikoina vuonna 2008, jolloin IoT:lla varustettujen tuotteiden hinta oli 30% kalliimpi kuin tavallisen tuotteen, mikä hintaero täytyi ostajalle perustella ja vakuuttaa myöhemmin koituvan yrityksen eduksi ja asiakkaan hyödyksi sekä toimivaksi kilpailueduksi. (Sinclar: XVII- XIX, 16, 41-42).

Itse IoT on fyysisten laitteiden muodostama verkko teollisuuslaitoksessa ja yhdistämällä IoT:hen liitetyt koneet ja laitteet Internetin kautta saavutetaan etuja, kuten reaaliaikainen suorituskyvyn seuranta, ennakoiva kunnossapito, ketterä vaste tuotteiden kysyntään ja parempi turvallisuus. Ennakoivalla kunnossapidolla voidaan vähentää suunnit-

telemattomia seisokkeja, vahinkoja ja keskeytyksiä tuotannossa. Reaaliaikaiset päivitykset etänä Internetin kautta lisäävät tehokkuutta korvaten manuaaliset päivitykset. (Epicor, Teollinen Internet)

Suomessa valtio on tukenut eri toimenpideohjelmilla Tekesin ja Business Finlandin kautta IoT käyttöönottoa teollisuudessa, vähittäiskaupassa ja logiikassa, rakentamisessa ja kiinteistöhuollossa sekä terveydenhuollossa. Tekes teki ohjelman vuosille 2014-2019 ”Teollinen Internet liiketoiminnan vallankumous”, jonka tavoitteena oli kuroa kiinni kansainvälisiä kilpailijoita IoT alla ja saada aikaiseksi uutta kasvavaa liiketoimintaa IoT:n parista. 5G ratkaisut ja verkkoa rakennettiin vuonna 2014 ja Tekes teki toimenpidevalmisteluja ja piti seminaareja aiheesta. Tuolloin 5G päätelaitteen akku kesti 3 minuuttia. 5G kapasiteetti mahdollisti 10000 kertaisen tiedonsiirron määrän 4G verkkoon verrattaessa. Kasvua IoT:stä odotettiin kaikille toimialoille, erityisesti palvelu, analysointi ja visualisointi, datan keruuseen ja hallintaan, sensoreihin. Teollisuudessa tärkeinä alueina mainittiin prosessien käyttö ja jakeluketjut. Kaikissa näissä IoT on jo ollut käytössä enenevässä määrin. (Teollinen Internet ja 5G)

7.2 Käyttö teollisuudessa

Käyttö teollisuudessa perustuu eri IoT ratkaisuihin, joita tarjoavat DNA, Elisa ja Telia mobiiliyhteyksillä, joiden avulla on mahdollista toteuttaa IoT-verkko, joiden avulla IoT antureiden tiedot saadaan ajantasaisesti yrityksen käyttöön. Elisalla on kokonaisratkaisu Elisa Business IoT, jonka luvataan mahdollistavan tiedolla johtamisen. Se on tiedonkeruupalvelu, jolla voi turvallisesti ja tehokkaasti kerätä tietoja IoT laitteilta, kuten vesimittareilta, joihin on tarjolla oma Elisa Business IoT Vesi, mikä tuottaa tietoa veden kulutuksesta, antaa hälytyksiä raja-arvojen ylittyessä, joten teollisuuden käyttöön kehitetty IoT on tullut myös kotikäyttöön. Itse yrityskäyttöön Elisa Business IoT tarjoaa valmis ratkaisua Elisa Business IoT paikannusjärjestelmää, jolla voi paikantaa kulkuneuvot, niiden liikkeet ja poistumisen rajatulta alueelta, ja sillä on keskitetty tietovarasto. DNA:n yritysratkaisu on IoT Connect- M2M-liittymät, joissa tiedonsiirto antureiden, laitteiden ja järjestelmien välillä tapahtuu mobiiliverkossa ja sen voi yhdistää IoT Cloud-palveluun, jonka tarjoaa Telenor. Hallintaportaalin, DNA Control Centerin, myötä käytössäsi on reaaliaikainen näkymä, minkä avulla pystyt hallinnoimaan tapahtumia IoT-verkossa. Telian skaalautuvassa ratkaisussa, NB-IoT, heikonkin kuulevuuden alueella

laitteita voi yhdistää toisiinsa, mitata- ja valvoa esimerkiksi lämpötilaa ja kosteutta, seurata rahtia ja IoT-alustalla analysoida saatuja tietoja laitteilta. (DNA: IoT, Eli-sa: IoT, Telia: IoT)

Säätö/ohjaussovellus	Sovelluskerros (Application Layer, APL)
Tiedon generointi ja purku	Kuljetuskerros (Transport Layer, API)
Tiedon reititys	Verkkokerros (Network Layer, NWK)
Tiedonsiirron varmistukset	Siirtoyhteyskerros (Data link layer, MAC)
Radiosignaalin modulointi, lähetys ja vastaanotto	Fyysinen kerros (Physical Layer, PHY)

Kuva 23: OSI- protokollapino langattomalle tiedonsiirrolle (Palomäki 2007: 8)

Mobiiliyhteyksillä toimivat IoT-verkot ja laitteet vaikuttavat mainonnan perusteella helposti käytteenotetaviksi ja edulliseksi ratkaisuksi IoT-verkolle ja myös turvallisuutta korostetaan. Isoissa teollisuusyrityksissä on verkkoratkaisuja, jotka toimivat omilla tehtaan verkkoratkaisuilla ja käyttävät eri tekniikoita yhteyksien toteuttamiseen ja yleisimmin toteuttavat OSI- protokollapinoa langattomalle tiedonsiirrolle, kuva 23, missä sovelluskerroksella tapahtuu säätö/ohjaus, tiedon generointi ja purku kuljetuskerroksella, verkkokerroksella reititetään tiedot, siirtoyhteyskerroksella varmistetaan tiedot ja radiosignaalin modulointi, lähetys ja vastaanotto tehdään fyysisellä kerroksella. Käytössä voi olla Bluetooth, Zigbee, RFID ja TUTWSN(Palomäki 2007: 2, 8). LoRa tekniikkaa on esimerkki vaihtoehdosta, jos mobiiliverkko ei ole käytettävissä, siinä käytetään LoRaWan-antureita.(Elkome: LoRaWan). Lähetin ja verkkoratkaisuja on olemassa lukuisa määrä eri käyttötarkoituksiin, myös eri antureilla on omia itsenäisiä lähettimiä, joilla liitytään yleensä mobiiliverkkoon ja itse sovellus anturintoimittajan palvelimella ja yhteys ja tiedot anturilta saadaan mobiiliverkon kautta puhelimeen tai tietokoneelle.

7.3 Tulevaisuuden käyttökohteet

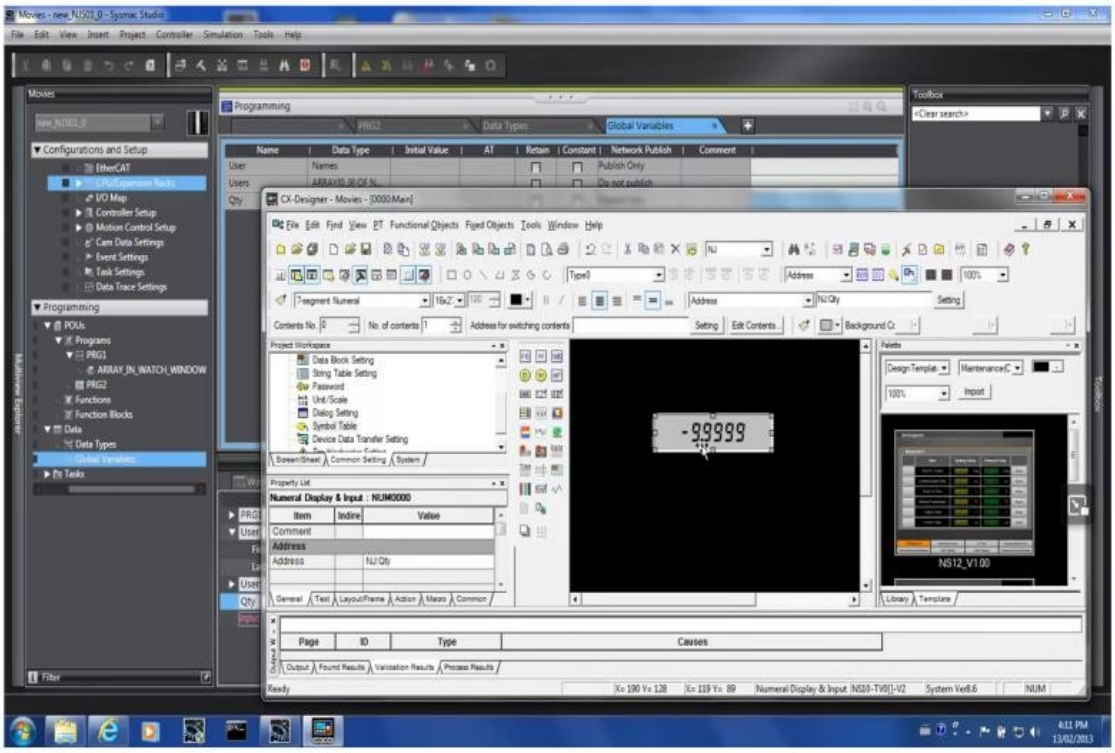
IoT sensoreiden tulevaisuutta on älypöly, joka on mahdollista levittää autonomisesti toimivina MENS antureiden ja robottien muodostamana ohjelmoitavana verkkona, hiekanjyvän ja pölyhiukkasten kokoisina laitteina, joilla mitataan ja tarkkaillaan kosteutta, lämpötilaa, ilmanpainetta, ilmansaasteista, seismisestä värähtelyä, maaperän myrkyllisyyttä tai magneettisuutta, esimerkiksi pelloille, sotilaskohteissa tai tiloihin joihin on vaikea asentaa langallista sensoria tai sensorille varattava tila on liian pieni normikokoiselle sensorille. Käyttökohteita älypölylle ovat myös teolliset tuotantolaitokset ja lääketieteen diagnostiikka, missä nanoteknologian avulla ihmiskehoon ujutetaan älypölyä tuhoamaan viruksia ja syöpäsoluja. (Wikipedia-artikkeli: Älypöly, Helsingin Sanomat 14.12.2022)

Kristofer Pister esitteli älypölyn vuonna 2001 ja sen jatkokehitystä on rahoittanut vuodesta 1998 Yhdysvaltain puolustushallinnon Darpa-tutkimuslaitos. Langattoman verkon avulla älypölyllä mitattu tieto siirtyy keskustietokoneeseen, jossa se tiivistetään ja jalostetaan ajantasaisiksi raporteiksi ja tunnusluvuiksi, älypölyn toimiessa riippumattomasti virtalähteidensä varassa. Älypöly ohjelmoidaan ennen käyttöä ja etukäteisohjelmointia ei enää kesken toiminnan pysty muuttamaan. (Wikipedia-artikkeli: Älypöly, Helsingin Sanomat 14.12.2022) Älypölyn ongelmia ovat sen käytön jälkeensä jättämä jätemäärä, sen käyttöön liittyvä lainsäädännön puute ja tuotannon kalleus. Jäteongelmaan on haettu ratkaisua ja yksi tällainen ratkaisumahdollisuus voisi olla nanosellun käyttö, mitä on jo käytetty radiosignaalien ohjauksessa käytettävissä linsseissä. Puupohjaista elektroniikkaa tutkitaan myös Oulun Yliopistossa mikroelektroniikan tutkimusyksikössä. (Wikipedia-artikkeli: Älypöly, Helsingin Sanomat 14.12.2022, Oulun Yliopiston verkkojulkaisu: Puusta elektroniikkaa)

8 Sysmac Studio

8.1 Virtuaaliympäristö

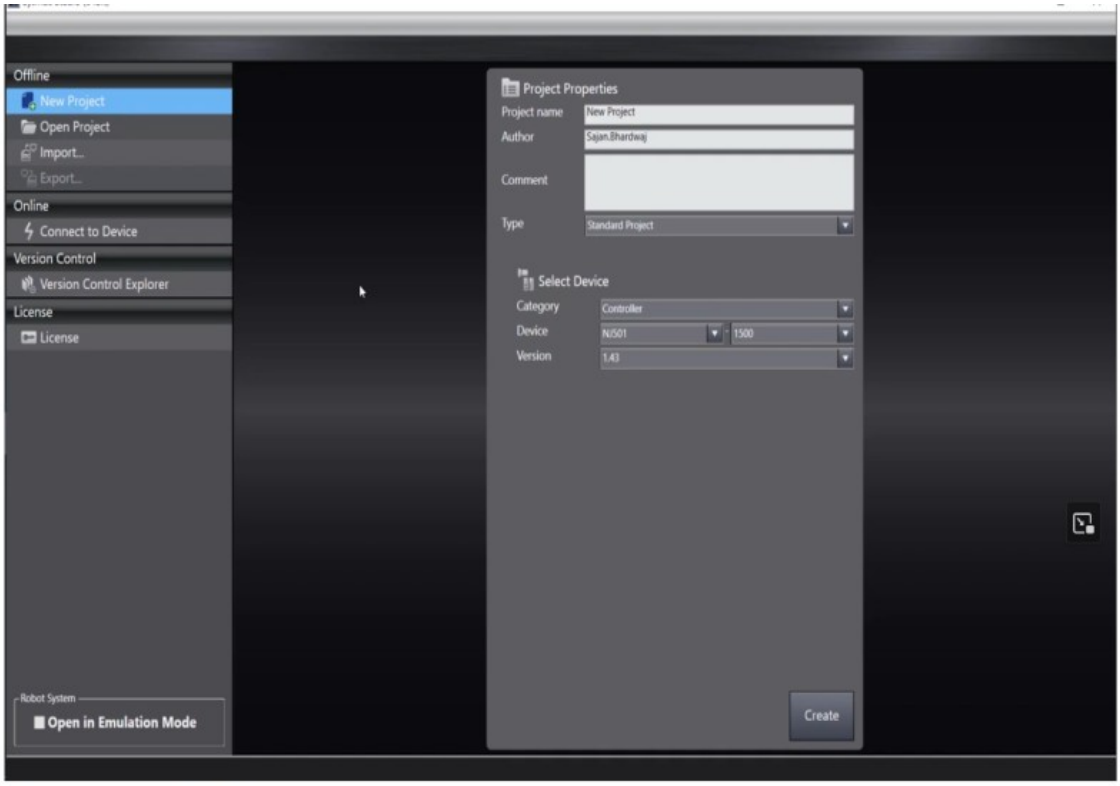
Simuloinnissa kuten PLC yhteydessäkin luodaan uusi projekti, valitaan yleensä standardiprojekti ja valitaan PLC laite. Simuloinnissa tarvitaan CX-designer, jonka avulla simulointi onnistuu. CX-designer on skaalattavissa oleva HMI ohjelmisto, joka on CX-one osa ja sallii tagien jakamisen PLC:n ja HMI välillä ja siihen voi siirtää drag & drop- toiminnolla tai kopioimalla Excelistä muuttujat. 1500 esiohjelmoidun osan kirjasto, helppo viedä ja tuoda projektia ja kääntää ne. Ensin luodaan muuttujat, sitten simuloidaan C# ohjelmakielellä tehty ohjelma CX- designerin avulla, jos olisi onnistunut muuttujien arvon muuttaminen, niin se olisi toiminut simulaatiolla, muuttamalla lomakkeen tilaa ja arvokenttään arvoja. Tämä osoittautui itselle liian vaikeaksi tehtäväksi tässä ajassa, kun toukokuun puolessa välissä sain uudelleen käyttööni Sysmac Studion, joka on vain 30 päivän koekäyttöversio. Aiemmin käytössäni ollut Sysmac Studio ohjelma oli lisenssillä, mutta se meni tuhoutuneen koneen myötä ja valitettavasti alkuvuoden jälkeen olin ilman työkonetta ja ainoastaan 5G modeemin varassa syksystä lähtien. (OMRON)



Kuva 24: Simulointi CX- Designerin avulla (OMRON)

8.2 PLC-yhteys

PLC yhteyden avulla ohjelman testaaminen oli helppoa, kytkettiin PLC Ethernet kaapelilla tietokoneeseen, jossa avoinna ohjelmat Sysmac Studio, Visual Studio ja siitä avoinna ohjelman projekti mitä aiotaan ajaa sekä samassa hakemistossa oleva sertifikaatti käyttövalmiina. Laitettiin virrat päälle PLC laitteeseen, avattiin Sysmac Studio, luodaan projekti tai avataan projekti, jos luotuna, Configuration Setup, sen jälkeen OPC UA Settings, hiiren vasemmalla Client Authentication, + merkki, pingataan komentoikkunaa 198.162.250.1. Muuttujat on laitettu ensimmäisellä kerralla Global variables, valitut muuttujat koirat ja kissat input ja output muuttujiksi eli Global Variablesin teko: offline, alimmainen data tools, Global Variables kaksoiklikkaus, hiiren vasemmalla add. Muuttujatyypit BOOL, INT, String[x] , x valitaan 256, LREAL. PLC:osoite valmiina, jos oikea laite ja sen maski. PLC yhteyden käyttöä olisin halunnut jatkaa, mutta PLC meni komponenttipulan takia asiakastestauksiin, mikä piti olla vain väliaikaista, mutta koronatilanne vain jatkui ja komponenttipula paheni entisestään ja vieläkin on toimituksissa viivettä komponenttien osalta. (OMRON)



Creating a Project Using OMRON Sysmac Studio

Kuva 25: Standardiprojektin luonti (OMRON)

9 Väyläprotokolla ohjelmointi

9.1Yleisesti

Tietokonetta oli alettu käyttää aiemmin automaation osana, mikä oli johtanut siihen, että väylä tietokoneelta automaatioon täytyi saada hallintaan ja siten tuli tarpeelliseksi väylän ohjaus ja ohjelmointi väyläprotokolla ohjelmoinnilla. Samalla automaation kehitys on ollut nopeaa ja se käyttö laajentunut teollisuudessa, terveydenhoidossa ja muilla aloilla, mikä on johtanut siihen, että väylän hallinnasta tietokoneelta automaatioon tuli kriittisempää ja automaation valmistajia oli useita, jolloin syntyi tarve standardoinnille. Väyläprotokolla ohjelmoinnilla pyritään ohjaamaan tietokoneelta yhteyttä automaatiolaitteeseen ja viemään tietokoneelta ohjelma PLC:lle sekä seuraamaan automaation tilaa ja tekemään tilanteesta raportointia, jonka avulla seurataan ja ohjataan automaatiota toimimaan optimaalisesti reaaliaikaisen tiedon perusteella eri lähteistä. Käytetään hyväksi valittua ohjelmointikieltä yhdistävänä tekijänä tietokantaohjelmointiin ja sen suomia mahdollisuuksia optimoida automaation käyttöä. (Iwanitz ja Lange 2006: Alkusanat)

9.2 C# (käytetty ohjelmointikieli)

Olioperusteinen C# on vahvasti tyypitetty kieli eli sen muuttujilla on aina tietotyyppi esimerkiksi luokka, merkkijono tai kokonaisluku, mutta käytettävissä on kuitenkin var- sana, kun muuttujalla on kääntäjälle ja lukijalle selvä tyyppi. Muistinhallintaa kuuluu roskienkeräys ja se on automatisoitu siten, että muistia vapautuu objektin ollessa ohjelman saavuttamattomissa. C# on Microsoftin kehitystyön tulos ja julkaistu 2000. C# on saanut vaikutteita C++ -, Small talk – ja java- ohjelmointikielistä (Wikipedia-artikkeli: C sharp). Itse ohjelmointi tehdään tekstieditorilla ja itsellä oli käytössä Visual Studio ohjelmointialusta. Ohjelmointikieli C# tuli valituksi väyläprotokolla ohjelmointi opinnäytetyöhön DA-Teamin ehdotuksesta. Opiskelin itse C# ohjelmointikieltä eri oppaiden ja netin esimerkkien avulla ja käymällä C# ohjelmoinnista verkkokurssin Metropoliassa. Tein seminaariesityksen C# ohjelmointikielestä ja siinä esiteltiin sen historia, rakenne ja käyttökohteet.

9.3 Visual Studio

Visual Studio on 1997 käyttöön otettu ohjelmistoalusta eri ohjelmointikielille, ja siitä on tullut 1-3 vuoden välein uusi versio. Vuonna 2002 Visual Studioon tuli .NET ja otettiin

käyttöön CLR, Common Language Runtime, eli tulkkausta varten yhteinen kieli. Vuosien 2019 ja 2022 versiot ovat Visual Studio Community nimikkeellä. Tarkoitettu ohjelmoijille ja siitä on ilmaisversioita osasta eri vuosien julkaisuista (Wkipedia-artikkeli: Visual Studio). Nykyisin on tarjolla ladattavaksi Visual Studio 2022. Visual Studiossa on koodin värjäys, jolloin eri väreillä korostetaan säännöllisiä lausekkeita (Wkipedia-artikkeli: Koodin väritys). Versionhallinnalla pyritään Visual Studiossa tallentamiseen käytettävän levytilan minimointiin (Wikipedia-artikkeli: Ohjelmiston versionhallinta). Visual Studio ohjelmasta tein osana seminaariesitystä, C# ohjelmointi Visual Studiolla, debuggauksesta ja muista ominaisuuksista Power Point-esityksen, jossa esitin myös silloin löydettävissä olevista linkeistä koottua tietoa ohjelma- alustan historiasta, rakenteesta ja käytöstä.

10 Ohjelmaesittely

10.1 Ohjelman rakennekuvaus

Ohjelman alussa esitellään kirjastot ja nimiavaruus kuten System.ComponentModel nimiavaruus, joka tarjoaa luokat, joita käytetään ajonaikaisille ja kuvan piirron aikaiselle toiminnalle ja se tarjoaa perusluokat ja rajapinnan ilmentämään ominaisuuksia ja tyyppien muutoksia sitoen datalähteet ja lisensioinnin komponenteille. Nimiavaruus TestConnectOPCUA sisältää ohjelman nimen ja nimiavaruus itsessään sitoo yhteen nippuun kielen symboleja, mitkä on käytössä tietojen käsittelyssä. Public partical class Form1 : Form rivillä partical class on erityispiirre C#, mikä tarjoaa mahdollisuuden ilmaista toiminnallisuutta yhdessä luokassa, jossa on monta tiedostoa, kun sovellus on käännetty. (Moghadampour: 179, 200). InitializeComponent(); on metodi Visual Studiossa .NET C# tai VB.NET:ssä, mikä automaattisesti on luotu ja hallinnoitu Windows lomakkeelle ja sen muodostamiselle määritellen kaiken, mitä voit lomakkeella nähdä. Luokassa määritellään yhteyden aikainen m_session ja aletaan yhteyden muodostaminen. Session on tila hallinta tekniikka ja se voi varastoida arvon palvelimelle tukien kaikkia olio tyyppejä oma asiakas olion kanssa. Session on yksi parhaista tekniikoista tilan hallinalle, koska se varastoi datan asiakaskohtaisesti (Client). (Moghadampour: 179, 200)
Luodaan muuttuja m_connection, jonka tietotyyppi on boolean. Private Subscription m_subscription, mikä tarkoittaa toisessa luokassa olevan tapahtuman tilausta. Private void connectServerCtrl1_ConnectionComplete(object sender, EventArgse) yhteys val-

mis, olio lähetetään. Istunto alkaa Serveriin Ctrl1 kun m_sessio on erisuuri kuin null ja m_connectedOnce on totta. Poikkeuksesta tulee määritelty virheilmoitus. Yhteys luodaan m_session.AddSubscription(m_subcription); m_subcription.Create();. Kutsutaan metodeja yksi, kaksi ja kolme. Ensimmäisessä metodissa määritellään taulukko, jonka tyyppi String ja nimi tags. Lomakkeelle tulevat Label1 ja Label2. Lomakkeella olevan Boxi1 sisältö määritellään metodissa kaksi. Metodissa kolme mahdollistaa muutokset. Lomakkeella oleva liitinkuva pictureBox1 vaihtaa väriä sinisen ja mustan välillä, kun yhteydenmuodostus nappulaa on painettu. (Microsoft 2020)

Yhteyksien uudelleenherätykset tehdään jokaiselle MonitoredItem_Notification kohdalle ohjelmassa tarvittaessa; if (InvokeRequired). Jos saadaan yhteys aikaan, näkyy form1. Lomakkeella IP-osoite tulee käskyn this.connectServerCtrl1.ServerUrl = ”opc.tpc://192.168.250.1:4840”; mukaisesti. Yhteyden muodostus vaatii sertifikaatin, joka tulee olla koneella samassa hakemistossa kuin itse ohjelma ja sinne on polku ohjelmassa ja tätä varten on ApplicatioCertificate = new CertificateIdentifier osasta alkava luotettavan sertifikaatin haku ja sen lisäys luotettujen sertifikaattien listaan ja sertifikaatin ominaisuuksien lisäys, kuten OperationTimeout = 15000. Niiden jälkeen on virheilmoitukset. Ja yhteydelle on pääsy turvallisesti, kun sertifikaatti luotu application.CheckApplicationIntanceCertificate(true, 2048).GetAwaiter().GetResult() ;. Lopuksi yhteyden lopetus private void From_FromClosed(object sender, FormClosedEventArgs e) osiossa. Sulku lomakkeella label3. (Microsoft 2020)

11 Yhteenveto

Väyläprotokolla ohjelmointi ja ohjelmointikieli C# tulivat valituksi aiheeksi opinnäytetyöhön DA-Teamin ehdotuksesta ollessani heillä työkokeilussa. Väyläprotokolla ohjelmointia on Internetin hakujen perusteella yrittänyt iso joukko C#- ohjelman käyttäjiä ja opinnäytetyökin on tehty aiemmin samalla menestyksellä kuin omani eli ei muuttujan arvon muutosta saatu aikaan. Työssä käsittelin OPC Classic historiaa ja sen osia. OPC UA, joka vastaa nykyisiin tarpeisiin automaatiossa mentäessä kohti digitaalista tuotantoa, on käsitelty varsin kiireellisellä aikataululla ja siihen olisi voinut paneutua paremmin oppimismielessäkin. Teollisuusautomaatio ja IoT- tekniikkaan tutustuminen ja siitä materiaalin tutkiminen antoi uutta tietoa itselle alan kehityksestä tässä teollisuus 4.0 vaiheessa, jossa täysin automatisoidut tehtaat vastaanottavat tilauksen asiakkaan mitoittamana ja tekevät tilatun tuotteen sarjatuotannon omaisesti.

Väyläprotokolla ohjelmointi PLC:n avulla onnistui yhteydenottoon ja arvojen muutoksen virheellisyyden toteamiseen asti, josta dokumentaatiota ei jälkipolville jäänyt ensinnäkin

sen takia, että en saanutkaan alkaneen komponenttipulan takia pitää PLC-laitetta käytössäni ja väärä olettamus, että komponenttipula on pian ohimenevää, esti PLC-laitteen käytön dokumentoinnin jälkikäteen uudelleen toistettuna. Lisäksi ne vähän verkkokaappaukset, jotka oli, menivät koneen kaaduttua viruksen takia eli jokseenkin tuli tunne, että tälle opinnäytetyölle ei ole ollut oikea aika eikä paikka tehdä sitä nyt. Ohjelma ottaa yhteyden PLC:hen ja siinä on käytössä Internetin eri sivuilla olevista kysely ja vastauspalstoista ja OPC säätiön sivuilta saatujen vihjeiden avulla tehty runko, johon kuitenkaan en pystynyt tekemään mitään lisäarvoa tuottavaa alkutilanteeseen nähden. Itse C# ohjelmoinnin opiskelu oli mielekästä ja toivon, että opin jotain, mistä on hyötyä tulevaisuudessa mahdollisella työuralla.

Olosuhteet tehdä opinnäytetyötä olivat jokseenkin haasteelliset, ainakin itselle, ulkoisista tekijöistä johtuen, sillä työtiloissa ja yhteyksissä oli haasteita. Mutta olosuhteet on tehty voitettavaksi ja oma terveys on asettanut myös haasteita opinnäytetyön teolle. Lisäksi ajan käytön hallinta ja ajan varaaminen opinnäytetyölle on ollut haasteellista. Ohjelman tekemiseen on vaikut-tanut korona, ei ole voinut käydä DA-Teamsissä, komponenttipula, ei ole voinut käyttää PLC yhteyttä ja konerikko, ei ole System Studio lisenssiä tällä hetkellä, ja paniikinomainen yritys selviytyä simulaation- toteutuksesta loppui ajan puutteeseen. Työn on siis kesken ja toivotan onnea seuraavalle yrittäjälle, jos joku ottaa aiheekseen väyläprotokolla ohjelmoinnin ja itse ohjelman tekemisen.

Lähteet

Asp, Risto & Heinonkoski, Risto & Hyppönen. 2008. Automaatio – helppoa elämää? Opetushallitus: Vammalan kirjapaino Oy

Automaatioväylä 2022/2, viitattu 10.2.2023: https://www.automaatiovayla.fi/site/assets/files/4747/automaatiovayla_2_2022.pdf

Creating a Project Using OMRON Sysmac Studio Youtube video. Viitattu 30.5.2023: https://www.youtube.com/watch?v=WeBS-6i1b0c

C sharp, Wikipedia-artikkeli, viitattu 29.5.2023: https://fi.wikipedia.org/wiki/C_sharp

CX-designer, Omron verkkosivut. Viitattu 30.5.2023: https://industrial.omron.fi/fi/products/cx-designer

Damm, Matthias, Latest News on OPC UA PubSub and Security Configuration esitys OPC päivillä 29.11.2022, viitattu 23.5.2023: https://www.automaatioseura.fi/site/assets/files 3391/04_2022_11_29_news_pubsub_securityconfig.pdf

DNA Iot, viitattu 30.5.2023: https://www.dna.fi/yrityksille/iot

Elisa IoT, viitattu 30.5.2023 :https://yrityksille.elisa.fi/iot

Groover, Mikel P. 2008. Automation, Production Systems, and Computer-Integrated Manufacturing, New Jersey: Person Eeducation Inc.

History. OPC Fountation, viitattu 2.5.2023: https://opcfoundation.org/about/opc-foundation/history/

History of OPC. OPCConnect.com verkkoartikkeli, viitattu 2.5.2023: https://www.opcconnect.com/history.php

IEC History, verkkojulkaisu, viitattu 29.5.2023: https://www.iec.ch/history
Informaatiomallin standardi IEC 62541-5, viitattu 29.5.2023: https://webstore.iec.ch/publication/61114

Iwanitz, Frank & Lange Jürgen. 2005. OPC Fundamentals, Implementation and Application. 2006 Germany: Hüthig GmbH & Co. KG Heidelberg.

Kippo, Asko K & Tikka, Aimo. 2008. Automaatiotekniikan perusteet. Helsinki: Edita Prima Oy.

Koodin väritys, Wikipedia-artikkeli, viitattu 29.5.2023: https://fi.wikipedia.org/wiki/Koodin_v%C3%A4ritys

Kuin tieteiselokuvasta, mutta kohta ehkä totta – kaikkialle levitettävä älypöly voisi mitata mitä vain, Helsingin Sanomat Tiede artikkeli, 14.2.2022, digilehti, luettu 14.12.2022: https://www.hs.fi/tiede/art-2000009184526.html

LoRaWan anturit, viitattu 30.5.2023: https://elkome.com/jarjestelmatuotteet/lorawan-anturit-lahettimet-ja-gatewayt/

Lutz, Peter, OPC UA Field Level Communication (FLC) and OPC UA Field Exchange (FX), esitys OPC päivillä 29.11.2022, viitattu 23.5.2023: https://www.automaatioseura.fi/site/assets/files/3391/02-2022-11-29_opc_ua_field_level_communication_and_opc_ua_field_exchange_-_finland.pdf

Microsoft, Mircrosoft learn, käskyjen selitykset, viitattu 29.5.2023: https://learn.microsoft.com/fi-fi/

Microsoft Visual Studio, Wikipedia-artikkeli, viitattu 29.5.2023: https://fi.wikipedia.org/wiki/Microsoft_Visual_Studio

Mission Statement, OPC Foundation verkkosivut, viitattu 15.4.2023: https://opcfoundation.org/about/opc-foundation/mission-statement/

Mikä on Teollisuus 4.0 – Teollinen esineiden Internet (IIoT, Industrial Internet of Things)?, Epicor verkkojulkaisu, viitattu 20.10.2022: https://www.epicor.com/fi-fi/resources/articles/what-is-industry-4-0/

Moghadampour, Ghodrat, C# Ohjelmointi. 2012. Vantaa, Hansaprint.

.NET historia, Wikipedia-artikkeli, viitattu 7.5.2023: https://fi.wikipedia.org/wiki/.NET_Framework

Ohjelmiston versionhallinta, Wikipedia-artikkeli, viitattu 29.5.2023: https://fi.wikipedia.org/wiki/Ohjelmiston_versiohallinta

OPC Foundation, viitattu 15.4.2023:

https://opcfoundation.org/about/opc-foundation/organization/

OPC Foundation and PLCopen release version 1.02 of the “OPC UA for IEC61131-3” specification, OPC Foundation verkkosivut, viitattu 29.5.2023: https://opcfoundation.org/news/press-releases/opc-foundation-and-plcopen-release-version-1-02-of-the-opc-ua-for-iec61131-3-specification/

The OPC Foundation releases the OPC UA Field eXchange (UAFX) Specification, viitattu 28.5.2023: https://opcfoundation.org/news/press-releases/the-opc-foundation-releases-the-opc-ua-field-exchange-uafx-specifications/

OPC-toimikunta, Suomen Automaatioseura, viitattu 2.5.2023: https://www.automaatioseura.fi/sas/jaostot/

OPC UA Meta Model, OPC Foundation verkkosivut, viitattu 28.5.2023: https://reference.opcfoundation.org/Core/Part3/v105/docs/B.1

OPC Mission Statement ja OPC säätiön verkkosivut, luettu 12.4.2023, https://opcfoundation.org/about/opc-foundation/mission-statement/

OPC Security, OPC Foundation verkkosivut, viitattu 29.5.2023: https://opcfoundation.org/wp-content/uploads/2022/10/FAQ-ON-INDUSTRIAL-ETHERNET-SECURITY-CONCEPTS.pdf

Palomäki, Heikki, Langaton tiedonsiirto teollisuudessa esitys 2007, viitattu 30.5.2023: https://slideplayer.fi/slide/2000539/

Puusta elektroniikkaa?, Oulun Yliopisto verkkoartikkeli 20.4.2021, luettu 19.12.2022: https://www.sttinfo.fi/tiedote/puusta-elektroniikkaa?publisherId=57858920&releaseId=69906575

Sinclair, Bruce. 2017. IoT Inc:how your company can use the internet of things to win in the outcome economy. New York:McGraw-Hill Education,

Robotiikkatilastot 2022, Automaatioväylä, viitattu 10.1.2023 https://automaatiovayla-fi.sites.avoine.com/site/assets/files/1616/robotiikkatilastot_2022_netti.pdf

STILL ACH- autonomiset mobiilirobotit, yrityksen verkkosivut, viitattu 10.1.2023: https://www.still-trukit.fi/logistiikkajaerjestelmaet/automaatiojaerjestelmaet/ach.html

Suomen Automaatioseuran jaostot, verkkosivut, luettu 28.4.2023: https://www.automaatioseura.fi/sas/jaostot/opc/

Suomen automaatioseuran julkaisu, eri tekijöitä mm. Ahonen, Pasi. Automaation Tietoturva. 2021. Grano, Helsinki.

Sysmac Studio Integrated NS Simulator Youtube video. Viitattu 30.5.2023: https://www.youtube.com/watch?v=ApYTiO3tMfM

Taloustiedot DA-Team, Finder asiakastiedot, viitattu 31.5.2023: https://www.finder.fi/Automaatio/DA-Team+Oy/Muurla/yhteystiedot/2260363

Teollinen Internet- Liiketoiminnan vallankumous, Tukesin ohjelma 2014-2019, viitattu 20.10.2022: https://docplayer.fi/1339445-Teollinen-internet-liiketoiminnan-vallankumous-tekesin-ohjelma-2014-2019.html

Teollinen Internet ja 5G (5GTI)- Toimenpidevalmistelu, viitattu 20.10.2022: https://docplayer.fi/3855367-Teollinen-internet-ja-5g-5gti-toimenpidevalmistelu-trial-seminaari-7-5-2014-mika-klemettinen-tekes.html

Telia IoT, viitattu 30.5.2023: https://www.telia.fi/yrityksille/iot/esineiden-internet

Tolonen, Jyri. Esitys teollisuus 4.0 kilpailussa alustuksena 2018, viitattu 23.5.2023: https://www.hamk.fi/wp-content/uploads/2018/10/Jyri-Tulonen-Teollisuus-4.0-esitys.pdf

Unified Architecture, OPC Foundation verkkosivut, viitattu 30.5.2023: https://opcfoundation.org/about/opc-technologies/opc-ua/

What is OPC? OPC Fountation verkkosivut, viitattu 2.5.2023: https://opcfoundation.org/about/what-is-opc/

Älypöly, Wikipedia artikkeli, luettu 15.12.2022: https://fi.wikipedia.org/wiki/%C3%84lyp%C3%B6ly

Ohjelmaesittely

```
using Opc.Ua;
using Opc.Ua.Client;
using Opc.Ua.Client.Controls;
using Opc.Ua.Configuration;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TestConnectOPCUA
{{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private Session m_session;
        private bool m_connectedOnce;
        private Subscription m_subscription;
        private void connectServerCtrl1_ConnectComplete(object sender, EventArgs
e)
        {
            try
            {
                m_session = connectServerCtrl1.Session;

                // set a suitable initial state.
                if (m_session != null && !m_connectedOnce)
                {
                    m_connectedOnce = true;
                    //connected
                    CreateSubscriptionAndMonitorItem();
                }


            }
            catch (Exception exception)
            {
                ClientUtils.HandleException(this.Text, exception);
            }
        }
        private void CreateSubscriptionAndMonitorItem()
        {
            try
            {
```

```
                if (m_session == null)
                {
                    return;
                }

                if (m_subscription != null)
                {
                    m_session.RemoveSubscription(m_subscription);
                    m_subscription = null;
                }

                m_subscription = new Subscription();
                m_subscription.PublishingEnabled = true;
                m_subscription.PublishingInterval = 1000;
                m_subscription.Priority = 1;
                m_subscription.KeepAliveCount = 10;
                m_subscription.LifetimeCount = 20;
                m_subscription.MaxNotificationsPerPublish = 1000;

                m_session.AddSubscription(m_subscription);
                m_subscription.Create();


                //Method 1
                string[] tags = new string[]
                {
                    "new_Controller_0_GlobalVars_Kissa",
                    "new_Controller_0_GlobalVars_Pekka"
                };



                Control[] ctrls = new Control[]
                {
                    label1,
                    label2
                };

                for (int ii = 0; ii < tags.Length; ii++)
                {
                    ctrls[ii].Text = "---";
                    MonitoredItem monitoredItem = new MonitoredItem();
                    monitoredItem.StartNodeId = new NodeId(tags[ii], 2);
                    monitoredItem.AttributeId = Attributes.Value;
                    monitoredItem.Handle = new object[] { ctrls[ii],
ctrls[ii].GetType().GetProperty("Text") };
                    monitoredItem.Notification += MonitoredItem_Notification1;
                    m_subscription.AddItem(monitoredItem);

                }

                //Method 2:
                this.textBox1.Text = "---";
                MonitoredItem monitoredItem2 = new MonitoredItem();
                monitoredItem2.StartNodeId = new
NodeId("new_Controller_0/GlobalVars/Kissa", 2);
                monitoredItem2.AttributeId = Attributes.Value;
                monitoredItem2.Notification += MonitoredItem2_Notification;
```

```
                m_subscription.AddItem(monitoredItem2);

                //Method 3:

                MonitoredItem monitoredItem3 = new MonitoredItem();
                monitoredItem3.StartNodeId = new
NodeId("new_Controller_0/GlobalVars/Pekka", 2);
                monitoredItem3.AttributeId = Attributes.Value;
                monitoredItem3.Notification += MonitoredItem3_Notification;
                m_subscription.AddItem(monitoredItem3);

                m_subscription.ApplyChanges();
            }
            catch (Exception exception)
            {
                ClientUtils.HandleException(this.Text, exception);
            }
        }

        private void MonitoredItem3_Notification(MonitoredItem monitoredItem,
MonitoredItemNotificationEventArgs e)
        {
            if (InvokeRequired)
            {
                BeginInvoke(new
MonitoredItemNotificationEventHandler(MonitoredItem3_Notification),
monitoredItem, e);
                return;
            }
            try
            {
                if ((bool)
((MonitoredItemNotification)e.NotificationValue).Value.WrappedValue.Value ==
false)
                {
                    this.pictureBox1.Image =
global::TestConnectOPCUA.Properties.Resources.BlueButton;
                }
                else
                {
                    this.pictureBox1.Image =
global::TestConnectOPCUA.Properties.Resources.BlackButtonPressed;
                }
            }
            catch (Exception ex)
            {
                ClientUtils.HandleException(this.Text, ex);
            }
        }

        private void MonitoredItem2_Notification(MonitoredItem monitoredItem,
MonitoredItemNotificationEventArgs e)
        {
            if (InvokeRequired)
            {
                BeginInvoke(new
MonitoredItemNotificationEventHandler(MonitoredItem2_Notification),
monitoredItem, e);
```

```
                return;
            }
            this.textBox1.Text =
((MonitoredItemNotification)e.NotificationValue).Value.WrappedValue.ToString();

        }

        private void MonitoredItem_Notification1(MonitoredItem monitoredItem,
MonitoredItemNotificationEventArgs e)
        {
            if (InvokeRequired)
            {
                BeginInvoke(new
MonitoredItemNotificationEventHandler(MonitoredItem_Notification1),
monitoredItem, e);
                return;
            }

            try
            {
                object[] objs = (object[])monitoredItem.Handle;
                Control control = (Control)objs[0];
                PropertyInfo proInfo = (PropertyInfo)objs[1];
                MonitoredItemNotification datachange = e.NotificationValue as
MonitoredItemNotification;

                if (datachange == null)
                {
                    return;
                }
                object v =
TypeDescriptor.GetConverter(datachange.Value.WrappedValue.Value.GetType()).Conve
rtTo(datachange.Value.WrappedValue, proInfo.PropertyType);
                if (proInfo != null) proInfo.SetValue(control, v);
            }
            catch (Exception exception)
            {
                ClientUtils.HandleException(this.Text, exception);
            }
        }

        private void MonitoredItem_Notification(MonitoredItem monitoredItem,
MonitoredItemNotificationEventArgs e)
        {
            throw new NotImplementedException();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.connectServerCtrl1.ServerUrl = "opc.tcp://192.168.250.1:4840";
            string AppName = "ToukoAalto";//Issusedto
            ApplicationConfiguration config = new ApplicationConfiguration()
            {
                ApplicationName = AppName,
                ApplicationUri = Utils.Format(@"urn:{0}:" + AppName,
System.Net.Dns.GetHostName()),
                ApplicationType = ApplicationType.Client,
                SecurityConfiguration = new SecurityConfiguration
```

```
                {
                    ApplicationCertificate = new CertificateIdentifier
                    {
                        StoreType = @"Directory",
                        StorePath = System.Windows.Forms.Application.StartupPath
+ @"\Cert\TrustedIssuer",
                        SubjectName = "CN=" + AppName + ", DC=" +
System.Net.Dns.GetHostName()
                    },
                    TrustedIssuerCertificates = new CertificateTrustList
                    {
                        StoreType = @"Directory",
                        StorePath = System.Windows.Forms.Application.StartupPath
+ @"\Cert\TrustedIssuer"
                    },
                    TrustedPeerCertificates = new CertificateTrustList
                    {
                        StoreType = @"Directory",
                        StorePath = System.Windows.Forms.Application.StartupPath
+ @"\Cert\TrustedIssuer"
                    },
                    RejectedCertificateStore = new CertificateTrustList
                    {
                        StoreType = @"Directory",
                        StorePath = System.Windows.Forms.Application.StartupPath
+ @"\Cert\RejectedCertificates"
                    },
                    AutoAcceptUntrustedCertificates = true,
                    AddAppCertToTrustedStore = true,
                    RejectSHA1SignedCertificates = false//important
                },
                TransportConfigurations = new
TransportConfigurationCollection(),
                TransportQuotas = new TransportQuotas { OperationTimeout = 15000
},
                ClientConfiguration = new ClientConfiguration
{ DefaultSessionTimeout = 60000 },
                TraceConfiguration = new TraceConfiguration
                {
                    DeleteOnLoad = true
                },
                DisableHiResClock = false

            };
            config.Validate(ApplicationType.Client).GetAwaiter().GetResult();
            if (config.SecurityConfiguration.AutoAcceptUntrustedCertificates)
            {
                config.CertificateValidator.CertificateValidation += (s, ee) =>
                { ee.Accept = (ee.Error.StatusCode ==
StatusCodes.BadCertificateUntrusted); };
            }

            this.connectServerCtrl1.Configuration = config;
            this.connectServerCtrl1.UserIdentity = new UserIdentity("kissa",
"koirakoira"); //if you want to login with user and pass
            //this.connectServerCtrl1.UserIdentity = new UserIdentity();
            this.connectServerCtrl1.UseSecurity = true;
```

```
            var application = new ApplicationInstance
            {
                ApplicationName = AppName,
                ApplicationType = ApplicationType.Client,
                ApplicationConfiguration = config

            };
            //set 0 Trace mask=>stop show log in output window.
            Opc.Ua.Utils.SetTraceMask(0);//
            application.CheckApplicationInstanceCertificate(true,
2048).GetAwaiter().GetResult();//create certificate

        }

        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
            this.connectServerCtrl1.Disconnect();
        }

        private void label3_Click(object sender, EventArgs e)
        {
            label3.Text =
System.Windows.Forms.Application.StartupPath.ToString();
        }
    }
}
```