Arto Keiski

OPC UA ja teollinen asioiden internet

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Automaatiotekniikka
Insinöörityö

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Arto Keiski OPC UA ja teollinen asioiden internet 43 sivua + 2 liitettä 29.02.16
Tutkinto	insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	
Ohjaaja(t)	Lehtori Antti Liljaniemi

Tutkielmassa perehdyttiin OPC:n historiaan, sekä tutkittiin syitä nykyisen version, OPC UA:n, kehitykseen. OPC UA:n määritelmistä esitetään pääkohdat. OPC:n ja OPC UA:n käyttöä teollisuudessa pohdittiin, sekä OPC UA:n käyttöä IIoT:n yhteydessä.

Lisäksi OPC UA:sta tehtiin harjoitustehtävä, jossa tulee tehdä tietoja palvelimelta lukeva asiakas sekä asiakkaan käyttöliittymä. Lisätehtäväksi määriteltiin OPC UA:n liittäminen pilvipalveluun.

OPC saavutti heti julkaisustaan vuonna 1996 suuren suosion teollisuusautomaatiossa, ratkaisten ajuriongelman, eli tarpeen kirjoittaa omat ajurit kymmenille eri järjestelmille ja uusia ne ohjelmistoversioiden muuttuessa. OPC tukeutui Microsoftin DCOM:n tiedonsiirrossa. Yhteen käyttöjärjestelmään sitoutuminen ja sen mukanaan tuomat ongelmat mm. tietoturvallisuudessa johtivat ajatuksiin verkkopalveluilla käytettävästä käyttöjärjestelmäriippumattomasta uudesta versiosta.

Vuonna 2006 julkaistu OPC UA on yhteensopiva OPC:n kanssa, mutta perustuu verkkopalveluihin ja on riippumaton käyttöjärjestelmästä. Tietoturvallisuus huomioitiin kehityksen alusta asti. UA:ssa on yhtenäinen osoiteavaruus, UA tukee olio-ohjelmointia, datan kuvauksessa voidaan käyttää metadataa ja UA myös skaalautuu hyvin.

Esineiden internet, IoT, sekä sen teollinen vastine IIoT, ovat kiivaan tutkimuksen kohteena ja nopeimmat yritykset ovat saaneet niistä uusia liiketoimintamalleja. IIoT:n merkityksen odotetaan kasvavan suuresti lähivuosina, kiihdyttävän taloutta ja muuttavan tuotantomalleja.

Avainsanat	OPC, OPC UA, IoT, IIoT

Author(s) Title Number of Pages Date	Arto Keiski OPC UA and Industrial Internet of Things 43 pages+ 2 appendices 02/29/16
Degree	Bachelor of Engineering
Degree Programme	Automation Engineering
Specialization option	
Instructor(s)	Antti Liljaniemi, Lecturer

This study familiarized with the history of OPC, and causes of the development of OPC UA. The main elements of OPC UA definitions are covered. Use of OPC UA and OPC in industry were discussed, as well as OPC UA's use in the context of IIoT.

In addition, an exercise of OPC UA was made, where students have to make a client to a server and an user interface to the client.

OPC reached high popularity in industrial automation immediately after its release in 1996, solving the need to write drivers for dozens of different automation systems and to write them again after each update. OPC relied on Microsoft's DCOM for data transmission. Commitment to a single operating system brought with it problems, such as lack of data security. This led to the ideas of a new version, which is independent of operating system and based on web services.

OPC UA published in 2006 is compatible with the OPC and is based on web services and independent of the operating system. Information security was taken into account from the beginning of development. UA has a unified address space, it supports object-oriented programming and meta-data can be used to describe data. UA is also highly scalable.

Internet of Things, IoT, as well as its industrial counterpart, IIoT, are the subject of intense study today. IIoT's significance is expected to increase greatly in the coming years, and to accelerate economy and to change production patterns.

Keywords	OPC, OPC UA, IoT, IIoT

Sisällys

Lyhenteet

COM	Microsoftin Component Object Model. Mahdollistaa ohjelmistojen tiedonvaihdon.
DCOM	Distributed COM. Tietojen vaihto tietokoneiden välillä.
DCS	Distributed Control System. Hajautettu ohjausjärjestelmä.
ERP	Enterprise Resource Planning. Yrityksen toiminnanohjausjärjestelmä.
HMI	Human–machine interface. Käyttöliittymä.
HTTP	Hypertext Transfer Protocol. Verkkopalveluiden ja -selainten protokolla.
IEC	International Electrotechnical Commission. Standardointiorganisaatio.
IIC	Industrial Internet Consortium. Teollisen IoT:n avoin etujärjestö.
IoT	Internet Of Things. Esineiden internet.
ISA	Internal Society of Automation. Standardointiorganisaatio.
MES	Manufacturing execution systems. Tuotannonohjaus.
M2M	Machine to Machine. Koneiden välinen viestintä.
OLE	Object linking and embedding. Objektin linkittäminen ja upottaminen.
OPC	OLE for Process Control. OLE prosessin ohjaukseen.
OPC UA	OPC Unified Architecture.
PLC	Programmable Logic Controller. Ohjelmoitava logiikka.
TCP	Transmission Control Protocol. Ethernetin/internetin perusprotokolla.
SCADA	Supervisory Control And Data Acquisition. Teollinen ohjausjärjestelmä.
SDK	Software Development Kit. Sovelluskehitysympäristö.
SOAP	Simple Object Access Protocol. Protokolla XML viestien välitykseen.
SQL	Structured Query Language. Kieli relaatiotietokantojen käsittelyyn.
WSS	Web Service Security. SOAP turvallisuuslaajennus.
XML	Extensible Markup Language. Standardoitu rakenteellinen kuvauskieli.

1 Johdanto

Insinöörityön tarkoituksena on perehtyä ja laatia tutkielma OPC:n uusimmasta kommunikaatioarkkitehtuurista OPC UA:sta. Työssä tulee myös tarkastella OPC UA:n soveltamista teollisuudessa sekä teollista esineiden internetiä, IIoT:tä.

OPC UA:sta tuli tehdä myös työn tilaajalle, Metropolia Ammattikorkeakoululle, harjoitustehtävä automaatiolaboratorioon. Tehtävä tuli tehdä c#-kielellä Microsoftin Visual Studio -sovelluskehitysympäristössä. Tehtävässä tuli toteuttaa palvelimeen yhdistyminen ja yhteyden katkaisu, lukeminen ja kirjoitus sekä kirjautuminen muuttujaan arvon muutosten seuraamista varten.

OPC, viestintäprotokolla teollisuuden laitteille, saavutti suosiota jo ensimmäisessä versiossaan. Sillä päästiin eroon lukuisista toimittajariippuvaisista automaatiojärjestelmistä, ja se saavutti teollisuusstandardin aseman. Uudistetun yhtenäisen osoiteavaruuden OPC:n (OPC Unified Architecture) myötä suosio on kasvanut. OPC UA:sta on tehty myös IEC:n standardi 62541. Tärkein tekninen ero OPC:n ja OPC UA:n välillä on Microsoftin COM:n vaihto verkkopalvelukeskeiseen arkkitehtuuriin, jonka myötä UA:n käyttö tuli yksinkertaisemmaksi ja valmistajariippumattomaksi. UA:n ohjelmointi on mahdollista lukuisilla eri ohjelmointikielillä. Lisäksi UA:ssa toteutettiin yhtenäinen osoiteavaruus, ja tietoturva otettiin mukaan jo suunnitteluvaiheessa. UA skaalautuu hyvin. Tarkoituksella abstrakteiksi tehdyt määrittelyt mahdollistavat helpon päivityksen teknologian kehittyessä.

IoT eli esineiden internet sekä IIoT, teollinen esineiden internet, ovat suuren kiinnostuksen kohteina ja kehittyvät nopeasti. Niiden taloudellisen vaikutuksen arvioidaan kasvavan lähivuosina huomattavasti sekä lisäävän uusia tuotantomahdollisuuksia ja työpaikkoja.

2 OPC:n historiaa

1990-luvulla markkinoilla oli lukuisia eri yritysten automaatiojärjestelmiä ja väyläratkaisuja, joista jokainen vaati omat ajurinsa järjestelmää tukeville laitteille. Ohjelmistotoimittajat joutuivat tuottamaan ajurit jopa sadoille eri järjestelmien laitteille tuodessaan uusia automaatio-ohjelmia myyntiin. Tämä sitoi suuren osan yrityksen resursseista vain ajurien tuotantoon ja ylläpitoon. Ongelmaa ratkaisemaan yritykset Fisher-Rosemount, Intellution, Intuitive Technology, Opto22, Rockwell ja Siemens AG kokoontuivat 1995 ja perustivat OPC-työryhmän (Object Linking and Embedding for Process Control). Mukana oli myös Microsoftin edustajia antamassa teknistä tukea. [1, s. 6; 2, s. 4; 3, s. 1.]

Microsoftin Windows oli 1990-luvulla lähes monopoliasemassa pc-koneiden käyttöjärjestelmänä. Samaan aikaan pc:t halpenivat ja niiden suorituskyky kasvoi. Jotkut yritykset kokivat ohjelmoitavat logiikat ylihintaisiksi ominaisuuksiinsa nähden ja toivat siksi Windows-pc:t teollisuuteen [1, s. 5]. Microsoft tuotti näihin aikoihin myös COM- ja DCOM-teknologiat tiedon jakamiseen Windows-ohjelmien ja -tietokoneiden välillä. Niinpä OPC-työryhmän oli luontevaa hyödyntää olemassa oleva valmis teknologia OPC:n tiedonsiirtoprotokollaksi. Tämä oli osaksi syynä siihen, että OPC 1.0 voitiin julkaista jo vuonna 1996. [1, s. 7; 2, s. 5; 4.]

Ilmestyessään OPC oli lähes vallankumouksellinen asia teollisuusautomaatiossa: mikä tahansa Microsoftin COM:a tukeva ohjelma kykeni lukemaan tietoa antureilta ja kirjoittamaan käskyjä toimilaitteille. Ensimmäistä kertaa sovelluskehittäjien ei tarvinnut panostaa ajurien kehitykseen. Tämä kaikki nopeutti pc:n tuloa tehtaiden tuotantotiloihin. [1, s. 6–7.]

Ajurien ja ohjelmien eriyttäminen nopeutti myös sovellus- ja laitetuotantoa, ja jo muutamassa vuodessa markkinoilla oli saatavilla tuhansia OPC-tuotteita. Nopea kasvu oli osoitus uuden tekniikan hyväksynnästä. OPC:stä tulikin teollisuusstandardi jo muutaman vuoden kuluttua julkaisustaan. [1, s. 9; 2, s. 22.]

2.1 OPC-määrittelyt: DA, A&E ja HDA

OPC-määrittelyt eli spesifikaatiot DA (Data Access) datan käsittelylle julkaistiin vuonna 1997, A&E (Alarms & Events) hälytyksille ja tapahtumille vuonna 1999 ja HDA (Historical Data Access) historiallisen datan käsittelyyn vuonna 2000. Nämä ovat OPC:n päämääritelmät. Omat määrittelynsä julkaistiin myös turvallisuudelle (Security) ja OPC DA:n laajennuksina määrittelyt eräajoille (Batch), yhdistelmädatalle (Complex) ja datan vaihdolle (Data eXchange). [2, s. 5; 3, s. 3–7.]

2.2 OPC XML-DA

OPC XML-DA -määrittely OPC DA:n laajennuksena julkaistiin vuonna 2003. Tämä oli ensimmäinen käyttöjärjestelmäriippumaton OPC-määrittely säilyttäen kuitenkin OPC DA:n toiminnallisuuden. XML-DA:ssa data siirretään verkkopalveluna käyttäen laajasti teollisuudessa käytössä olevia rakenteellista tiedonkuvaustapaa XML:ää ja SOAP:ta, joka on protokolla XML-viestien välitykseen. XML-DA suunniteltiin internetiin liittymistä ja yritysverkkoon integroitumista varten. XML-DA osoittautui kuitenkin resursseja kuluttavaksi ja hitaaksi tiedon välitystavaksi eikä täyttänyt siihen kohdistettuja odotuksia. [2, s. 6; 3, s. 7.]

3 OPC-säätiö

Spesifikaatioiden haltijaksi perustettiin yrityskyselyn jälkeen riippumaton ja voittoa tuottamaton OPC-säätiö (OPC Foundation) vuonna 1996 [4]. Lähes kaikki teollisuusautomaation toimittajat liittyivät pian OPC-säätiöön. Säätiön tehtävinä on muokata ja laajentaa määrittelyitä yritysmaailman toiveiden mukaisesti, säilyttäen yhteensopivuus aiempien versioiden kanssa. Säätiön tulee myös säilyttää määritelty tavoitteensa selvänä ja säilyä poliittisen kiistelyn ulkopuolella. Säätiö tukee jäseniään OPC:n käyttöön liittyvissä asioissa. Säätiön rahoitus tulee kokonaan yritysten jäsenmaksuista, eikä se saa rahallista tukea valtioita tai muilta organisaatioilta. [2, s. 5, 16; 3, s. 1; 5, s. 32.] Säätiön nettisivulla www.opcfoundation.org on laaja aineisto OPC:sta, osa saatavilla vain jäsenille.

OPC-säätiö on onnistunut säilyttämään OPC:n yhtenäisenä, estäen OPC:n jakautumisen yhteensopimattomiin versioihin. Säätiön alueellisia organisaatioita ovat OPC Eurooppa, OPC Japani ja OPC Kiina. Ne palvelevat alueellisia jäseniä omilla kielillään. Jäseninä säätiössä ovat yritysjäsenet, loppukäyttäjäjäsenet ja äänettömät jäsenet. Yritysjäsenten vuotuinen jäsenmaksu riippuu liikevaihdosta. Loppukäyttäjäjäsenillä ja äänettömillä jäsenillä on kiinteä jäsenmaksu [6]. Äänettömiksi jäseniksi hyväksytään yliopistoja ja vastaavia organisaatioita. [2, s. 18–19; 3, s. 2.]

Testausohjelma

OPC-säätiö on myös laatinut kaksiportaisen testauskäytännön, jonka tarkoitus on varmistaa OPC-sertifioitujen tuotteiden yhteensopivuus. Loppukäyttäjien on voitava luottaa siihen, että eri valmistajien OPC-sertifioidut tuotteet toimivat yhdessä. [2, s. 251; 3, s. 2, 302.]

Testausohjelman ensimmäinen vaihe

Ensimmäinen porras sisältää valmistajan itse suorittaman testauksen sekä ryhmätestaustapahtumat. Itsetestauksessa valmistaja käyttää OPC-säätiön tarjoamaa testiohjelmaa. Testitulokset lähetetään internetin kautta salattuna OPC-säätiölle. Ryhmätestaustapahtumia järjestetään vuosittain Euroopassa, Japanissa ja Pohjois-Amerikassa. Näissä eri valmistajat testaavat tuotteidensa yhteensopivuuden. Kun yhteensopivuus-

testit on läpäisty valmistaja voi käyttää itsetestattu-logoa (Self-tested, kuva 1). [2, s. 266; 3, s. 2, 302.]

Testausohjelman toinen vaihe

Toisessa tasossa riippumattomat testilaboratoriot suorittavat syvällisemmät yhteensopivuustestit. Tämän tason testit sisältävät perustason toiminnallisuuden testaamisen lisäksi esimerkiksi kuormitustestejä, yhteiskäyttötestejä ja käytettävyystestejä. Tämän tason testien suorittamisen jälkeen tuotteissa saa käyttää OPC:n varmentama (OPC Certified) -logoa (kuva 2). Loppukäyttäjiä suositellaan käyttävän vain OPC-sertifioituja tuotteita. OPC-säätiön sivuilla, www.opcfoundation.com, on luettelo OPC-tuotteista, joka tuotekuvauksen lisäksi ilmoittaa tuotteelle suoritetut yhteensopivuustestit. [1, s. 51; 2, s. 266; 3, s. 2, 302.]

Kuva 1. Logo ensimmäisen testivaiheen läpäisseille tuotteille [3, s. 302]

Kuva 2. Logo OPC:n testaamille tuotteille [3, s. 302].

4 Perusteet OPC UA:lle

OPC:stä tuli hyvin suosittu heti julkistuksestaan lähtien. OPC-pohjaisia tuotteita on asennettu miljooniin kohteisiin, yli 3 500 toimittajalta on saatavilla yli 20 000 OPC-tuotetta. OPC:tä käytetään monilla teollisuuden aloilla ympäri maailman myös aloilla, mihin sitä ei alun perin suunniteltu. OPC:tä käytetään useanlaisissa käyttötarkoituksissa: horisontaalisen ja vertikaalisen datan yhdentämisessä, käyttöliittymissä (HMI), teollisissa ohjausjärjestelmissä (SCADA), hajautetussa prosessinohjauksessa (DCS), ohjelmoitavissa logiikoissa (PLC), valmistuksen ohjauksessa (MES) ja yrityksen toiminnanohjausjärjestelmissä (ERP). [2, s. 6, 86; 3, s. 8.]

Kaikesta menestyksestä huolimatta OPC:ssä oli myös ongelmia. Microsoft Windowsin COM/DCOM-tekniikka mahdollisti OPC:n nopean leviämisen, mutta oli samalla suurin tyytymättömyyden aihe OPC:n käytössä ja monessa tapauksessa esti OPC:n käytön kokonaan. [1, s. 9; 2, s. 89; 3, s. 8.]

4.1 Yhteen käyttöjärjestelmään sitoutuminen

DCOM sitoo OPC:n Microsoft Windowsiin, muita käyttöjärjestelmiä ei tueta. Usein valmistavan teollisuuden tuotantolinjat suunnitellaan huomattavasti pidempiaikaiseen käyttöön kuin Microsoft Windowsin päivityssykli [1, s. 11]. Esimerkiksi lääke- ja kemianyritykset eivät ole suosineet Windowsia tietoturvaongelmien ja lyhyiden Windows-versioiden tukiaikojen vuoksi. IT-ala taas on suosinut UNIX- ja Linux-käyttöjärjestelmiä. Sulautetuissa järjestelmissä Microsoft on vasta aivan viime vuosina saavuttanut menestystä. [2, s. 89.]

4.2 Turvallisuusasetusten hankaluus

Helppo verkkoasetusten teko oli yksi OPC:n menestystekijöistä. DCOM:n turvallisuusasetukset olivat poikkeus. DCOM:n turvallisuusasetukset muuttuivat Windows-versioiden välillä, niiden hallinta oli monimutkaista ja vaati syvällistä perehtyneisyyttä. Asennusprosessia nopeuttaakseen insinöörit usein tekivät hyvin väljät tietoturva-asetukset, mikä altisti vihamielisille verkkohyökkäyksille. Usein operaattorit ”korjasivat” toimimattoman verkkoyhteyden poistamalla DCOM:n turvallisuustarkistukset. DCOM:n turvallisuusasetuksiin liittyvät kyselyt ovat enemmistönä OPC-toimittajille osoitetuissa tukipyynnöissä. [1, s. 10; 2, s. 88.]

4.3 Toimintakatkot

OPC DA määritteli, kuinka tiheään prosessin tila tarkistetaan. Tarkistusväli on säädettävissä. Jos prosessin tila vaihtuu tiheämmin kuin tarkistusväli, dataa kadotetaan. Jos asiakkaan ja palvelimen välisessä fyysisessä yhteyslinkissä on häiriöitä, määritelmän mukaan sen tiedonsiirto katkaistaan. Yhteys palaa jälleen, kun fyysisen yhteyden korjaamisen jälkeen OPC-yhteys avataan uudelleen 6 minuutin kuluttua, eikä aikaa voi säätää. Katkon aikana tapahtuneet prosessin tilan muutokset menetetään. Jos OPC:tä käytetään esimerkiksi trenditietojen keräämiseen, lyhyet katkot eivät ole kriittisiä. OPC:n suosion myötä sitä käytetään myös aloilla, joissa katkoja yhteydessä ei voi hyväksyä, esimerkiksi kemian- ja lääketeollisuudessa. Tähän eräät toimittajat ovat reagoineet tekemällä yrityskohtaisia laajennoksia kuten puskurointi, nopea katkoksen havainnointi ja uudelleenkytkentä. Laajennokset ovat toimivia, mutta kuitenkin yrityskohtaisina toteutuksina OPC-määritelmien ulkopuolelta. [2, s. 88, 91, 282.]

4.4 Toimintaongelmat internetissä

OPC:ssa konekohtaisen rajan ylittäminen on läpinäkyvää eikä käyttäjän kannalta ole väliä, onko OPC-palvelin samassa tai toisessa tietokoneessa. DCOM:n turvallisuusasetukset samassa verkossa (intranetissä) ovat varsin suoraviivaisia. Kommunikointi internetin kautta on ongelmallista. Internetissä tapahtuva dataliikenne vaatii nykypäivänä riittäviä turvallisuusasetuksia. Internetin TCP-liikenne tapahtuu porttien kautta. www-osoitteita hallinnoiva viranomainen IANA ylläpitää listaa varatuista porttinumeroista. Tunnettuja varattuja portteja ovat esimerkiksi 21 tietojensiirtoprotokolla FTP:lle, 22 salatulle tietojensiirtoprotokollalle ssh:lle ja 80 verkkoselainten protokollalle HTTP:lle [7]. DCOM vaatii monta avointa porttia, mm. yhteyden avaamiseen, käyttäjän todentamiseen, datan välitykseen jne. DCOM arpoo porttinumerot kullekin tarvitsemalleen yhteydelle. Jos portti ei ole vapaana DCOM arpoo uuden. Seurauksena edellisestä palomuurista täytyisi avata suuri määrä portteja DCOM:n yhteyksiä varten. Jokainen avattu portti palomuurissa on potentiaalinen turvallisuusuhka. Näin toimiessaan DCOM myös poissulkee mahdollisuuden käyttää palomuuria. DCOM:a ei myöskään voi käyttää osoitteenmuunnoksen eli NAT:n (Network Address Translation) kanssa. Osoitteenmuunnoksella esimerkiksi yksi internetpalveluntarjoajan ip-osoite jaetaan monelle tietokoneelle. [2, s. 3, 88; 3, s .4.]

DCOM ei myöskään tue metadataa, eli tietoa tiedosta [1, s. 12.]

4.5 Verkkopalvelut

Microsoft julkisti uuden .NET-sovelluskehyksen vuonna 2002 ja ilmoitti samalla luopuvansa DCOM:n kehittämisestä. Tarkoitus ei ollut poistaa tukea välittömästi seuraavista Windows-versioista, mutta kehityksen loppuessa oli selvää, että DCOM-tekniikka vanhenee ja poistetaan tulevaisuudessa. OPC-säätiö reagoi nopeasti, ja XML DA -määritelmässä vuonna 2003 säätiö ensimmäisen kerran käytti Microsoftin DCOM:sta riippumatonta käyttöjärjestelmävapaata verkkoteknologiaa. XML DA:ta käytetään esimerkiksi rakennusautomaatiossa, energiateollisuudessa sekä prosessiteollisuudessa. Käytössä on UNIX- ja Linux-palvelemia sekä kenttätason laitteita, joissa monissa on sulautettuna käyttöjärjestelmänä Linux. [2, s. 89.]

XML-DA käyttää SOAP XML -protokollaa ja kommunikointi tapahtuu HTTP-viesteinä, joissa OPC-data on koodattu tekstiin. Datan koodaus tekstiksi ja takaisin vie koneaikaa, siksi XML-DA on 5–7 kertaa hitaampi kuin DA. Hitaus oli liikaa monille teollisuuden aloille, mutta ensi askel toimivasta verkkopalvelutekniikasta oli otettu. Nopeampi datan välitys jäi kehityskohteeksi. [2, s. 89.]

4.6 Yhtenäisen osoiteavaruuden puute

Erityisesti prosessi- ja rakennusteollisuus esittivät pyyntöjä OPC DA:n, A&E:n ja HDA:n osoiteavaruuksien yhdistämisestä. OPC:ssä näiden kolmen objektimallit erosivat suuresti toisistaan. Nämä olivat kuitenkin mahdollista toteuttaa samassa sovelluksessa. Eri objektimallien vuoksi tiedon käsittely käytettäessä eri malleja yhdessä oli hidasta ja hankalaa. Yhteinen osoiteavaruus hyödyttäisi OPC-tuotteiden toimittajia, operaattoreita ja loppukäyttäjiä. [2, s. 90.]

4.7 Tuki yhdistelmädatalle

Yksi OPC:n pääsovelluskohteista on ollut sarjaliikenneprotokollalla ohjattujen tai kenttäväyliin kytkettyjen laitteiden valvonta. Tietojen lukemiseen ja kirjoittamiseen riittävät yksinkertaiset datatyypit, kuten tavu (byte), kokonaisluku (integer), reaaliluku (real) ja taulukot (arrays). Säätiölle tuli pyyntöjä myös strukturoidun eli rakenteellisen datan tukemiseen. Niitä tarvittiin esimerkiksi kenttälaitteiden asetusten tekoon, rakenne kunkin väylän valmistajan mukaan. [2, s. 90.]

4.8 Yhdistelmädatan määrittely

OPC julkaisi yhdistelmädatan määrittelyn (Complex Data Specification), mutta julkaisu tapahtui vuonna 2003, vuosia päämäärittelyjen jälkeen. OPC-asennuksia oli tehty jo tuhansittain, siksi määritelmä jäi vähäiselle huomiolle. Tilannetta tuli parantaa jatkossa. [2, s. 91.]

4.9 Suojaus luvattomalta käytöltä

Ethernet alkaa olla hallitseva tiedonsiirtostandardi automaatiossakin. Näin automaatio- ja toimistoverkot yhdentyvät mikä parantaa vertikaalista integraatiota. Prosessitason dataa OPC-palvelimelta voidaan tallentaa OPC-asiakassovelluksena tietokantaan tai esittää OPC-asiakassovelluksena vaikka Excel-taulukossa. Rakennuksen normaali verkkokaapelointi riittää. [2, s. 91.]

Yhdentyminen tuo mukanaan myös uusia tietoturvariskejä. Asennuksessa tulee huomioida suojautuminen luvattomalta tietojen käytöltä ja muokkaukselta. OPC:ta käytetään myös etäkäytöllä verkon yli, myös tässä yhteydessä tulee huomioida samat uhat. Verkkorikollisuus kasvaa ja saa uusia muotoja. OPC:ssä asiaan ei ollut paneuduttu, tietoturva jäi yritysten sovellusten varaan. [1, s. 10; 2, s. 91.]

4.10 Metodien käyttö

Monissa kohteissa tietojen lukeminen ja kirjoittaminen eivät enää riitäkään. Myös komentoja tai metodeja, kuten pysäytys, käynnistys, tallenna tiedot tietokantaan jne. tarvitaan. OPC-komentomäärittelyn (OPC Commands Specification) tekeminen aloitettiin. Luonnosversio määrittelystä on saatavilla. Määrittely ei kuitenkaan ehtinyt valmiiksi ennen kuin aloitettiin OPC UA:n määrittelyjen työstäminen. [2, s. 92.]

5 OPC UA:n kehitys

Uuden OPC arkkitehtuurin kehitys alkoi vuonna 2003, kun hälytysten ja tapahtumien (A&E) työryhmä oli suunnittelemassa uutta määritelmäversiota verkkopalveluilla käytettäväksi. Hahmotelmat johtivat uuden työryhmän perustamiseen, jonka tehtäviksi asetettiin DA:n, A&E:n ja HDA:n yhdistäminen samaan osoiteavaruuteen sekä muuntaminen verkkopalveluilla käytettäviksi. Tämä oli lähtöpiste OPC UA:lle (Unified Architecture). [2, s. 92.]

5.1 OPC UA:n tavoitteet

OPC-säätiö teki markkinatutkimuksia ja kyselyitä OPC:n käyttäjiltä ja tuotetoimittajilta. Näiden perusteella tehtiin OPC UA:n tavoitteet [1, s. 13–16; 2, s. 92; 3, s. 8]

- selkeä rakenne, peruskäytön pitää olla helppoa
- yhteensopivuus OPC:n kanssa, näin turvataan OPC:hen sijoitettu pääoma
- yhteinen malli kaikelle datalle
- oliopohjainen (OOP), metadata
- käyttöjärjestelmäriippumattomuus ja skaalattavuus
- abstrakti perusmalli, helpottaa tulevia laajennuksia ja päivityksiä
- suojattu käyttö, suojaudutaan verkkouhilta
- datan varmuus, vakaa rakenne, luotettava tiedonsiirto
- redundanssilla ym. keinoilla varmistetaan ettei dataa menetetä
- hyvä suorituskyky, nopea tiedonsiirto, toimivuus internetissä ja palomuurien läpi.

OPC oli suunniteltu liittymästandardiksi laitteiden ja ohjelmistojen väliin. OPC UA:sta ei haluttu vain päivitettyä OPC-versiota, vaan tavoitteena oli kattava yhteensopivuus ja standardoitu tiedonsiirto riippumatta toimittajasta, ohjelmointikielestä, käyttöjärjestelmästä ja paikasta. [2, s. 93.]

5.2 Kehitysryhmä

OPC UA:n kehitysryhmässä toimi työntekijöitä 30 yrityksestä. Monet näistä olivat markkinajohtajia aloillaan ja kilpailijoita keskenään. Jäsenyritysten lisäksi kehitysryhmässä oli mukana myös standardointiorganisaatioita, kuten IEC ja ISA, jotka olivat mukana työryhmän alkuaikoina. Organisaatiot olivat kiinnostuneita käyttämään OPC UA:ta tie-

donsiirtoon. OPC-säätiö määrittelee, miten välitettävä tieto kuvataan ja siirretään. Organisaatiot määrittelevät mitä tietoa kuvataan standardoitujen informaatiomalliensa mukaisesti. [2, s. 94; 3, s. 9.]

6 OPC UA:n määrittelyt

OPC UA julkaistiin vuonna 2006 [8, s. 22]. Määrittelyt kattavat kesällä 2015 osat 1–14, yhteensä yli 700 sivua. On sanottu, että OPC UA on sisäisesti mutkikas, mutta kuitenkin helppo käyttää [8, s. 25]. Palveluita, joilla OPC UA:ta käytetään, on 37 [3, s. 130; 8, s. 21].

Osissa 1–7 määritellään OPC UA:n perustoiminta, osissa 8–11 määritelmät OPC:n mukaisesti jaetuille tietomalleille (DA, A&E, Programs, HDA). Osa 12 on määritelmä palvelimen löytymiselle, osassa 13 on määritelty tapa yhdistellä dataa ja osassa 14 ovat päivitykset sekä virheiden korjaukset.

6.1 Yleisesittely ja käsitteet

Ensimmäisessä määritelmässä [9] esitellään UA:n tavoitteet ja esitellään lyhyesti mallit niiden tavoittamiseksi. OPC UA on sovellettavissa esimerkiksi kenttätason laitteiden ohjaukseen, ohjausjärjestelmiin sekä MES- ja ERP-järjestelmiin. Näissä kohteissa välitetään dataa ja ohjataan teollisia prosesseja.

OPC UA käyttää asiakas-palvelinarkkitehtuuria. OPC UA:ta käyttävässä järjestelmässä (esim. PLC, tietokone) voi olla useita asiakkaita sekä palvelimia. Asiakas voi olla yhtäaikaisesti yhteydessä yhteen tai useampaan palvelimeen. Palvelin voi olla yhtäaikaisesti yhteydessä yhteen tai useampaan asiakkaaseen. Yhdessä sovelluksessa voi olla toteutettuna sekä asiakas että palvelin. Sovellus voi toimia asiakkaana toiselle palvelimelle tai palvelimena asiakkaalle. [9, s. 9.]

OPC UA:n määritelmät ovat kerroksittaisia. Toiminta on eristetty tietokoneen sekä verkkoyhteyden teknologioista. Näin teknologioiden muutokset eivät pakota muuttamaan perusjärjestelmää. [9, s. 9.]

OPC UA:ssa on huomioitu laajasti käytössä oleva vanha Microsoftin COM-teknologiaan perustuva OPC. Jos ei halua heti siirtyä UA:han saatavilla on emulaattoreita molempiin suuntiin OPC COM:sta UA:han tai päinvastoin. [2, s. 329; 3, s. 293.]

Osoiteavaruus

OPC UA on käyttöjärjestelmäriippumaton ja siinä on yhtenäinen osoiteavaruus, jossa yksittäisessä palvelimessa voivat olla data, tapahtumat ja hälytykset sekä historia. Osoiteavaruuteen pääsee määritellyillä palveluilla. Palveluissa on mukana UA:han integroitu tietoturva. Palvelin voi esitellä asiakkaille, mitä palveluita tukee. OPC UA voi käyttää erilaisia tiedonsiirtoprotokollia. Data voidaan koodata eri tavoin. Kevyempi koodaus helpottaa siirrettävyyttä. Tiukempi parantaa tiedonsiirron tehokkuutta. Tiedonsiirto on turvallista. Asiakkaan ja palvelimen identiteetti varmistetaan. Myös verkkohyökkäyksiä ehkäistään. [9, s. 8.]

Asiakkaat voivat tehdä pyyntöjä palvelimille ja saada palvelimien vastaukset. Asiakkaat voivat myös tehdä tilauksia palvelimille. Palvelin ilmoittaa tapahtumista, kuten hälytyksistä, data-arvon muutoksesta tai ohjelman suorituksen tuloksesta ilmoituksilla. [9, s.12.]

Solmut ovat OPC UA:n osoiteavaruuden perusyksiköitä. Ne voivat olla täysin kytketyssä verkossa tai hierarkkisesti kytkettyinä tai jotain näiden väliltä. Reaalimaailman kohteet, esimerkiksi lämpötila-anturi, ovat mallinnettuina solmuina palvelimen osoiteavaruudessa. [9, s. 14.]

OPC UA:n palvelimet voivat olla tuotantotilan PLC:ssä tai yrityksen järeässä keskuspalvelimessa. Skaalautuvuus on laaja [9, s. 9].

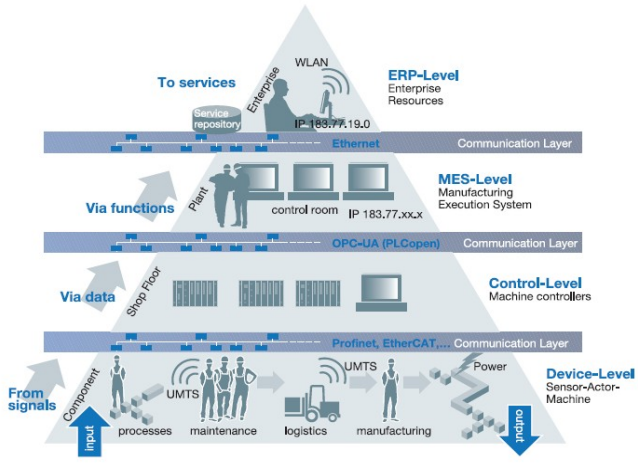
Istunnot

Istunto on looginen yhteys asiakkaan ja palvelimen välillä. Palvelin voi rajoittaa yhtäaikaisten istuntojen määrää resurssien riittävyyden takaamiseksi, käyttöoikeuksien vuoksi tai muilla perusteilla. [9, s. 12.]

Istunto on riippumaton tiedonsiirtoprotokollasta. Häiriöt tiedonsiirrossa eivät automaattisesti katkaise istuntoa. Istunnot päätetään asiakkaan tai palvelimen pyynnöstä tai asiakkaan toimettomuusajan perusteella. Aikaraja neuvotellaan istunnon käynnistyksessä. [9, s. 12.]

6.2 Tietoturvallisuus

Jo OPC:tä käytettiin mm. ERP- ja MES-järjestelmissä. Siksi oli selvää, että UA:n turvallissuutta kehitettäessä oli huomioitava myös yritysten informaatioteknologia [3, s. 201; 10, s. 7]. OPC UA:n turvallisuusarkkitehtuuri on yleinen ratkaisu, joka mahdollistaa eri tilanteisiin sopivan turvallisuustason käytön. Tietoturva kuluttaa resursseja, joten riittävä taso on päätettävä tapauskohtaisesti. Esimerkiksi kuvan 3 automaatiopyramidin mallissa pohjatasolla olevat toimilaitteet ja anturit yleensä eristetään palomuurilla eivätkä ne välttämättä vaadi suurempaa panostusta. Ylemmissä ERP- ja MES-kerroksissa tietokoneiden teho kasvaa, samoin uhat tietoverkkoyhteyksien vuoksi. [1, s. 47; 2, s. 201; 3, s. 203, 252; 10, s. 7.]



Kuva 3. Automaatiopyramidi [8, s. 67].

Asiakas ja palvelin voivat olla samassa koneessa, samassa talossa tai eri mantereilla. OPC UA suojaa yhteyden asiakkaan ja palvelimen välillä. Määritelmä ei ota kantaa ympäristöön, johon UA asennetaan. Sen turvataso on määriteltävä yrityskohtaisesti. OPC-säätiö suosittelee kuitenkin vahvasti, että jokaisessa kohteessa tietoturvallisuutta mietitään. [2, s. 204; 10, s. 11.]

Määritelmän kehityksen aikana konsultoitiin riippumattomia tietoturva-asiantuntijoita. Asiantuntijoiden suositukset huomioitiin ja määritelmä annettiin vielä asiantuntijoiden tarkistettavaksi. UA:ssa käytetään vakiintuneita turvallisuusstandardeja. [2, s. 204.]

Tekniset tiedot

Asiakkaan ja palvelimen yhteyttä muodostettaessa käytetään julkisen avaimen epäsymmetristä salausta, jota voidaan käyttää myös turvattoman yhteyden läpi. Kun asia-

kas ja palvelin ovat tunnistaneet toisensa, vaihdetaan symmetriseen salaukseen, koska se kuormittaa vähemmän kuin epäsymmetrinen salaus. Symmetriset avaimet vaihdetaan ajoittain uusiin, mikä parantaa tietoturvaa. Vasta kun turvallinen yhteys on muodostettu, aloitetaan istunto asiakkaan ja palvelimen välillä. [2, s. 212; 3, s. 213; 10, s. 15.]

Istunnon salaukseksi voidaan valita seuraavista: ei salausta, digitaalinen allekirjoitus tai allekirjoitus ja salaus. Istuntoa ilman salausta suositellaan käytettäväksi vain, jos ollaan varmasti turvallisessa ympäristössä tai käytössä on alemmilla tasoilla oleva suojaus. [2, s. 212; 3, s. 213]

Käyttäjä voidaan tunnistaa käyttäjätunnuksella ja salasanalla, X509-salausmenetelmällä tai verkkopalvelu WSS:llä [3, s. 220; 10, s. 22]. WSS (Web Service Security) on määritelmä verkkopalveluiden turvaamiseen.

OPC UA:n turvallisuusarkkitehtuuri on kerroksittainen. Kuljetuskerros, viestintäkerros sekä sovelluskerros käyttävät kukin omia turvallisuusmenetelmiään. [2, s. 207; 3, s. 211; 10, s. 12.]

OPC UA:han otetut alalla jo vakioituneet tietoturvamenetelmät ovat [2, s. 204; 10, s. 8–11]

– käyttäjän todennus (authentication)
– käyttöoikeudet (authorization)
– luottamuksellisuus (confidentiality)
– käytettävyys (availability)
– eheys (integrity)
– jäljitettävyys (auditability).

Lisäksi käytössä ovat aikaleimat, viestien järjestysnumerot, digitaaliset allekirjoitukset ja todistukset (certificates) [2, s. 207; 10, s. 21].

Turvallisuusuhkat täytyy tunnistaa, ennen kuin niitä vastaan voidaan suojautua. OPC UA:ssa on huomioitu [2, s. 204; 10, s. 9]

– viestitulva (message flooding)
– salakuuntelu (eavesdropping)
– viestien väärennys (message spoofing)
– väärinmuodostetut viestit (malformed messages)
– palvelimen profilointi (server profiling)
– istunnon kaappaus (session hijacking)
– vihamielinen palvelin (rogue server)
– kaapatut käyttäjätunnisteet (compromising user credentials).

Tietoturvasta on soveltuvin osin lisää tietoa myös määritelmien osissa: 4 palvelut, 6 datan liittäminen sekä 7 profiilit. [10, s. 12.]

Istunnon salaustaso on ohjelman kehittäjän valittavissa. Muun osuuden teknisestä osiosta suorittavat OPC-säätiön UA-viestintäkerrospino (communication layer stack) tai OPC-säätiön/kaupallisten toimijoiden sovelluskehitysympäristöt, SDK:t, jotka sisältävät säätiön pinon. Riippumattomat tietoturva-asiantuntijat ovat tarkastaneet ja hyväksyneet pinon toiminnan tietoturvan osuudelta. [2, s. 204, 207.]

Tietoturva OPC-säätiön testausohjelmassa

OPC-säätiön testauksessa edellytetään ennen ohjelmien hyväksyntää, että ohjelmien tietoturva-asetukset ovat kunnossa. Oletustilana turvallisuusasetusten tulee olla päälle kytkettyinä, mutta käyttäjällä pitää myös olla mahdollisuus sammuttaa palvelut, joita ei tarvitse. [2, s. 205.]

6.3 Osoiteavaruus

OPC UA:n osoiteavaruuden päätehtävänä on tarjota standardoitu malli palvelimille esittää asiakkaille objektit, muuttujat ja metodit. Malli kuvaa myös, miten objektit voivat olla suhteissa toisiinsa [11, s. 6]. OPC UA:ssa on yksi yhteinen osoiteavaruus kaikille palvelimelle saatavilla oleville yksiköille [2, s. 105]. OPC UA:ta käytetään monilla aloilla ja monenlaisissa ympäristöissä, joten osoiteavaruus on skaalautuva [1, s. 38].

Solmut ja viittaukset

Solmut kuvaavat palvelimelle kytkeytyviä yksiköitä, kuten objekteja, muuttujia ja metodeja. Solmut liittyvät toisiinsa viittauksilla. Viittaus kytkee aina kaksi eri solmua yhteen, toinen on lähde (SourceNode) ja toinen kohde (TargetNode) [2, s. 105; 11, s. 7]. Viittaus voi myös osoittaa toisen palvelimen osoiteavaruuden solmuun [3, s. 23]. Solmu kuuluu johonkin seuraavista määritelmän luokista [2, s. 105; 11, s. 18–34]

– objekti (Object)
– muuttuja (Variable)
– metodi (Method)
– näkymä (View)
– datatyyppi (DataType)
– muuttujatyyppi (VariableType)
– objektityyppi (ObjectType)
– viittaustyyppi (ReferenceType).

Solmuluokkia ei voi lisätä [2, s. 105]. Tyyppiluokkia voi laajentaa [3, s. 10]. Jokaisella luokkatyypillä on attribuutteja, joista osa on pakollisia. Kaikille solmuluokille yhteiset attribuutit ovat [2, s. 106]

– NodeID; pakollinen, solmutunniste, joka identifioi solmun osoiteavaruudessa
– NodeClass; pakollinen, luokka, johon solmu kuuluu
– BrowseName; pakollinen, ihmisystävällinen nimi solmulle
– DisplayName; pakollinen, paikallistettava (kieliversio) ihmisystävällinen nimi
– Description; vapaaehtoinen, paikallistettava kuvaus solmun tarkoituksesta.

Määritelmässä on lisää luokkakohtaisia attribuutteja. Määritelmässä on kerrottu myös, millä solmutyypeistä riippuvilla viittauksilla solmut voidaan yhdistää. [2, s. 106.]

Objektit, muuttujat ja metodit ovat olio-ohjelmoinnin käsitteitä.

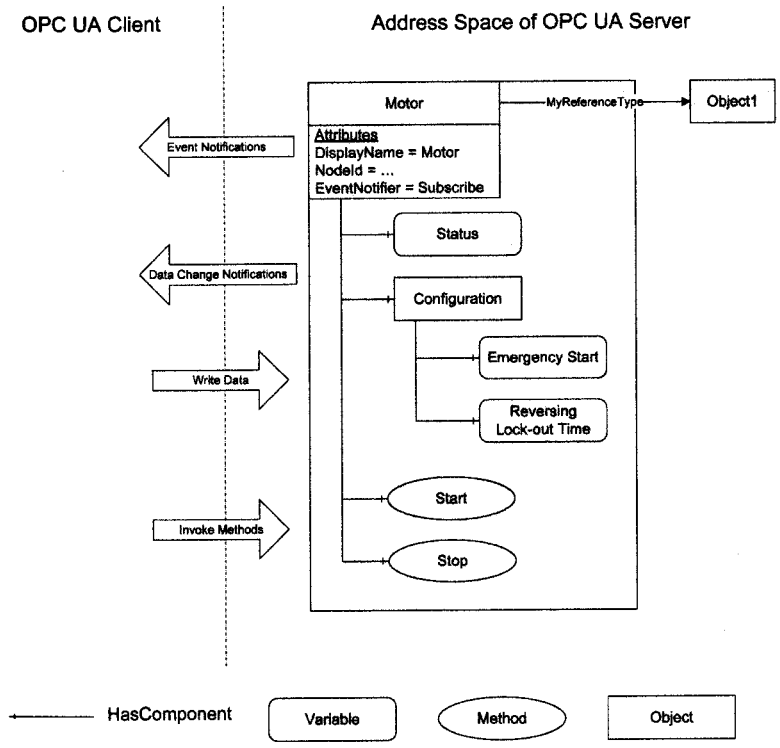
Standardisolmut

Osoiteavaruudessa on oltava tietyt standardisolmut, joilla on hyvin tunnetut solmutunnisteet (well-known NodeIds). Standardisolmut organisoivat osoiteavaruuden. Osoiteavaruus alkaa juurisolmusta (Node Root), joka on kansiotyypin (FolderType) objekti ja sisältää kolme kansiota [2, s 110]

– objektit (Objects), sisältää kaikki palvelimen objektit ja muuttujat
– näkymät (Views), kaikki palvelimen näkymät
– tyypit (Types), kaikki palvelimen tyypit: objektityypit, muuttujatyypit, datatyypit, viittaustyypit ja tapahtumatyypit (Object, Variable, Data, Reference, Event Types).

Objektit

Objekteja ovat esimerkiksi todellisen maailman asiat, esimerkkinä mittausanturi tai kokonainen tehdas [1, s. 43]. Objekti voi myös olla tietokoneohjelman olio, ohjausjärjestelmän laite ym. Objekti voi myös sisältää muita objekteja ja muuttujia (kuva 4). Yleisten attribuuttien lisäksi objekteilla on attribuutti, jolla määritellään, tuottaako objekti tapahtumia (Event Notifications), joihin asiakkaat voivat kirjautua. Kirjautuneet asiakkaat saavat ilmoitukset tapahtumista. Objekteilla voi käsitellä myös historiallisia tapahtumia (Historical Events). [2, s. 30; 3, s. 107.]



Kuva 4. Yleisesitys objekteista, muuttujista ja metodeista. Motor-objektissa on muuttuja Status, johon asiakkaat voivat tehdä tilauksen ja saada ilmoitukset Statuksen muutoksesta. Metodit ovat Start ja Stop. Configuration on objekti, jossa muuttujat Emergency Start ja Reversing Lock-out Time. [3, s. 31.]

Muuttujat

Muuttujilla esitetään arvoja. Muuttuja on tyypiltään dataa (DataVariable) tai ominaisuus (Property). Datana esitetään mittausarvo, esimerkiksi lämpötila, ominaisuutena esimerkiksi lämpötila-anturin mittausala. Kahdella solmulla esitetään anturin mittausala sekä lämpötila. Muuttujalla on myös attribuutit, joilla säädellään luku- ja kirjoitusoikeuksia sekä käyttäjän oikeus muuttujaan. [2. s. 107; 11, s. 25.]

Metodit

Metodit ovat asiakkaan kutsuttavissa olevia funktioita. Metodeilla voi olla ominaisuuksia (Properties), joilla kuvataan metodin tulo- ja lähtöargumentit. Yleisten attribuuttien lisäksi metodeilla on kaksi argumenttia, joilla metodi voi olla suoritettava (Executable) ja käyttäjän suoritettavissa (UserExecutable). Käyttäjän suoritettavissa -argumentilla määritellään, onko kirjautuneella käyttäjällä oikeus suorittaa metodi. [2, s. 107; 3, s. 30, 34.]

Näkymät

Esimerkiksi isossa prosessilaitoksessa yhden palvelimen osoiteavaruudessa saattaa olla jopa tuhansia solmuja ja viittauksia. Näkymällä voidaan rajata vain palvelimen asiakkaan kannalta relevantit solmut ja viittaukset. Näkymä on siis osoiteavaruuden osajoukko. [2, s. 108; 3, s. 71.]

Objekti- ja muuttujatyypit

OPC UA:ssa on mahdollista tyypittää objekteja ja muuttujia, esimerkiksi voitaisiin määritellä lämpötila-anturin tyyppi, jonka pohjalta voisi periyttää uusia antureita, olio-ohjelmoinnin termein uusia instansseja. Tyypissä voisi olla mukana vaikka graafinen kuva lämpötila-anturista, tai muita haluttuja ominaisuuksia. [2, s. 108; 3, s. 30.]

Viittaustyyppi

Viittaus on yhteys kahden solmun välillä. Viittauksiin ei voi suoraan liittää attribuutteja tai ominaisuuksia. Jokaisella viittauksella kuitenkin on tyyppinsä mukainen semanttinen tieto, miten solmut liitetään. Määritelmässä on muutamia valmiita viittaustyyppejä, esimerkiksi hierarkkiselle/ei-hierarkkiselle viittauksille. Viittaustyypin argumenteilla määritellään yleisten attribuuttien lisäksi, onko viittaus symmetrinen vai epäsymmetrinen tai

abstrakti eli tyyppi, josta ei suoraan voi johtaa uusia ilmentymiä. Viittaustyypin käsite on laajennettavissa eli palvelin voi määritellä oman viittaustyyppinsä. [2, s. 109; 3, s. 25.]

6.4. Palvelut

Palvelut ovat abstrakteja, kuten kaikki määritelmät lukuun ottamatta määritelmää 6 datan liittäminen. Palveluilla hallitaan kaikkea asiakkaan ja palvelimen välistä viestintää. OPC UA:ssa on 37 palvelua: 16 informaation vaihtoon ja 21 infrastruktuurin hallintaan. [2, s. 131; 3, s. 130; 8, s. 21.]

Vanhassa OPC:ssä laskien yhteen DA:n, HDA:n sekä A&E:n, metodeja oli yli 150. Syynä määrän pienenemiseen murto-osaan on UA:n yhteinen osoiteavaruus, vanhassa oli kaikille kolmelle esimerkiksi omat luku- ja kirjoitusmetodit. Myös osa COM:in mallin metodeista on UA:ssa korvautunut istuntopalveluiden vakioaloituksella, jota käytetään kaikissa pyynnöissä. Kuitenkin UA:n palveluilla katetaan sama toiminnallisuus kuin vanhan OPC:n metodeilla, ja uusiakin ominaisuuksia UA:n palveluilla saatiin toteutettua. [2, s. 140; 3, s. 125.]

Yleistä palveluista

Palveluissa käytetään pyyntö-vastausmallia, eli asiakas tekee pyynnön, johon palvelin vastaa. Viestien vaihto sekä palvelujen pyyntö ovat asynkronisia. Useat UA-pinon sisältävät ohjelmointirajapinnat (API, Application Programming Interface) kuitenkin tarjoavat myös synkroniset versiot. [3, s. 126.]

Toimintakatkot

Toimintakatkojen hallinta on tärkeää, kun tietoa siirretään tietoverkon yli. Katkot on syytä huomata ja reagoida niihin hallitusti. Siksi aloitusparametreissa on mukana asiakkaan antama aika, jonka kuluttua palvelin voi peruuttaa vielä vastaamattomat vastauksensa. [3, s. 127.]

Virheiden käsittely

Virheiden käsittely on myös tärkeää koska hajautetuissa tai laajoissa järjestelmissä virheitä voi tapahtua monella tasolla. Esimerkiksi asiakas voi käyttää vääriä parametreja, kuten väärää solmun tunnistetta, tai solmua, jota ei enää ole tai johon on yhteys poikki. Yhteysongelmia voi olla myös asiakkaan ja palvelimen välillä sekä palvelimella taustajärjestelmäänsä, esimerkiksi tehtaan kenttälaitteisiin. [3, s. 127.]

Palveluissa käytetään statuskoodia (StatusCode), joka voi olla hyvä (Good), epävarma (Uncertain) tai huono (Bad). Nämä ovat UA:ssa määriteltyjä, eikä niitä voi laajentaa. Toisena virheenilmoituskeinona käytetään diagnostiikkatietoa (DiagnosticInformation), jossa on lisätietoa statuskoodiin kuten toimittajakohtaisia virhekoodeja, paikallistettu virheen kuvaus sekä tekstikenttä lisätiedolle. [3, s. 128.]

Palvelinten etsiminen

Etsintäpalveluilla asiakkaat etsivät saatavilla olevia palvelimia ja niiden liityntärajapintoja (Endpoints). Etsintäpalvelut ovat [2, s. 132; 3, s. 132]

– palvelinten etsiminen (FindServers)
– hae liityntärajapinnat (GetEndpoints)
– palvelimen rekisteröiminen (RegisterServer).

Liityntärajapintojen kuvaus sisältää kaikki tarvittavat tiedot, jotka tarvitaan tietoturvalliseen yhteyteen ja istunnon muodostamiseen asiakkaan ja palvelimen välille.

Palvelin rekisteröidään etsintäpalvelimelle (Discovery server) palvelimen rekisteröinnillä. Palvelua käyttävät vain palvelimet. Etsintäpalvelin ylläpitää listaa saatavilla olevista palvelimista. Etsintäpalvelin voi olla itsenäinen palvelin tai itsenäinen sovellus, mutta jos sellaista ei ole järjestelmässä, kaikissa palvelimissa ovat ne etsintäpalvelimen toiminnot, jotka tarvitaan liityntärajapintojen esittelyyn. [2, s. 132; 3, s. 131.]

Tietoturvallinen yhteys

Asiakkaan ja palvelimen välille luodaan yhteys, joka varmistaa luottamuksellisuuden sekä välitettävien viestien eheyden. Suurin osa tästä toiminnallisuudesta on UA-pinossa (ks. 6.6), joten ryhmässä on vain kaksi palvelua: avaus (OpenSecureChannel) ja sulkeminen (CloseSecureChannel). [2, s. 132; 3, s. 135.]

Istunto

Myös istuntopalvelut liittyvät turvallisen ja luotettavan yhteyden luomiseen, eikä niitä käytetä tiedonsiirtoon. Asiakas todentaa kirjautuneen käyttäjän, joka voi hallita istuntoja. Istuntopalvelut ovat [2, s. 133; 3, s. 136–138]

– istunnon luonti (CreateSession)
– istunnon aktivointi *(*ActivateSession)
– istunnon sulkeminen (CloseSession)
– peruutus (Cancel).

Istunto hyväksytään vasta, kun aktivointi on onnistunut. Jos aktivointi ei onnistu määräajassa, esimerkiksi verkkovirheen vuoksi, palvelin päättää istunnon ja vapauttaa resurssit. [3, s. 136.]

Istunnon aktivointia voidaan käyttää myös käyttäjän vaihtamiseen, esimerkiksi vuorotyössä operaattori vaihdetaan työvuoron vaihtuessa, mutta istuntoa ei tarvitse katkaista. [2, s. 133; 3, s. 137.]

Solmujen ja viittaustenhallinta

Palveluita, joilla asiakas voi lisätä ja poistaa osoiteavaruuden solmuja ja viittauksia solmujen väliltä. Turvallisuusasetukset ja käyttäjän oikeudet voivat rajata käyttöä. Palvelut solmujen ja viittausten hallintaan ovat [2, s. 134; 3, s. 187–189]

– solmujen lisäys (AddNodes)
– solmujen poisto (DeleteNodes)
– viittausten lisäys (AddReferences)
– viittausten poisto (DeleteRefences).

Selaus ja kyselyt

Osoiteavaruudesta haetaan tietoa selaamalla ja kyselyillä. Selauspalvelut ovat seuraavat [2, s. 134; 3, s. 186]

– selaa (Browse)
– selaa seuraava (BrowseNext)
– solmujen rekisteröinti (RegisterNodes)
– rekisteröinnin peruutus (UnregisterNodes)
– selauspolun muunto solmutunnisteiksi (TranslateBrowsePathsToNodeIds).

Selaamalla voi edetä osoiteavaruudessa ilman tietoa sen sisällöstä. Selaa seuraava -palvelua käytetään jatkamaan selausta eli näyttämään jäljellä olevat solmut, mikäli niitä vielä on jäljellä. Solmujen rekisteröinnillä palvelin optimoi objektin pääsyn osoiteavaruuden kohteeseensa. Usein objektin rekisteröinti nopeuttaa sen suorittamista. Rekisteröinti on hyvä peruuttaa, kun sitä ei enää tarvita. Näin vapautetaan palvelimen resursseja. [2, s. 135; 3, s. 140–145.]

Selauspolun muunnolla solmutunnisteiksi päästään objektin komponentteihin objektityypin tietojen perusteella. Solmutunnistetta tarvitaan kun kirjaudutaan muuttujaan seuraaman sen arvon muutoksia tai kun kutsutaan objektin metodeja. [3, s. 142.]

Selaaminen on kankea tapa etsiä tietoja, jos osoiteavaruus on suuri tai dynaaminen. Kyselyllä voi tehdä hakuja palvelimen solmuista. Kyselyiden suodattimet voivat olla mutkikkaitakin. Kyselyiden palvelut ovat: kyselen aloitus (QueryFirst) ja jäljellä olevat (QueryNext), mikäli eivät mahtuneet ensimmäiseen. [3, s. 186; 12, s. 52.]

Kyselyjen suodattamista kehitettäessä mietittiin useita mahdollisia skenaarioita. Esimerkiksi kyselyt osoiteavaruudesta, joka on tallennettu laitteen muistiin tai relaatiotietokantaan. Muistikyselyt tulee suorittaa peräkkäisinä järjestyksessä. Kyselyt relaatiotietokannasta täytyy muuntaa SQL-komennoiksi. Molemmat ovat mahdollisia kyselyillä. Pienen osoiteavaruuden järjestelmissä kyselyistä ei juurikaan ole hyötyä, mutta jos osoiteavaruus on suuri, palvelimen suositellaan tukevan kyselyitä. [2, s. 135–136.]

Attribuutit

Attribuuttipalveluilla voi lukea ja kirjoittaa solmujen attribuutteja, myös niiden historiallisia arvoja. UA:ssa muuttujan arvo on mallinnettuna attribuuttina, siksi asiakas voi näillä palveluilla lukea ja kirjoittaa muuttujan arvon. Yksi objekti voi esittää nykyiset arvot sekä historian tiedoista ja tapahtumista. Palvelut ovat [2, s. 136; 3, s. 155–157]

– lue (Read)
– lue historiasta (HistoryRead)
– kirjoita (Write)
– päivitä historia (HistoryUpdate).

Luku ja kirjoitus ovat UA:ssa yleisiä palveluita, eli niillä voi lukea muuttujien arvoja, ominaisuuksia, metadataa ym. Kuten UA:n palvelut yleisesti, luku- ja kirjoituspalvelutkin ovat optimoituja massalukuun tai -kirjoitukseen. Yksittäisiä arvojakin pystyy lukemaan

ja kirjoittamaan. Toistuvan syklisen datan lukuun sopivat paremmin tilaukset (ks. 6.4). [3, s. 155.]

Historian päivityksellä voi muokata historiatietokantaa lisäämällä, vaihtamalla tai päivittämällä tietoa [3, s. 183].

Metodit

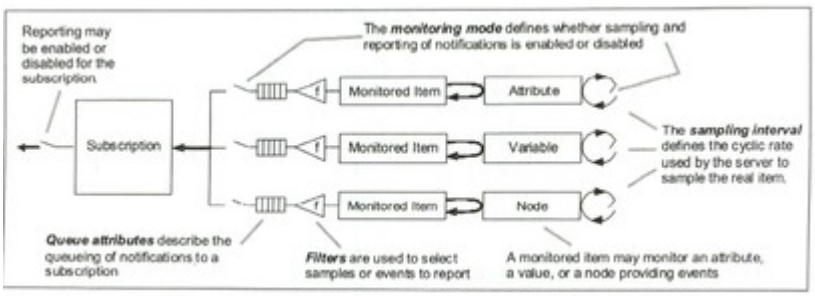
Metodit ovat objektin komponentteja. Asiakas voi kutsua niitä palvelulla kutsu (Call). Metodeilla voi olla tuloparametreja ja ne voivat palauttaa lähtöparametreja [2, s. 137; 3, s. 176]. Metodit sopivat parhaiten yksinkertaisille nopeasti suoritettaville funktioille, laajemmat kokonaisuudet tulisi tehdä ohjelmina [3, s. 30]. Ohjelmat ovat määriteltyinä UA:n spesifikaatiossa 10.

Tarkkailupalvelut (Monitored Items)

Tarkkailupalveluilla seurataan osoiteavaruuden solmuja. Tarkkailuja voidaan luoda, muokata ja poistaa. Tarkkailtavat kohteet huomioivat muutokset attribuuttien arvojen muutoksissa sekä objektien tapahtumat. Ilmoitukset välitetään jonon kautta tilauksen tehneille asiakkaille (kuva 5). Käytettävissä ovat seuraavat palvelut [2, s. 138; 3, s. 158–166]

– luo tarkkailu (CreateMonitoredItems)
– muokkaa tarkkailua (ModifyMonitoredItems)
– aseta tarkkailutila (SetMonitoringMode)
– laukaisu (SetTriggering)
– poista tarkkailu (DeleteMonitoredItems).

Muokkauksella muutetaan näytteenottotaajuutta millisekunteina, jonon pituutta ja näytteiden suodatusta [2, s. 138; 3, s. 167].



Kuva 5. Kaavioesitys tarkkailtavista kohteista [2, s. 145].

Asetuksella muutetaan tarkkailun tila estetyksi, sallituksi (sampling) tai ilmoittavaksi (reporting). Sallittu ottaa näytteet, ilmoittava ilmoittaa näytteenotot asiakkaille. [3, s. 167, 170.]

Laukaisulla määritellään tarkkailtava kohde, joka ilmoittaa arvonsa vain silloin kun toinen kohde laukaisee eli toteuttaa jonkun ehdon. Laukaisijan ja laukaistavan kohteen välille luodaan linkki. [3, s. 170.]

Asiakkaat tekevät usein tilauksia ja kohteiden tarkkailua, joten palvelimien suositellaan tukevan näitä palveluita. Kaikkein pienemmissä sulautetuissa järjestelmissä ei niitä välttämättä kuitenkaan ole. [2, s. 138.]

Tilaukset

Asiakkaat voivat tehdä tilauksia tarkkailtavista kohteista, jotka lähettävät ilmoituksia tilaaville asiakkaille. Tilausten palvelut ovat [2, s. 139; 3, s. 158]

– luo tilaus (CreateSubscription)
– muokkaa tilausta (ModifySubscription)
– aseta tilauksen tila (SetPublishingMode)
– julkaisupyyntö (Publish)
– uudelleenjulkaisupyyntö (Republish)
– muuta tilausta (TransferSubscription)
– poista tilaus (DeleteSubscription).

Julkaisupyynnöllä asiakkaat pyytävät palvelimelta päivitystä. Palvelu on asynkroninen, eli palvelimella ei ole velvollisuutta välittömästi vastata pyyntöön [3, s. 160]. Uudelleenjulkaisupyynnöllä asiakas voi pyytää välistä jääneet ilmoitukset uudelleen. Jokaisessa ilmoituksessa on järjestysnumero. Näitä vertaamalla asiakas voi päätellä saamatta jääneet ilmoitukset. [3, s. 162.]

Tilauksiin liittyy tarkistusväli. Jos mitään ilmoitettavaa ei ole, tarkistetaan yhteys määrätyin väliajoin lähettämällä tarkistusviesti (heartbeat connection). [2, s. 146; 3, s. 162.]

6.5 Informaatiomalli

OPC UA -palvelimen päätehtävänä on esittää tosiaikaista informaatiota esimerkiksi tehtaan prosesseista asiakkaille ja tarjota asiakkaille mahdollisuus hallita prosesseja [2, s. 111; 3, s. 21].

Informaatio on abstraktia, joten informaatiomallin täytyy kuvata syntaksi. OPC UA:n informaatiomalli muistuttaa ohjelmointikieliä, joilla tehdyt ohjelmat kuitenkin pysyvät kääntämisen jälkeen staattisina, toisin kuin OPC UA:ssa. [2, s. 112.]

Palvelimen esittämä informaatio voi olla monimutkaista, siksi asiakkaat saattavat tarvita kuvauksia ja tarkennuksia informaatiosta eli semanttista tietoa [3, s. 19]. Tietoa voidaan lukea vapaalla haulla (random access) tai selaamalla. Vapaa haku edellyttää että kohteella on tunnus, joka erottaa sen muusta ympäristön kohteista, ja että asiakkaat tuntevat kohteen ennestään. OPC UA:ssa näitä kutsutaan hyvin tunnetuiksi osoitteiksi (well-known addresses). Selaaminen on kuormittavaa, kuva rakenteesta muodostuu etenemällä askel kerrallaan. Etuna kuitenkin on, ettei rakenteesta tarvitse tietää etukäteen. Kuljettu polku voidaan muuntaa erotteleviksi osoitteiksi, solmutunnisteiksi, ja näin vähentää kuormitusta. Kaikki palvelimen saavutettavat kohteet muodostavat osoiteavaruuden (ks . 6.3). [2, s. 112–113.]

Osoiteavaruutta varten täytyy tehdä instanssit solmuista ja liittää niihin attribuutit. Solmut kytketään viittauksilla. Solmuluokka (NodeClass) on solmun muodollinen kuvaus, joka määrittelee sallitut attribuutit ja viittaukset. Tyyppi (Type) on solmun muodollinen kuvaus, ja sillä määritellään sallitut attribuuttien ja viittausten arvot. [2, s. 113.]

OPC UA:ssa on valmiiksi määriteltyjä tyyppejä, jotka ovat laajennettavissa [2, s. 108, 113; 3, s. 25, 36]

– objektityypit
– muuttujatyypit
– viittaustyypit.

Tyypit ovat metadataa, koska ne kuvaavat datan rakennetta, eivät itse datan arvoja. Datatyyppi (DataType) solmuluokka (ks. 6.3) on erikoistapaus, eli se kuvaa palvelimen dataa asiakkaille. Esimerkiksi datatyypin solmulla voi olla numeerinen arvo. Asiakkaat voivat datatyypin perusteella tulkita arvon oikein. Useimmat OPC UA:n datatyypeistä ovat tuttuja ohjelmointikielistä, kuten totuusarvo, kokonaisluku, liukuluku jne. Lisäksi mukana on UA:n omia kuten solmutunniste (NodeID), statuskoodi (StatusCode) ym. [2, s. 120; 3, s. 61; 13, s. 6.]

Vaikka UA:ssa on monia valmiita tyyppejä, voidaan tyyppejä määritellä tarpeen mukaisesti. Uusia tyyppejä johdetaan määritellyistä. Johdetut tyypit perivät vanhempiensa ominaisuudet, mutta näitä voidaan muokata. [2, s. 114.]

Tyyppeihin perustuva osoiteavaruus voi olla lähtökohtana mille tahansa vaadittavalle informaatiolle. Asiakkaat ymmärtävät osoiteavaruuden ja voivat navigoida siinä. Selauksen alkupisteenä käytetään standardisolmuja (ks. 6.3). [2, s. 114.]

Tilakone

Informaatiomallissa on mukana myös tilakone, jolla mallinnetaan objektia tiloina ja tilojen muutoksilla. Tilakoneet ovat objekteja, jotka sisältävät muita objekteja. Käytössä ovat objektityypit, muuttujatyypit sekä viittaustyypit. [2, s. 126; 3, s. 117.]

Tila on olotila, jossa objekti voi olla jonkin aikaa elinaikansa aikana. Muutos on siirtyminen tilasta toiseen. Muutoksen aiheuttaa määritelty tapahtuma. [2, s. 126; 3, s. 117.]

6.6 Datan liittäminen

OPC UA:n 6. spesifikaatio on ainoa, joka ei ole abstrakti ja kuvaa, miten abstraktit määritelmät kuten informaatiomalli, palvelut ja turvallisuusmalli liitetään olemassa oleviin teknologioihin [2, s. 97; 3, s. 191; 13, s. 4].

Datan koodaus, binäärimuoto ja XML

Koodauksella määritellään, miten data muutetaan siirrettävään muotoon. OPC UA:n tapauksessa koodataan palveluviestit sekä niiden tulot ja lähdöt sarjamuotoon välitettäväksi verkon kautta. UA:ssa on kaksi koodaustapaa: binäärimuoto ja XML. Yhteisiä ominaisuuksia molemmille koodauksille ovat yksinkertaisten datatyyppien (esim. kokonaisluku, tavu, liukuluku, totuusarvo) käyttö mutkikkaampiin rakenteisiin sekä laajentuvan objektin (ExtensionObject) käyttö. Laajentuva objekti on ”säiliö” mille tahansa datalle, joka datan lisäksi sisältää tunnisteen, mitä dataa sisältää ja miten se on koodattu, esimerkkinä video. [2, s. 218; 3, s. 192; 13, s. 7.]

Monissa teollisuusautomaation kohteissa suorituskyky ja nopea tiedonsiirto ovat tärkeitä parametreja. Siksi OPC UA:han kehitettiin oma binäärimuoto, joka on nopea koodata ja purkaa ja välittyy nopeasti verkon kautta. Binäärimuodon toteutus on myös ohjelmakoodiltaan lyhyt ja sopii siksi hyvin sulautettuihin järjestelmiin. [2, s. 218; 3, s. 192; 13, s. 8.]

Standardoitua XML:ä tukevat useimmat teollisuusautomaation sovellukset, lisäksi XML on ihmisten luettavissa. OPC UA:ta käytetään myös MES- ja ERP-tasoilla, siksi XML-tuki on tärkeässä osassa. [2, s. 218; 3, s. 193.]

Turvallisuusprotokollat

OPC UA käyttää kahta turvallisuusprotokollaa abstraktien palveluiden toteuttamiseen: verkkopalveluiden turvattua yhteyttä (WS-SecureConversation) ja UA:n omaa turvattua yhteyttä (UA-SecureConversation). Verkkopalveluiden turvattu yhteys on laajennos verkkopalveluiden turvallisuusstandardiin (WS-Security, WSS), jossa määritellään käsitteet datan turvaamiseen verkkopalveluilla. XML:n salaukseen ja digitaaliseen allekirjoitukseen käytetään XML-standardeja. [2, s. 218; 3, s. 194.]

Siirtoprotokollat

OPC UA:n kaksi siirtoprotokollaa, HTTP/SOAP ja TCP/IP, määrittelevät, miten sarjamuotoon saatetut viestit välitetään OPC UA -sovellusten välillä [2, s. 219; 3, s. 198]. Tässäkin näkyy säätiön pyrkimys käyttää vakiintuneita standardeja.

Siirtoprotokollia käytetään palvelimen ja asiakkaan välisessä yhteydessä verkkotasolla. Erot niiden välillä ovat suorituskyky sekä ohjelmakoodin koko. HTTP/SOAP on hitaampi ja koodiltaan pitempi. Etuina ovat yksinkertaisuus ja palomuuriystävällisyys: palomuurista ei tarvitse avata uusia portteja, koska se käyttää samaa porttia kuin verkkoselaimet. TCP/IP on verkon perusprotokollana alemmalla tasolla. OPC UA:n TCP/IP-liikenne vaatii oman avoimen portin (4840) palomuurissa. [2, s. 219; 3, s. 198; 7.]

OPC UA -pino

OPC UA -palvelimia ja -asiakkaita käytetään monissa eri tarkoituksissa: ohjaamaan teollisuuden prosesseja, tiedon analysointiin jne. Yhteinen perusta kuitenkin kaikelle toiminnalle on turvatun yhteyden muodostus asiakkaan ja palvelimen välille sekä datan lukeminen ja kirjoitus. Tämä on OPC UA:ssa toteutettu protokollapinona. Protokollapino, eli ohjelmakerrostenpino, ei ole helppo ohjelmointitehtävä. Siksi OPC-säätiö tarjoaa jäsenilleen UA-pinon, joka on yleinen ja sopii moneen eri tarkoitukseen. Monet ohjelmistotoimittajat tekevät omat ohjelmansa UA:n pinon päälle. UA:n pino on toteutettu c# -(.NET), ANSI C- ja Java-versioina. [2, s. 296; 3, s. 256.]

6.7 Profiilit

Profiileilla määritellään UA-sovellusten tukemat toiminnot. OPC-säätiö testaa profiilit. Sovelluksen tulee pystyä tarjoamaan toiminnot, joita se ilmoittaa tukevansa profiilissa. Profiilien perustella ihmiset osaavat hankkia oikeanlaisia laitteita ja ohjelmia, myös sovellukset keskenään hyödyntävät profiileja. Jos vastapuoli ei profiilinsa mukaisesti tue jotain toisen osapuolen tarvitsemaa toimintoa, yhteyttä ei hyväksytä. [2, s. 223; 3, s. 299.]

OPC UA on skaalautuva, esimerkiksi sulautetuissa järjestelmissä täytyy miettiä, mitä rajalliseen tilaan mahtuu. Pois jätetään usein historiallinen data, usein myös tilaukset. Minimissään riittää kun UA-palvelimessa on turvallisuuden ja istunnonhallinnan profiilit. [2, s. 223.]

OPC-säätiön testaustoiminta perustuu profiileille. Hyväksyttyjen profiilien digitaalisten todistusten allekirjoittajana toimii OPC-säätiö. [1, s. 51; 3, s. 303.]

Profiilitilanne elää koko ajan. Voimassa olevan tilanteen voi tarkistaa www-sivulta: http://opcfoundation-onlineapplications.org/profilereporting/ [2, s. 228; 3, s. 302.]

Määritelmät

Profiili on muodollinen kuvaus osasta UA:n määritelmissä esitetyistä käyttömahdollisuuksista. Profiili muodostuu vaatimustenmukaisuusyksiköistä ja mahdollisesti muista profiileista. Vaatimustenmukaisuusyksikkö (conformance unit) on pieni erä UA:n toiminnoista, joka voidaan testata itsenäisenä yksikkönä. Sama vaatimustenmukaisuusyksikkö voi olla useammassa profiilissa. Samat ominaisuudet ovat vain yhdessä vaatimustenmukaisuusyksikössä. [1, s. 49; 2, s. 224; 3, s. 299; 14, s. 4.]

Täysin määriteltyjen profiilien (full-featured profile) lisäksi on profiilityyppi, hankalasti suomennettava *facet*, jossa on tietty perustoiminta (vaatimustenmukaisuusyksiköt), mutta joka ei toimi yksinään. Tyypin profiili liitetään täysin määriteltyyn profiiliin tuomaan lisäominaisuuksia. Oma nimi profiilityypillä on siksi, että se erottuu itsenäisesti toimivista, täysin määritellyistä profiileista. [1, s. 50; 2, s. 224; 3, s. 300; 14, s. 2.]

Profiilit ovat jaoteltuina palvelin-, asiakas-, siirto- ja turvallisuusprofiileihin. [1; s. 50; 2, s. 226; 3, s. 300.]

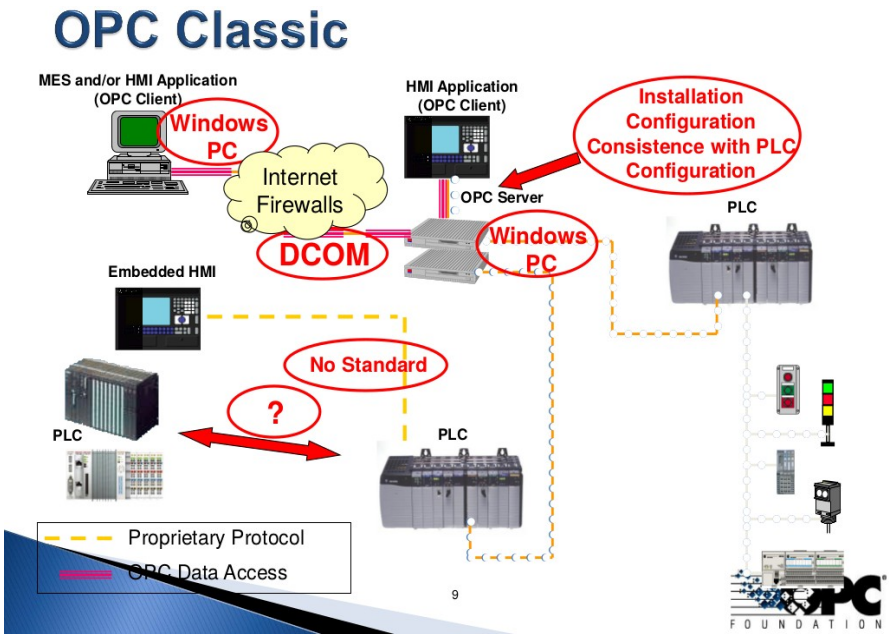
7 OPC ja OPC UA teollisuudessa

OPC kehitettiin ratkaisemaan automaatioteollisuuden ”ajuriongelma”, eli tarve tuottaa eri automaatiojärjestelmille, -laitteille ja versioille kullekin omat ajurinsa (ks. 2). Tavoite saavutettiin abstrahoimalla, eli sovellukset lukevat tietoa laitteilta tietämättä, miten laitteet toimivat [1, s. 6]. Suosiosta tuli huima. Aluksi OPC:tä käytettiin HMI-järjestelmissä, esimerkiksi prosessien valvomoissa ja SCADA- ja DCS-järjestelmissä ohjaamaan teollisia prosesseja. Pian käyttökohteiksi tulivat myös ERP- ja MES-järjestelmät (ks. 4).

OPC ja OPC UA ovat molemmat olleet markkinoilla jo vuosia, kummatkin ovat teollisuusstandardeja ja molemmille on saatavilla monia kaupallisia HMI-, ERP- ja MES-ohjelmistoja.

7.1 OPC

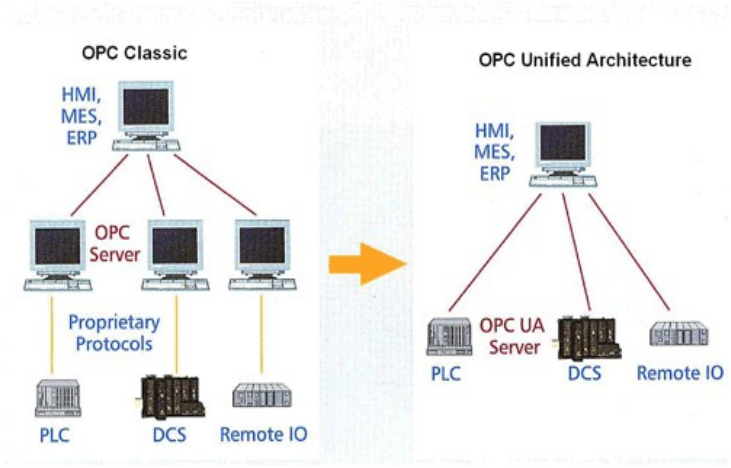
OPC käyttää Microsoftin DCOM:a tiedonsiirtoon, siksi se sitoo käytön Windows-käyttöjärjestelmään (kuva 6). DCOM tuli Windows CE:n 3.0 -versioon vuonna 2000 [15], näin monet sulautetut järjestelmät, esimerkiksi PLC:t, voitiin tehdä OPC-palvelimen sisältäviksi. DCOM itse on staattinen, melko mutkikas rakenteeltaan ja vaatii paljon muistia [2, s. 15].



Kuva 6. OPC:n kuvaus [16, s. 9].

Vaikka Microsoft ilmoitti jo vuonna 2002 luopuvansa DCOM:n kehittämisestä, on se vielä mukana myös Windows 10:ssä, tosin rekisteriavaimella on valittavissa, onko se aktiivinen [17]. Julkaistut Windows-versiot eivät siis ainakaan vielä rajoita OPC:n käyttöä. Tulee kuitenkin huomioida DCOM:n rajoitteet (ks. 4.1).

7.2 OPC UA



Kuva 7. OPC:n ja OPC UA:n vertailu [18].

OPC UA toteuttaa OPC:n toiminnallisuuden ja on myös alaspäin yhteensopiva OPC:n kanssa. Olemassa olevaa OPC-toteutusta ei tarvitse muuttaa kerralla jos siirrytään UA:han. Päivitys voidaan tehdä suunnitelmallisesti ilman suuria taloudellisia kertaponnistuksia. Etuna teollisuusnäkökulmasta OPC UA sallii myös suuremman joustavuuden. UA ei rajoita käyttöjärjestelmää, ohjelmoinnin voi tehdä lukuisilla kielellä. Samaan sovellukseen voi toteuttaa asiakkaan ja palvelimen (kuva 7). Skaalautuvuudesta esimerkkinä kuvan 8 Matrikonin piirilevylle toteutettu OPC UA -palvelin.

Kuva 8. OPC UA -palvelin piirilevyllä, 240 kB flash-muistia ja 35 kB RAM-muistia [8, s. 58].

Yhteinen osoiteavaruus helpottaa käyttöä verrattuna OPC:n erillisiin osoiteavaruuksiin. Ohjelmoinnissa tyypit (ks. 6.3) ovat suuri etu, samoin tietysti muut olio-ohjelmoinnin ominaisuudet. Uusia ominaisuuksia olivat myös metodit, kyselyt, solmujen hallinta ja tapahtumien historia, tietoturvallisuus (ks. 6.2), tilaukset (ks. 6.4) ja redundanssi. [2, s. 142.]

Tilaukset perustuvat tapahtumiin, siis ilmoitetaan vain jos on jotain ilmoitettavaa. OPC ”pollaa”, eli jokaiselta anturilta kysytään vuorollaan sen tila. Jos kysely tehdään liian harvoin, voidaan menettää dataa; jos taas liian usein, verkon kuormitus kasvaa. [19.]

Redundanssi on tärkeä asia monilla teollisuuden aloilla, esimerkiksi prosessiteollisuudessa. Tiedon täytyy siirtyä jatkuvasti ilman katkoja. Redundanssi voidaan toteuttaa kahdella tai useammalla palvelimella tai asiakkaalla. [2, s. 142, 196.]

Useat OPC UA:n kumppanuusstandardit (ks. 9) helpottavat UA:n soveltamista monilla teollisuuden aloilla.

8 IEC-standardi 62541

IEC:n työryhmä sekä monet yritykset pyysivät OPC-säätiöltä UA:n määritelmiä julkaistavaksi IEC-standardina. Vuonna 2007 päätettiin viedä OPC UA:n määritelmät IEC:n standardointiprosessiin. OPC UA:sta tuli IEC-standardi 62541. [2, s. 22.]

OPC-säätiön kannalta standardointi nostaa määritelmien näkyvyyttä ja edistää niiden leviämistä. Useat käyttäjät ja valmistajat pitävät IEC:n standardeja yhdenmukaisempina ja yleisesti hyväksytympinä kuin vain säätiön julkaisemia. Lisäksi IEC-prosessi varmistaa jatkuvuuden. Käyttäjät pitävät tätä etuna. [2, s. 24; 20.]

OPC UA:n perusteella on julkaistu muutama muukin IEC-standardi, jotka viittaavat OPC UA:han. Viittauksia pidetään vahvempina jos kohteena on IEC-standardi OPC-säätiön julkaisun tilalla. [2, s. 24.]

Standardinmukaisuudella on merkitystä julkisissa hankinnoissa ja lainsäädännössä [2, s. 24]. Etenkin Euroopan ja Aasian markkinoilla arvostetaan standardeja [20].

Nykytilanne

Viimeisin päivitys työtä kirjoitettaessa oli tehty 22.4.2015. Päivitetty ja laajennettu IEC-standardi 62541 sisälsi päivitetyt versiot vuonna 2011 julkaistuista määritelmistä 03–10. Uusina mukaan oli otettu osat 11 ja 13, kokonaan uutena 100, laitteiden yhdistäminen (Device Integration). [21.]

9 Yhteistyö eri organisaatioiden kanssa

OPC UA:ta käytetään teollisuusautomaation lisäksi mm. rakennusautomaatiossa, turvallisuuspalveluissa, autoteollisuudessa, uusiutuvassa energiantuotannossa ja älykkäissä sähköverkoissa. Informaation integroituminen tehostaa yritysten toimintaa. OPC-säätiötä on pyydetty monen eri organisaation yhteistyökumppaniksi. [2, s. 230, 236; 22.]

Kaasun- ja öljyntuottajilla on jo ennestään lukuisia yhteistyöorganisaatioita. OPC-säätiö tekee yhteistyötä näistä useiden kanssa. [22.]

9.1 Kenttälaitteet

Kenttälaitteiden toimittajat FDT Group, Fieldbus Foundation, HART Communication Foundation, PROFIBUS & PROFINET International ovat liittyneet yhteen OPC-säätiön kanssa valmistelemaan määritelmää kenttälaitteiden integroinnista. Useimmat ryhmän toimittajista olivat jo ennestään OPC-yhteensopivien laitteiden valmistajia, mikä helpottaa kehitystyötä. Työtä tehdessä kenttälaitteiden standardi on julkaisuehdotuksena. [2, s. 231; 22.]

9.2 Rakennusautomaatio

Rakennusautomaatiossa on laajasti käytössä BACnet, standardi ISO 16484-5. BACnetin Euroopan eturyhmä (The BACnet Interest Group Europe, BIG-EU) ja OPC-säätiö sopivat vuonna 2012 yhteistyöstä. [22; 23.]

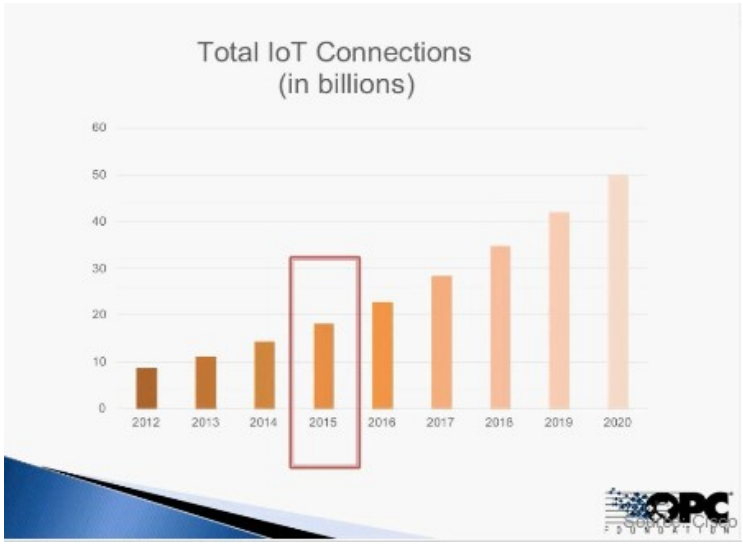
9.3 Valmiit kumppanuusstandardit

Työtä tehdessä olivat valmistuneet yhteystyöstandardit prosessien ja laboratorioiden analysaattorilaitteille (ADI) [2, s. 236], laajasti käytössä olevalle ohjelmoitavien logiikoiden PLCopenin standardille IEC 61131-3 [2, s. 248], automaatiojärjestelmien laitteille (DI) sekä valmistuksenohjausstandardille ISA-95:lle. [24; 25.]

10 IoT

Elektroniikan ja tietotekniikan kehittyessä laitteiden koko ja virrankulutus ovat pienentyneet samalla kun niiden suorituskyky on kasvanut. Anturitekniikan edistyksen myötä mittausdataa saadaan lähes mistä tahansa mitattavasta kohteesta. Kun älykkäät anturit vielä saadaan verkotettua langallisesti tai langattomasti, puhutaan esineiden internetissä (IoT, Internet of Things). [26.]

IoT on suosittu aihe politiikassa ja teknologiassa, sen arvioidaan kasvattavan maailman bruttokansantuotetta 25 % 10 vuodessa. Kuvassa 9 arvioidaan IoT-liittymien kasvua. Suomessa alan arvioidaan synnyttävän jopa 50 000 työpaikkaa. Esineiden internet on myös yksi Sipilän hallituksen kärkihankkeista. [27].



Kuva 9. IoT-liittymät, toteutuma ja ennuste miljardeina [14].

Maailmanlaajuisesti IoT-markkinoiden arvo vuonna 2013 oli 1,9 miljardia US dollaria. Arvio vuodelle 2020 on 7 miljardia US dollaria. [20.]

10.1 IIoT, teollinen asioiden internet

Teollisuudessa on myös nähty IoT:n edut. IIoT:n eli teolliseen asioiden internetiin liittyvät läheisesti käsitteet koneiden välisestä tietojen vaihdosta (M2M), pilvipalveluista, sekä isoista datamääristä (Big Data), eli suuresta määrästä ja nopeasti kertyvästä datasta, esim. mittausantureilta teollisuudessa. Tietovarannot ovat niin suuria ettei niitä

voi käsitellä yksittäisellä tietokoneella. Suuresta tietomäärästä voi kuitenkin löytyä mielenkiintoisia trendejä tai korrelaatioita oikeilla laskenta-algoritmeilla. [20; 28; 29, s. 5]

IIoT:n edellytyksiä

Anturit ja toimilaitteet tarvitsevat langallisen tai langattoman liitännän internetiin sekä ohjauselektroniikkaa eli ”älyä”. Ne voivat siis itsenäisesti tehdä analyyseja esim. kunnostaan, toimintatunneistaan jne. Tuotantoympäristö saadaan paremmin haltuun, kun koneet viestivät keskenään (M2M) ja saadaan parempi käsitys prosessista ja informaatio hajautettua kenttälaitteisiin. [20.]

Tiedon oikeellisuus täytyy turvata, samoin kontekstin eli tietojen ja metatiedon tulee pysyä yhdessä. Tiedon tulee olla luotettavaa. Riippumattomuus myyjätahosta on myös tavoiteltava asia. Pilvipalveluita ei kaikissa ratkaisussa tarvita. [20.]

10.2 IIoT:n kehitystä ohjaavat yhteisöt

Saksassa hallituksen aloitteesta aloitettiin vuonna 2011 teollisuus 4.0 (Industrie 4.0) -suositusten laatiminen tulevaisuuden tehtaille, jonka teknologiat ovat kyberfyysiset järjestelmät ja IoT. Suosituksia olivat laatimassa myös saksalaisten yritysten ja tutkimuslaitoksen edustajat. [8, s. 3; 30, s. 4.]

IIoT:n edistämiseksi perustettiin Yhdysvalloissa vuonna 2014 voittoa tavoittelematon avoin Industrial Internet Consortium, IIC [31; 32, s. 12].

10.3 IIoT:n hyödyt ja uhat

IIoT:n on katsottu tehostavan teollisuudessa ennakoivaa huoltoa ja parantavan tuotantoa pienentämällä tuotantokatkoja. Jotkut innovatiiviset yritykset ovat kuitenkin jo saaneet IIoT:stä uusia liiketoimintamalleja. [32, s. 4.].

Hyötyjen lisäksi tulee myös huomioida uhat, mitä internetiin liittymisestä voi seurata. Joitakin esimerkkejä teollisuudessa on jo tapahtunut: hakkerit onnistuivat kääntämään öljynporauslautan laituria ja haittaohjelma sai toisen lautan ohjausjärjestelmän niin sekaisin, että kesti viikkoja saada se järjestykseen [32, s. 13]. Tietoturvan siis tulee olla kunnossa.

Joillakin perinteisen teollisuuden aloilla on vaikea nähdä uusien digitaalisten palvelujen hyötyjä, näissä tapauksissa on monesti kannattavaa tehdä yhteistyötä nykyaikaisen liiketoiminnan hallitsevien yritysten kanssa [32, s. 9]. Suomessa IIoT:n yleistymisen esteeksi on muodostunut yritysten haluttomuus avata tuotantolaitteidensa tietoja toisille osapuolille [33, s. 16].

Kielteisenä voi nähdä myös nykyisen kehityksen, jossa useat isot tietotekniikkayritykset markkinoivat omia pilvipalveluitaan, jotka ovat keskenään yhteensopimattomia. Luodaan siis taas yrityskohtaisia tietovarantoja, joiden kesken ei voi vaihtaa tietoa. Esimerkiksi OPC UA on käyttöjärjestelmäriippumaton, se tukee monia ohjelmointikieliä ja on standardoitu. Kumppanuusorganisaatioidensa kanssa OPC UA saavuttaa yhä suuremman käyttäjäkunnan. [20.]

IIoT tulee myös viemään suorittavan portaan työpaikkoja. Uusia korkeamman osaamisen työpaikkoja IIoT tulee synnyttämään. [20; 27.]

10.4 OPC UA ja IIoT

Artikkeleissa IIoT:n haasteista ja vaatimuksista toistuvat usein avoimuus, skaalautuvuus ja turvallisuus [32, s. 13; 29]. Asioita jotka olivat keskeisiä tavoitteita OPC UA:ta suunnitellessa (ks. 5). Myös teollisuus 4.0:n suunnitteluvaiheessa monet vaatimukset olivat yhteneviä OPC UA:n tavoitteiden kanssa. Niinpä oli luonnollista, että teollisuus 4.0:n viestintäkerrokseksi suositellaan OPC UA:ta. [20; 34, s. 25.]

OPC UA:n yleistymisen ja yhteensopiviksi sertifioitujen laitteiden määrän kasvaessa säästetään jo näilläkin, koska yhdessä tietokoneessa voi yhtäaikaisesti olla käynnissä useampiakin asiakkaita ja palvelimia. Jos vielä toimilaitteissa on mukana UA-palvelin, tuotannonohjauksesta voidaan poistaa monta pc:tä/PLC:tä. [20.]

Useimmissa tuotantolaitoksissa on joku automaatioväylä. Lähes kaikkia voidaan ohjata OPC UA:lla ylemmistä kerroksista esimerkiksi MES:stä. Kumppanuusstandardien kanssa päästään yhteiseen osoiteavaruuteen jo kenttätasolla esimerkiksi PLCopen, joka on standardoitu IEC 61131-3:ssa, suosittu teollisuusautomaatiossa ja käytössä monissa ohjelmoitavissa logiikoissa. IoT:hen siirtyminen voidaan siis tehdä askelittain. [20.]

11 Harjoitustehtävä

Tavoitteena insinöörityössä oli myös malliesimerkki harjoitustyöstä, jossa oli tarkoitus tehdä OPC UA -asiakassovellus. Sovelluksen tuli liittyä palvelimeen, selata palvelimen osoiteavaruutta, lukea ja kirjoittaa muuttujia sekä kirjautua muuttujaan, jonka arvon muutoksia seurataan. Asiakkaalle tuli myös toteuttaa käyttöliittymä. Lisätehtävänä oli OPC UA:n yhdistäminen pilvipalveluun.

Harjoitustyö oli tarkoitus toteuttaa c#-kielellä Microsoftin Visual Studiolla ohjelmoiden. Tarvittavat kirjastot Windows-koneilla OPC UA -viestintään ovat

– Opc.Ua.Client.dl
– Opc.Ua.Core.dll
– Opc.Ua.Server.dll.

11.1 Harjoitustehtävän toteutus

Työ aloitettiin hyvin dokumentoidulla vapaalla pythonin OPC UA -kirjastolla, jolla toimiva asiakas-palvelinyhteys onnistuikin päivässä. Palvelinta voitiin hyödyntää myös jatkossa.

Kirjallisista lähteistä [1; 2; 3] saatiin hyvä yleiskuva OPC UA:sta, mutta ne eivät sisältäneet lainkaan ohjelmaesimerkkejä. OPC-säätiön www-sivuilla on paljon materiaalia, jopa esimerkkiohjelmia. Ohjelmien lähdekoodit ovat kuitenkin saatavilla vain yritysjäsenille. Viikkojenkaan hakukoneiden käyttämisen jälkeen ei löytynyt kuin yksi c#-kielinen konsolitoteutus (tekstimoodi-), jota soveltamalla saatiin muuttujan lukeminen, kirjoitus ja selaus toteutettua eri koneissa olevien palvelimen ja asiakkaan välillä. Kyselyitä OPC UA c#-mallista löytyi kyllä monia, mutta vastauksina annetut linkit olivat lähes poikkeuksetta toimimattomia. Toimivaa esimerkkiä kirjautumisesta c#-kielellä ei löytynyt, joten harjoitustyön malliesimerkkiä ei pystytty toteuttamaan loppuun asti.

Liitteessä 1 on harjoitustehtävän tehtävänanto, liitteessä 2 on esitetty vajaaksi jäänyt ohjelmakoodi.

11. 2 OPC UA:n sovelluskehitysympäristöt

Internethauissa usein esiintyivät kaupallisia OPC UA -kehitysympäristöjä tarjoavien yritysten malliesimerkit omille sovelluksilleen.

Useimmat yritykset eivät tarjonneet sovelluskehitysympäristöjensä hintatietoja sivuillaan. Muutamilta hinta löytyi, halvimmillaankin useampia tuhansia euroja. Ilmaiset ver-

siot löytyivät Matrikonilta ja Unified Automationilta, mutta lisenssiehdot olivat tiukat. Ne sopivat lähinnä opiskelukäyttöön.

11.3 Pohdinta

c#:lla onnistuivat palvelimeen liittyminen, yhteyden katkaisu, lukeminen, kirjoitus ja osoiteavaruuden selaus. Kirjautuminen muuttujan arvon seuraamiseen ei onnistunut. Syvällisemmällä c#-kielen tuntemuksella olisi saattanut saada niukoista esimerkeistä enemmän hyötyä. Nyt ne lähinnä kopioitiin sellaisenaan, muutamaa arvoa muuttamalla, eikä toimivaa asetusta löytynyt.

OPC UA -sovelluskehitysympäristöissä ei vaadita ohjelmointiosaamista. Kehitysympäristöissä toimintoja on yksinkertaistettu, esimerkiksi palvelimeen liittyminen toteutuu yhdellä komennolla, samoin yhteyden katkaisu, selaus jne. Mukana seuraa vielä hyvä dokumentaatio. Korkeahkot hinnat voivat olla käytön esteenä.

12 Yhteenveto

Lähdekirjoista ja määritelmistä saatiin varsin hyvä kuva OPC UA:sta, sen tavoitteista ja toteutuksesta. Hyvää materiaalia löytyi myös OPC-säätiön sekä eräiden OPC UA:lle ohjelmistoja tekevien yritysten www-sivuilta. Esimerkiksi Matrikonilla on laaja kokoelma artikkeleita, jotka saa luettavaksi rekisteröitymällä.

UA:n suosio ja yhteistyö eri organisaatioiden kanssa todistaa avoimuuden, huolellisen suunnittelun ja standardoinnin eduista.

Teollinen internet on vielä suhteellisen uusi asia eikä suoraan aiheesta painettua materiaalia löytynyt. Mitään varsinaista standardiakaan ei asiasta ole. Ohjaavana tekijänä toimii saksalaisten teollisuus 4.0. Todennäköisesti IIC:kin alkaa toimia suunnannäyttäjänä.

Harjoitustehtävän ohjelmointi oli haastavaa, koska OPC UA -asiakkaan toteutuksesta c#-kielellä ei juurikaan löytynyt malliprojekteja. Tästä voisi päätellä, että OPC UA:n sovelluskehitysympäristöt ovat suosiossa. Näille toteutettuja esimerkkejä löytyikin internetistä merkittävä määrä.

Lähteet

1 Rinaldi, John. 2013. OPC UA: The Basics.

2 Lange, J., Iwaniz, F., Burke, T. J. 2010. OPC From Data Access To Unified Architecture, 4th rev.ed. : VDE Verlag.

3 Mahnke, W., Leitner, S.-H., Damm, M. 2009. OPC Unified Architecture: Springer.

4 OPC history. 2015. Verkkodokumentti. OPC Foundation. <https://opcfoundation.org/about/opc-foundation/history/>. Luettu 31.7.2015.

5 OPC Unified Architecture, Pioneer of the 4th industrial (r)evolution. 2015. Verkkodokumentti. <https://downloads.prosysopc.com/downloads/opc-brochure-pioneer-v2.pdf>. Luettu 17.10.2015.

6 OPC membershift benefits. 2015. Verkkodokumentti. OPC Foundation. <https://opcfoundation.org/membership/benefits/>. Luettu 19.10.2015.

7 Service Name and Transport Protocol Port Number Registry. 2015. Verkkodokumentti. IANA. <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt>. Päivitetty 17.2.2016. Luettu 11.8.2015.

8 How does OPC Foundation software support the Industry 4.0 revolution? 2015. Verkkodokumentti. OPC Foundation. <http://www.dau.dk/Content/file_knowledge_item/2014-03-13-Dau-OPC-UA-enabler-for-industry4-en_125_INT.pdf>. Luettu 17.10.2015.

9 OPC Unified Architecture Specfication, Part 1: Overview and Concepts, Release 1.02. 2012. Verkkodokumentti. OPC Foundation. <https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts/>. Luettu 7/2015.

10 OPC Unified Architecture Specification, Part 2: Security Model, Release 1.02. 2013. Verkkodokumentti. OPC Foundation. <https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-2-security-model/>. Luettu 7/2015.

11 OPC Unified Architecture Specification Part 3: Address Space Model, Release 1.02. 2012. Verkkodokumentti. OPC Foundation. <https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-3-address-space-model/> Luettu 7/2015.

12 OPC Unified Architecture Specification Part 4: Services, Release 1.02. 2012. Verkkodokumentti. OPC Foundation. <https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-4-services/>. Luettu 7/2015.

13 OPC Unified Architecture Specification Part 6: Mappings, Release 1.02. 2012. Verkkodokumentti. OPC Foundation. <https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-6-mappings/>. Luettu 7/2015.

14 OPC Unified Architecture Specification Part 7: Profiles, Release 1.02. 2013. Verkkodokumentti. OPC Foundation. <https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-7-profiles/>. Luettu 7/2015.

15 COM and DCOM in Microsoft Windows CE 3.0. 2000. Verkkodokumentti. Microsoft Corporation. <https://msdn.microsoft.com/en-us/library/ms834352.aspx>. Luettu 5.11.2015.

16 Burke, Thomas. OPC UA Connectivity. 2013. Verkkodokumentti. OPC Foundation. <http://www.slideshare.net/InduSoft/opc-ua-connectivity-with-indusoft-and-the-opc-foundation>. Luettu 10/2015.

17 EnableDCOM. 2015. Verkkodokumentti. Microsoft Corporation. <https://msdn.microsoft.com/en-us/library/windows/desktop/ms687298(v=vs.85).aspx>. Luettu 5.11.2015

18 Süss, Georg. 2012. On the Verge of a Breakthrough. Verkkodokumentti. Automation.com. <http://www.automation.com/automation-news/article/on-the-verge-of-a-breakthrough>. Luettu 5.11/2015.

19 Chen, Charles. 2015. Report by exception. Verkkodokumentti. Technews Publishing (Pty) Ltd. <http://www.instrumentation.co.za/8479a>. Luettu 27.11.2015.

20 IoT, OPC UA, Industry 4.0 An Overview. 2015. Verkkodokumentti. OPC Foundation. <http://www.matrikonopc.com/portal/downloads/webcasts/MOPC_InternetOfThings_Webcast_EN.wmv>. Katsottu 08/2015.

21 Update for IEC 62541 (OPC UA) Published. 2015. Verkkodokumentti. OPC Foundation. <https://opcfoundation.org/news/opc-foundation-news/update-iec-62541-opc-ua-published/>. Luettu 30.11.2015.

22 Markets & Collaboration. 2015. Verkkodokumentti. OPC Foundation. <https://opcfoundation.org/markets-collaboration/ >. Luettu 15.10.2015.

23 BIG-EU and OPC Foundation to Cooperate. 2012. Verkkodokumentti. Automation.com. <http://www.automation.com/automation-news/industry/big-eu-and-opc-foundation-to-cooperate>. Luettu 15.10.2015.

24 Vieille, Jean. 2010. An ISA-95 companion standard for OPC UA. Verkkodokumentti. ISA. <http://www.isa-france.org/telechargement/fichiers/OPCUAISA95_WhitePaper_v1.pdf>. Luettu 15.10.2015.

25 OPC Unified Architecture Specification. 2015. Verkkodokumentti. OPC Foundation. <https://opcfoundation.org/developer-tools/specifications-unified-architecture>. Luettu 2.11.2015.

26 IoT Overview. 2015. Verkkodokumentti. Cisco. <http://www.cisco.com/web/solutions/trends/IoT/overview.html>. Luettu 3.11.2015.

27 Ylen tv-uutiset. 25.9.2015.

28 Beal, Vangie. 2015. big data. Verkkodokumentti. webopedia.com. <http://www.webopedia.com/TERM/B/big_data.html>. Luettu 3.11.2015.

29 INDUSTRIAL AUTOMATION LANGUAGE OF THE FUTURE. 2015. Verkkodokumentti. euautomation.com. <http://www.euautomation.com/assets/pdf/automated/specialreports/special-report-04.pdf>. Luettu 3.11.2015.

30 Reference Architecture Model Industrie 4.0 (RAMI4.0). 2015. Verkkodokumentti. VDI/VDE-Gesellshaft. <http://www.zvei.org/Downloads/Automation/5305%20Publikation%20GMA%20Status%20Report%20ZVEI%20Reference%20Architecture%20Model.pdf>. Luettu 1.11.2015.

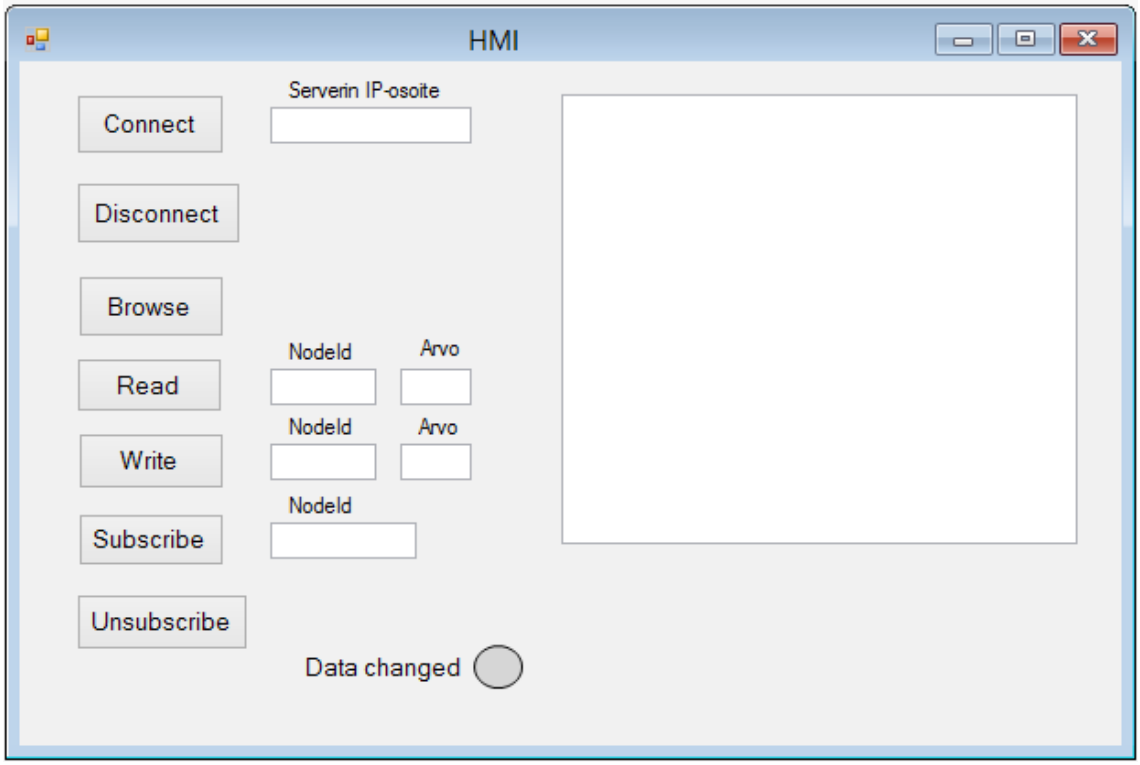
31 Industrial Internet Consortium. 2015. Verkkodokumentti. Industrial Internet Consortium.<http://www.iiconsortium.org/>. Luettu 4.11.2015.

32 Daugherty, P., Banerjee, P., Negm, W., Alter, A. Driving Unconventional Growth through the Industrial Internet of Things. 2014. Verkkodokumentti. Accenture. <https://www.accenture.com/us-en/_acnmedia/Accenture/next-gen/reassembling-industry/pdf/Accenture-Driving-Unconventional-Growth-through-IioT.pdf>. Luettu 3.11.2015.

33 Automaatioväylä 5/2015.

34 Hoppe, Stefan. 2015. OPC Unified Architecture as a Standard Integration Platform for the (Real-Time) Enterprise. Verkkodokumentti. OPC Foundation. <IoTi4sep1020151505opcfoundationua-from-sensor-to-cloud-150914090049-lva1- app6892.pdf>. Luettu 4.11.2015.

Liite 1. Harjoitustehtävän ohje



Harjoitustehtävässä tulee tehdä otsikkokuvan mukainen käyttöliittymä OPC UA -asiakkaalle, sekä toteuttaa sen toiminnat. Palvelin löytyy osoitteesta: _____

Esitiedot

OPC UA tarvitsee kirjastot Opc.Ua.Server.dll, Opc.Ua.Client.dll ja Opc.Ua.Core.dll
Ne löytyvät esimerkiksi osoitteesta:
http://opcuaservicesforwpf.codeplex.com/downloads/get/806246

Lataa tiedosto opcuaservices.zip, pura esimerkiksi työpöydälle. Dll:t ovat kansiossa v50/lib/.

Klikkaa Visual Studiossa oletusasetuksissa oikealla olevassa Solutions Exploresissa oikealla, avautuvasta valikosta valitse Add Reference, hae yllä mainitut dll:t ja ne sisältöineen tulevat mukaan projektiin.

Lisätehtävä

Kytke OPC UA asiakas johonkin pilvipalvelimeen, eli talleta luetut muuttujan arvot pilveen.

Liite 2. Harjoitustehtävän esimerkkikoodi ilman kirjautumista

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Opc.Ua;
using Opc.Ua.Client;
using System.Diagnostics;
using System.Net;
using System.Net.Security;

namespace inssi_harjoitus
{

    public partial class HMI : Form
    {
        private ushort namespaceIndex = 2;
        private NotificationMessage nm;
        private Session session;
        public HMI()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
```

```
    private void connect_Click(object sender, EventArgs e)
    {
        browse_ikkuna.Clear();
        string osoite = serverin_ip.Text; /// ip:portti
        var config = new ApplicationConfiguration()
        {
            ApplicationName = "harjoitus",
            ApplicationType = ApplicationType.Client,
                                SecurityConfiguration  =  new
SecurityConfiguration
            {
                ApplicationCertificate =
                new CertificateIdentifier
                {
                    StoreType = "",
                    StorePath = "",
                    SubjectName = Utils.Format("", "", null)
                },
                TrustedPeerCertificates =
                  new CertificateTrustList { StoreType = "",
StorePath = "", },
                NonceLength = 32,
                AutoAcceptUntrustedCertificates = true
            },
                                TransportConfigurations  =  new
TransportConfigurationCollection(),
                      TransportQuotas  =  new  TransportQuotas
{ OperationTimeout = 15000 },
                ClientConfiguration = new ClientConfiguration
{ DefaultSessionTimeout = 60000 }
        };
        config.Validate(ApplicationType.Client);
        try
        {
                       session  =  Session.Create(config,  new
ConfiguredEndpoint(null,                                    new
```

```
EndpointDescription("opc.tcp://"  +  osoite)),  true,  "",
60000, null, null);
            disconnect.Enabled = true;
        }
        catch
        {
            browse_ikkuna.Clear();
            browse_ikkuna.ForeColor = Color.Red;
                   browse_ikkuna.Text  =  ("Tarkista  serverin
osoite!");
        }
    }

     private void disconnect_Click(object sender, EventArgs
e)
    {
        if (session.Connected.Equals(true))
        {
            read_arvo.Clear();
            browse_ikkuna.Clear();
            disconnect.Enabled = false;
            session.Close();
        }
    }

    private void browse_Click(object sender, EventArgs e)
    {
        browse_ikkuna.Clear();
        browse_ikkuna.ForeColor = Color.Black;
        browse_ikkuna.WordWrap = true;
        browse_ikkuna.AcceptsReturn = true;
        browse_ikkuna.Multiline = true;
        ReferenceDescriptionCollection refs;
        Byte[] cp;
        try
        {
            StringBuilder naytto = new StringBuilder();
```

```
                string otsikko = "DisplayName: BrowseName,
NodeClass";
            naytto.Append(otsikko).AppendLine();
                                session.Browse(null,  null,
ObjectIds.ObjectsFolder, 0u, BrowseDirection.Forward,
              ReferenceTypeIds.HierarchicalReferences, true,
(uint)NodeClass.Variable | (uint)NodeClass.Object |
            (uint)NodeClass.Method, out cp, out refs);

        foreach (var rd in refs)
        {
               string rivi = String.Format("{0}: {1}, {2}",
rd.DisplayName, rd.BrowseName, rd.NodeClass);
            naytto.Append(rivi).AppendLine();
            ReferenceDescriptionCollection nextRefs;
            byte[] nextCp;
                                session.Browse(null,  null,
ExpandedNodeId.ToNodeId(rd.NodeId,   session.NamespaceUris),
0u,
                                    BrowseDirection.Forward,
ReferenceTypeIds.HierarchicalReferences,              true,
(uint)NodeClass.Variable |
                                  (uint)NodeClass.Object  |
(uint)NodeClass.Method, out nextCp, out nextRefs);
            foreach (var nextRd in nextRefs)
            {
                string rivi2 = (String.Format("+  {0}: {1},
{2}",         nextRd.DisplayName,         nextRd.BrowseName,
nextRd.NodeClass));
              naytto.Append(rivi2).AppendLine();
            }
            browse_ikkuna.Text = naytto.ToString();
        }
        }
        catch { }
        }
```

```
    private void read_Click(object sender, EventArgs e)
    {
        browse_ikkuna.Clear();
        try
            {
                                        NodeId val = new
NodeId(Convert.ToUInt16(read_ni.Text), namespaceIndex);
                                           read_arvo.Text =
session.ReadValue(val).ToString();
            }
        catch
            {
                browse_ikkuna.Clear();
                browse_ikkuna.ForeColor = Color.Red;
                browse_ikkuna.Text = ("Tuntematon NodeId!");
            }
    }

    private void write_Click(object sender, EventArgs e)
    {
        browse_ikkuna.Clear();
        try
        {
                                        NodeId val = new
NodeId(Convert.ToUInt16(write_ni.Text), namespaceIndex);
                      NodeIdCollection nodesToWrite = new
NodeIdCollection();
            nodesToWrite.Add(val);

            DataValue arvo = new DataValue();
            arvo.Value = write_arvo.Text;

                     DataValueCollection WriteValues = new
DataValueCollection();
            WriteValues.Add(new DataValue(arvo));
```

```
                WriteValue attributeToWrite = new WriteValue();
                attributeToWrite.NodeId = val;
                attributeToWrite.AttributeId = Attributes.Value;
                attributeToWrite.Value = WriteValues[0];
                       WriteValueCollection valuesToWrite = new
    WriteValueCollection();
                valuesToWrite.Add(attributeToWrite);
                StatusCodeCollection results = null;
                DiagnosticInfoCollection diagnosticInfos = null;
                 session.Write(null, valuesToWrite, out results,
    out diagnosticInfos);
            }
            catch
            {
                browse_ikkuna.Clear();
                browse_ikkuna.ForeColor = Color.Red;
                browse_ikkuna.Text = ("Tuntematon NodeId!");
            }
        }
    }
}
```