TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MAURI MUSTONEN

SÄHKÖASEMAN ÄLYKKÄÄN ELEKTRONIIKKALAITTEEN VIESTIEN TILAUS JA PROSESSOINTI

Diplomityö

Tarkastaja: Professori Kari Systä

Jätetty tarkastettavaksi 17. toukokuuta 2018

TIIVISTELMÄ

MAURI MUSTONEN: sähköaseman älykkään elektroniikkalaitteen viestien tilaus ja prosessointi
Tampereen teknillinen yliopisto
Diplomityö, 31 sivua
Toukokuu 2018
Tietotekniikan koulutusohjelma
Pääaine: Ohjelmistotuotanto
Tarkastaja: Professori Kari Systä

Avainsanat: Ohjelmistotuotanto, IEC 61850, MMS, AMQP

Tiivistelmä on suppea, 1 sivun mittainen itsenäinen esitys työstä: mikä oli ongelma, mitä tehtiin ja mitä saatiin tulokseksi. Kuvia, kaavioita ja taulukoita ei käytetä tiivistelmässä.

Laita työn pääkielellä kirjoitettu tiivistelmä ensin ja käännös sen jälkeen. Suomenkieliselle kandidaatintyölle pitää olla myös englanninkielinen nimi arkistointia varten.

Sähkönjakeluverkko on tärkeä osa nykyistä yhteiskuntaa ja sen päivittäistä toimintaa. Siksipä verkon toiminta on elintärkeää yhteiskunnan sujuvan toiminnan kannalta. Sähköverkko koostuu sähköntuotantolaitoksista, sähkölinjoista ja sähköasemista. Sähköverkon eri komponenttien avulla sähkö toimitetaan tuontantolaitoksesta kuluttajan seinäpistokkeeseen asti. Sähköasemat ja niiden automatisointi ovat tärkeässä roolissa verkon yleisen toiminnan ja turvallisuuden takaamiseksi.

Nykypäivänä sähköverkon sähköasemien automatisointiin käytetään tietokoneita, kommunikointi- ja verkkolaitteistoja [14, s. 659]. Asemien automatisointi toteutetaan *älykkäillä elektronisilla laitteilla* (eng. IED), jotka toteuttavat aseman monitoroinnin ja ohjaamisen. IED:t konfiguroidaan toteuttamaan aseman erilaisia funktioita ja toiminnallisuuksia. Monen eri toimijan toimiessa allalla, voisivat kaikki vapaasti toteuttaa oman version laitteistaan ja määrittää kuinka näitä käytettäisiin. Tästä seurauksena olisi, että laitteet eivät olisi yhteensopivia muiden toimittajien laitteiden kanssa. Tilannetta helpottamaan ja mahdollistamaan eri valmistajien yhteensopivuuden laitteiden välillä. On *International Electrotechnical Commission* määrittänyt standardin nimeltä IEC 61850. Standardi määrittää kommunikointi protokollat sähköasemien IED laitteille. Standardi määrittää abstrakteja malleja sähköaseman kommunikoinnin mallintamiseen oliopohjaisesti. Tekniikasta tai toteutuksesta riippumaton standardimalli voidaan määrittää toimivaksi eri tekniikoilla. Standardi määrittää malleja myös erilaisten datapisteiden tilaukseen viesteinä IED:ltä. Eri asiakastoteutukset voivat tilata laitteeseen määritettyjä datapisteitä standardin määrittämillä säännöillä. Standardin ansiosta viestien muoto on ennaltamäärätty ja on tekniikasta riippumaton. Asiakasohjelmisto voi käyttää viestejä esimerkiksi jännitemittatietojen tilaukseen.

Tässä diplomityössä keskitytään yllämainitun viestejä tilaavan asiakasohjelmiston suunnitteluun ja toteutukseen. Työ sisältää tutkimusta ja vertailua arkkitehtuuriin ja toteutukseen liittyen. Lopullinen toteutus asiakasohjelmistosta tilaa viestejä IED:ltä, prosessoi ja uudelleenjulkaisee ne myöhempää käyttöä varten. Ennen työn aloitusta, ohjelmistosta oli jo saatavilla protoversio, jota hyödynnettiin uuden toteutuksen suunnittelussa. Protoversiossa olevat ongelmat ja puutteet korjattiin uuteen toteutukseen.

ALKUSANAT

Mistä tämän diplomityönaiheen sain ja kiittää eri ihmisiä ketä työssä oli sidoshenkilöinä.

Tampereella, 19.4.2018

Mauri Mustonen

SISÄLLYSLUETTELO

KUVALUETTELO

TAULUKKOLUETTELO

LYHENTEET JA MERKINNÄT

ACSI	engl. *Abstract Communication Service Interface*, IEC 61850 -standardin käyttämä lyhenne kuvaamaan palveluiden abstraktimalleja
AMQP	engl. *Advanced Message Queuing Protocol*
FFI	engl. *Foreign Function Interface*, mekanismi, jolla ajettava ohjelma voi kutsua toisella kielellä implementoitua funktiota
GIL	engl. *Global Interpreter Lock*, tulkattavassa kielissä oleva globaali lukitus, joka rajoittaa yhden säikeen suoritukseen kerrallaan
HAL	engl. *Hardware Abstraction Layer*, laitteistoabstraktiotaso abstraktoimaan laitteen toiminnalisuus lähdekoodista
IED	engl. *Intelligent Electronic Device*, sähköaseman älykäs elektroninen laite, joka tarjoaa toimintoja monitorointiin ja kontrollointiin
MMS	engl. *Manufacturing Message Specification*
RCB	engl. *Report Control Block*, raporttien konfigurointiin ja tilaukseen tarkoitettu lohko asiakasohjelmalle
XML	engl. *Extensible Markup Language*, laajennettava merkintäkieli, joka on ihmis- ja koneluettava

1. JOHDANTO

Kirjoita tähän johdantoa työstä ja aiheesta. Kuinka työ valittiin ja miksi tekijä valitsi tämän työn. Kirjoita myös mitä tehtiin. Kokonaiskuva työstä pitäisi saada johdannosta. Alusta lukijaa todella hyvin yleismaallisella kuvalla ja taustalla. Asiaa pitäisi olla hyvin hallussa ennen teoriaosuuteen siirtymistä.

Tämän osion lukemalla lukijan pitää tietää:

- *Miksi lukija valitsi tämän aiheen?*
- *Mikä on sähköaseman IED ja mitä se asemassa tekee?*
- *Mitä standardi suurinpiirtein on ja mitä se tarkoittaa työn kannalta?*
- *Kenelle työ tehtiin?*
- *Mikä on työn haluttu lopputulos ja tavoitteet?*
- *Mihin työ keskittyy kaikesta eniten?*
- *Mikä on työn tausta ennen työn aloittamista?*

Tämä teksti tarvitsee vielä hiomista ja huomiota. On kuitenkin suuntaa antava miltä lopullinen tulee näyttämään.

Tämä diplomityö on tehty Alsus Oy:lle, joka oli työn tekohetkellä tekijän työpaikka vuonna 2018. Tekijä valitsi työn aiheen mielenkiinnon ja ajankohdan sopiivuden takia. Työ liittyi sopivasti ajanhetkellä sen hetkisiin työtehtäviin.

Nykypäivänä sähköverkot ovat iso yhteiskuntaa ja sen sujuvaa toimivuutta. Ilman sähköä ei moni asia nykypäivänä toimisi niinkuin se on. Sähköä tarvitaan joka paikassa ja tietotekniikan lisääntyessä vieläkin enemmän. Nykypäivän sähköverkko koostuu useista erillisitä komponenteista, joita on mm. sähköntuontantolaitos, sähkölinjat ja sähköasemat. Sähköasemien tehtävä verkossa on toteuttaa erilaisia toiminnallisuuksia, kuten muuntaja, jakaminen ja verkon toiminnan tarkkailu. Nykyisin sähköasemat eivät tarvitse henkilökuntaa paikalle, kuin houltotehtävissä. Asemien toiminnallisuutta voidaan seurata etänä. Vian tai huollon tarpeen sattuessa, huoltomies käy tarkistamassa asian paikanpäällä. Sähköaseman yksi tärkeä tehtävä on tarkkailla verkon toimivuutta ja vikatilanteen tapahtuessa esimerkiksi katkaista linjasta virrat pois vikatilanteen sattuessa. Vikatilanne voisi olla kaapelin poikkimeno ja virta pääsisi tätä kautta maihin.

Sähköasemien funktionaalisuutta nykypäivänä toteuttaa niin sanottu älykäs elektroniikkalaite (engl. Intelligent Electronic Device, lyhennetään IED). IED voidaan kytkeä ja konfiguroida toteuttamaan monta aseman eri funktionaalisuutta. IED:t voivat kommunikoida

verkon yli aseman muun logiikan ja muiden IED-laitteiden kanssa. Nykypäivänä verkon nopeus ja mahdollistaa reaaliaikaisen kommunikoinnin asemilla sen eri laitteiden välillä. IED voidaan esimerkiksi konfiguroida hoitamaan sähkölinjan kytkimenä oloa, joka myös tarkkailee linjan toimintaa mittaamalla konfiguroituja arvoja, kuten jännitettä ja virtaa. Vikatilanteen sattuessa IED katkaisee linjan virrasta enemmän vahingon välttämiseksi. Linjan korjauksen jälkeen virta kytketään takaisin päälle.

Monen eri toimijan toimiessa laitteita tuottavalla allalla ja sähköasema suuren elektronisen laitteiden määrän takia. On määritetty maailmanlaajuinen standardi IEC 61850 määrittämään koko aseman laitteiden välistä kommunikointiprokollia varten. Standari määrittää eri valmistajien IED:laitteille samat yhteiset kommunikointiprotokollat joita noudattamalla eri valmistajien laitteet sopivat yhteen.

Standardi määrittää mallit erilaisten viestien tilaukseen, jolla tilaaja voi tilata haluttuja konfiguroituja datapisteitä verkon yli. Tässä työssä keskitytään tämän asiakasohjelmiston suunniteluun ja toteutukseen.

1.1 Tausta

Aikaisemmin mainitusta asiakasohjelmisto oli jo olemassa protoversio ennen työn aloittamista. Ohjelmisto pystyi tilaamaan, prosessoimaan ja tallentamaan saatuja viestejä tietokantaan. Ohjelmassa oli kuitenkin suorituskykyongelmia ja eikä se tukenut kaikkia standardin määrittämiä ominaisuuksia. Niinpä uudellelle asiakasohjelman suunnittelulle ja toteutukselle oli tarve, joka myös korjaisi entisen toteutuksen ongelmat.

1.2 Laajuus

Työssä keskityttiin uuden asiakasohjelman suunnitteluun toteutukseen. Työssä tehtiin tutkimusta vertailemalla erilaisia arkkitehtuureita ja näistä valitsemalla tarpeisiin sopiva. Tutkimusta tehtiin myös protoversioon, selvittämään sen suorityskykyongelmia ja kuinka nämä voitiin ottaa huomioon uudessa ohjelmistossa. Samalla käytiin läpi entisen toteutuksen arkkitehtuuria, tarkoituksena miettiä laajennusmahdollisuuksia tulevaisuutta varten uudelle toteutukselle.

1.3 Tavoitteet

Tavoitteena työssä on toteuttaa uusi asiakasohjelmisto, joka kokonaan korvaisi entisen protoversion, ja toteuttaa kaikki standardin määrittämät toiminnallisuudet. Ohjelmiston pitäisi myös olla suorituskykyinen ja siinä ei saa olla samoja ongelmia kuin protoversiossa. Uuden toteutuksen pitäisi myös olla laajennettavissa uusille ominaisuuksille ja vaatimuksille.

1.4 Työn rakenne

Ensin työssä pohjustetaan toteutuksen ymmärtämiseen tarvittua teoriaa. Teorian jälkeen suunnitellaan itse toteutus ja argumentoidaan miksi tiettyihin valintoihin työssä päädyttiin. Suunnitelma pohjustaa tulevaa toteutusta ja sen arkkitehtuuria. Suunnittelun jälkeen tulee itse toteutus ja sen läpikäynti. Mitä kirjastoja ja tekniikoita toteutukseen on käytetty ja mikä on minkäkin tarkoitus. Toteutuksen tarkoitus on tarjoa lukijalle kuva toteutuksen tärkeistä rakenteista ja komponenteista. Lopussa tehdään tuloksien katselmointi ja tuloksia verrataan alussa asetettuihin tavoitteisiin. Tarkoituksena on antaa lukijalle kuva kuinka hyvin työn tavoitteisiin päästiin ja mitä olisi voinut tehdä toisin. Lopussa myös esitellään asioita jatkokehitystä ja tutkimusta varten.

2. TEORIA

Tässä osiossa lukijaa perehdytetään työn kannalta tärkeään teoriaan. Teoriaosuuden kokonaan lukemalla lukija ymmärtää, mitä IEC 61850 -standardi tarkoittaa sähköasemien kannalta ja mitä kaikkea se määrittää. Kuinka standardi määrittää viestien tilauksen, ja mitä malleja ja palveluita niihin liittyy. Työn lopullisessa toteutuksessa viestit prosessoitiin ja julkaistiin jonopalvelimelle myöhempää käyttöä varten. Teorian lopussa lukija perehdytetään jonopalvelimen toteukseen liityvään teoriaan. Teoriaosuus lukemalla lukija varmistaa että, ymmärtää ohjelmiston suunnittelu- ja toteutusvaiheessa käsitellyt käsitteet ja aiheet.

2.1 IEC 61850 -standardi yhteiseen kommunikointiin

Sähköasemilla nykypäivänä käytössä olevilla älykkäillä elektronisilla laitteilla (engl. Intelligent Electronic Device, lyhennetään IED) toteutetaan aseman toiminnalisuuden funktioita. Aseman toiminnallisuuteen liittyy sen kontrollointi ja suojaus. Aseman komponenttien suojauksen lisäksi, siihen kuuluu myös asemalta lähtevät sähkölinjat. Hyvä esimerkki sähköaseman suojauksesta on korkeajännitelinjan katkaisija, joka katkaisee virran linjasta vikatilanteissa, kuten linjan poikkimeno kaatuneen puun tai pylvään takia. Fyysistä katkaisijaa ohjaa aseman automatiikka, joka toteutetaan IED-laitteilla. Eli IED-laite voi olla kytketty fyysisesti ohjattavaan laitteeseen [5, s. 63–64]. Koko sähköaseman toiminnallisuus koostuu monesta funktiosta, jotka on jaettu monelle IED-laitteelle. Jotta systeemi pystyy toimimaan, täytyy IED-laitteiden kommunikoida keskenään ja vaihtaa informaatiota toistensa kanssa. IED-laitteiden täytyy myös kommunikoida asemalta ulospäin erilliselle ohjausasemalle monitorointia ja etäohjausta varten [1, s. 1]. On selvää, että monimutkaisen systeemin ja monen valmistajien kesken tarvitaan yhteiset säännöt yhteistä kommunikointia varten.

Maailmanlaajuisesti määritetty IEC 61850 -standardi määrittää sähköaseman sisäisen kommunikoinnin IED-laitteiden välillä. Standardi määrittää myös kommunikointisäännöt asemalta lähtevään liikenteeseen, kuten toiselle sähköasemalle ja ohjausasemalle [5, s. 10]. Ilman yhteisiä sääntöjä jokainen valmistaja olisi vapaa toteuttamaan omat säännöt ja protokollat kommunikointiin. Seurauksena tästä olisi, että laitteet eivät olisi keskenään yhteensopivia eri valmistajien välillä. Standardin on tarkoitus poistaa tämä ja määrittää yhteiset pelisäännöt kommunikoinnin toteuttamiseen [9, s. 1].

Todella tärkeä ja iso osa standardia on sähköaseman systeemin funktioiden abstrahointi mallien kautta. Standardi määrittää tarkasti kuinka abstraktit mallit määritellään aseman oikeista laiteista ja niiden ominaisuuksista. Tarkoituksena on tehdä mallit tekniikasta ja

toteutuksesta riippumattomaksi. Tämän jälkeen abstrahoidut mallit mallinnetaan erikseen jollekin tekniikalle, joka sen mahdollistaa. Abtrahoituja malleja käytetään myös määrittämään sähköaseman IED-laitteiden ja aseman muiden osien konfigurointi. Abstrahoitujen mallien ansiosta standardi on pohjana tulevaisuuden laajennoksille ja tekniikoille. Uusien tekniikoiden ilmaantuessa, voidaan standardin lisätä osa, joka mallintaa abstraktimallit kyseiselle tekniikalle [1, s. 2]. Teoriassa ensin käsitellään standardin asioita abstraktilla tasolla, liittämättä sitä mihinkään tekniseen toteutukseen. Vasta tämän jälkeen se liitetään erikseen tekniikkaan mitä työssä käytettiin.

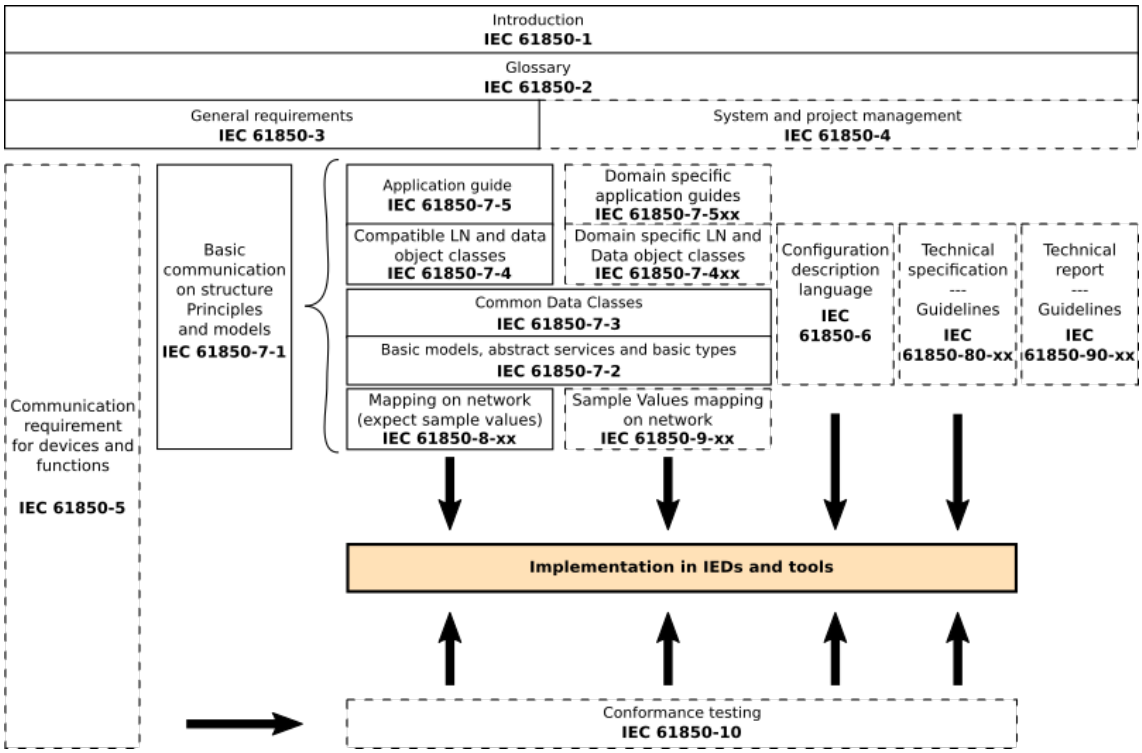
2.1.1 Standardin eri osat ja niiden merkitykset

IEC 61850 -standardi on todella laaja kokonaisuus. Tämän takia se on pilkottu erillisiin dokumentteihin, josta jokainen käsittelee omaa asiaa. Historian saatossa standardiin on lisätty uusia dokumentteja laajentamaan standardia [8, 11] [3, s. 13]. Tämän työn kirjoitushetkellä standardiin kuului lisäki paljon muitakin dokumentteja, esimerkiksi uusiin mallinnuksiin muille tekniikoille ja vesivoimalaitoksien mallintamiseen liittyviä dokumentteja. Laajuudesta huolimatta standardin voi esittää 10:llä eri pääkohdalla ja näiden alakohdilla. Taulukossa 1 on esitetty standardin pääkohdan dokumentit ja niiden alkuperäiset englanninkieliset otsikot [10, s. 2] [8]. Kuvassa 1 on esitetty kaikki standardin eri osat ja niiden väliset relaatiot toisiinsa. Kuvaan on merkitty yhteinäisellä viivalla ne osat, jotka ovat tämän työn kannalta tärkeitä. Ja katkoviivalla ne, jotka eivät ole.

***Taulukko 1**. IEC 61850 -standardin pääkohtien ja niiden alakohtien dokumentit.*

Osa	Otsikko englanniksi
1	Introduction and overview
2	Glossary
3	General requirements
4	System and project management
5	Communication requirements for functions and device models
6	Configuration description language for communication in power utility automation systems related to IEDs
7-1	Basic communication structure - Principles and models
7-2	Basic information and communication structure - Abstract communication service interface (ACSI)
7-3	Basic communication structure - Common data classes
7-4	Basic communication structure - Compatible logical node classes and data object classes
8-1	Specific communication service mapping (SCSM) - Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3
9-2	Specific communication service mapping (SCSM) - Sampled values over ISO/IEC 8802-3
9-3	Precision time protocol profile for power utility automation
10	Conformance testing

Standardin ensimmäiset osat 1–5 kattavat yleistä kuvaa standardista ja sen vaatimuksista. Osiossa 6 käsitellään IED-laitteiden konfigurointiin käytetty XML (engl. Extensible Mar-



Kuva 1. *IEC 61850 -standardin osat ja niiden väliset relaatiot [5, s. 14] [3, s. 22].*

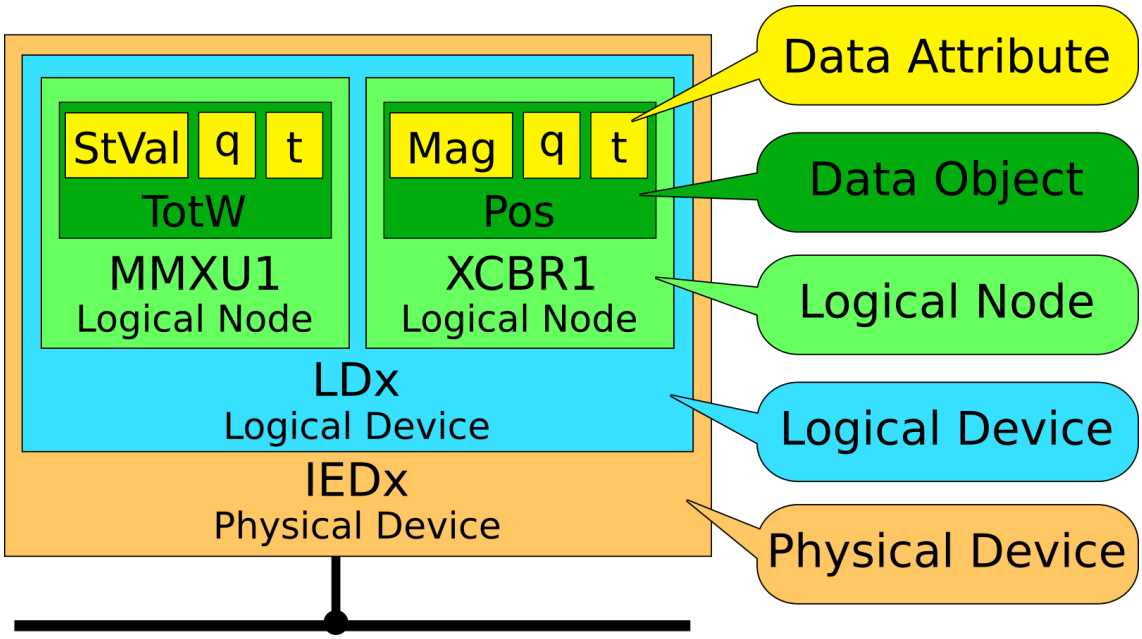
kup Language) -pohjainen kieli [4, s. 7–8]. Tämä osuus ei ole tämän työn kannalta tärkeä ja sitä ei sen tarkemmin käsitellä. Osat 7.1–7.4 käsittelevät standardin abstraktia mallia, niiden palveluita ja kuinka se rakentuu. Abstrahoidut palvelut standardissa lyhennetään ACSI (engl. Abstract Communication Service Interface), ja samaa lyhennettä käytetään tässä työssä [5, s. 72]. Osissa 8–9 ja niiden alakohdissa käsitellään abstraktimallien mallintamista erillisille protokollille, jolloin malleista tulee kyseisestä tekniikasta riippuvaisia. Abstrakteja malleja ja niiden mallintamista tekniikalle käsitellään teoriassa erikseen. Osa 10 käsittelee testausmenetelmiä, joilla voidaan varmistaa standardin määritysten noudattaminen. Tämä osuus ei myöskään ole tämän työn kannalta tärkeä, ja sitä ei teoriassa sen takia käsitellä. [5, s. 15]

2.1.2 Abstraktimalli ja sen osat

Kirjoita tähän aseman funktioiden, fyysisen laitteiden ja loogisten noodien suhteesta toisiinsa. Kuva myös piirrä ja mallia ja tietoa tähän ota [3, s. 19]. Tälle tarve jos teksti ei ole muuten tarpeeksi ymmärrettävä.

Piirrä tähän itse esimerkkikuva kuinka standardi mallintaa fyysisen laitteen loogisiksi laitteiksi. Ota mallia standarin 7-1 kuvasta sivulla 17.

IEC 61850 -standardin lähtökohtana on pilkkoa koko sähköaseman toiminnalisuuden funktiot pieniksi yksilöiksi. Pilkotut yksilöt abstrahoidaan ja pidetään sopivan kokoisina, jotta ne voidaan konfiguroida esitettäväksi erillisellä IED-laiteella. Yksi aseman funktio voi-



Kuva 2. *IEC 61850 -standardin abstraktimallin osat ja niiden hierarkia.*

daan hajauttaa monelle eri IED-laitteelle. Esimerkiksi linjan suojaukseen liittyvät komponentit, katkaisija (engl. circuit braker) ja ylivirtasuoja (engl. overcurrent protection) omilla IED-laitteillaan. Toimiakseen, laitteiden täytyy vaihtaa informaatiota keskenään [5, s. 31]. Näitä pilkottuja yksilöitä kutsutaan standardissa nimellä looginen noodi (engl. logical node ja lyhennetään LN). Loogiset noodit siis mallinetaan jostakin systeemin käsiteellisestä osasta. Loogisia noodeja käytetään rakentamaan looginen laite (engl. logic device, lyhennetään LD). Looginen laite on aseman ohjausyksikkö ja jokin fyysisen laitteen osa, joka toteuttaa loogisten noodien ohjauksen yhtäaikaa. Ylläolevasta esimerkistä looginen laite olisi aseman linjan suojaukseen liittyvät osat sisältä laite. Looginen laite siis vastaa aseman fyysistä laitetta, joka on kytketty aseman verkkoon ja sillä on IP-osoite. Yksi aseman fyysinen laite voi hoitaa monen loogisen laitteen funktionaalisuuden. Kuvassa 2 on esitetty standardin mallin eri osien hierarkia ja kuinka ne rakentuvat [2, s. 2] [3, s. 24].

Standardin määrittämissä osien hierarkiassa looginen laite on ylin yksilö, joka sisältää loogisia noodeja. Kuvassa 2 IEDx ja LDx vastaavasti. Looginen noodi sisältää data objekteja (engl. data object, lyhennetään DO). Kuvassa 2 loogiset noodit XCBR1 ja MMXU1. Ja data objektit Pos ja TotW. Data objekti sisältää data attribuutteja (engl. data attribute, lyhennetään DA). Kuvassa 2 StVal, Mag, q ja t. Data objekti on tapa koostaa yhteen samaan asiaan liittyvät data attribuutit. Data attribuutit ovat laitteen konfiguroitavia ja luettavia datapisteitä. Data attribuutit kuvaavat esimerkiksi fyysisen laitteen tilaa ja mittausarvoja. Esimerkiksi mitattua jännitettä tai katkaisimen tilaa (kiinni tai auki). Standardin määrittämät data- objektit ja attribuutit voidaan lajitella 5 eri ryhmään:

- yleinen loogisen noodin informaatio,
- tila informaatio,

- asetukset,
- mitatut arvot ja
- ohjaus [3, s. 25].

Tässä vaiheessa on hyvä mainita, että standardi pyrkii esittämään aseman funktioiden toiminnallisuutta hierarkialla ja oliopohjaisesti. Oliopohjaisesti siten, että standardi määrittää valmiita luokkia erilaisille loogisille noodeille. Esimerkiksi katkaisijalle on määritelty luokka nimeltä XCBR (circuit braker) [7, s. 105–106]. Ja mittaukselle MMXU (measurement) [7, s. 57–58]. Kun aseman toiminnallisuutta esitetään konfiguraatiossa ja IED-laitteella, luokia instanssioidaan tarpeen mukaan. Esimerkiksi kuvassa 2 aikaisemmin mainitut loogisen noodin luokat on instansioitu nimellä XCBR1 ja MMXU1.

Standardi määrittää todella paljon erilaisia valmiita luokkia erilaisille loogisille noodeille valmiina käytettäväksi. Standardi myös määrittää laajenoksien mahdollisuudet luokkia käyttäen. Kaikki määritetyt luokat loogisille noodeille voi löytää standardin osasta 7-4.

2.1.3 Loogisen noodin luokkien ja attribuuttien rakentuminen

Ennen kirjoittamista lue 7-2 osuus ja selvitä meta-luokat standardista ja mitä ne pohjalla määrittävät ennen CDC-luokkia. Sen jälkeen kirjoita tähän kuinka standardin loogiset noodit rakentuvat. Kerro mitä common data classes sisältää ja kuinka ne määrittää data monessa paikassa käytetyt data attribuutit. Tämän jälkeen kuinka loogisen noodin luokat rakennetaan käyttämällä CDC määrityksiä [3, s. 26]. Ota tähän myös kuvia taulukosta ja rakenna esimerkkinä yksi looginen noodi niitä käyttäen. Tätä samaa mallia voi myöhemmin käyttää määrittämään kuinka referensssipolut luokkien nimillä rakentuu. Mallia loogisen noodin rakentamiseen voi myös ottaa [5, s. 27].

Kuvat: Kuva tai taulukko common data classin kentistä, logical noden kentistä mitä käytetään. Kuva lopullisen loogisen noodin luokan attribuuteista.

IEC 61850 -standardissa määritettyjen luokkien rakennetta on lähestytty oliopohjaisesti. Kaikki luokat määritellään standardissa taulukoilla, joissa on standardoitu kentän nimi, tyyppi, selitys ja onko kenttä optionaalinen Standardissa osassa 7-4 on lista kaikista sen määrittämistä loogisen noodin luokista eri tarkoituksiin.

Loogisen noodin luokan kentät, kuten Pos, rakentuvat standardin yleisistä luokista (engl. Common Data Class, lyhennetään CDC). CDC-luokat ovat luokkia jotka sisältävät data attribuutteja, ja ovat yhteisiä monelle eri määritykselle. CDC-luokkien määritykset löytyy standardin osasta 7-3.

Kuvassa 3 on esitetty standardin XCBR-luokan määritys taulukkomuodossa. Taulukko määrittää luokan kentät ja niiden nimet. Taulukon viimeinen sarake M/O/C, kertoo onko kenttä pakollinen (Mandatory, M), optionaalinen (Optional, O), vai konditionaalinen (Conditional, C). Taulun

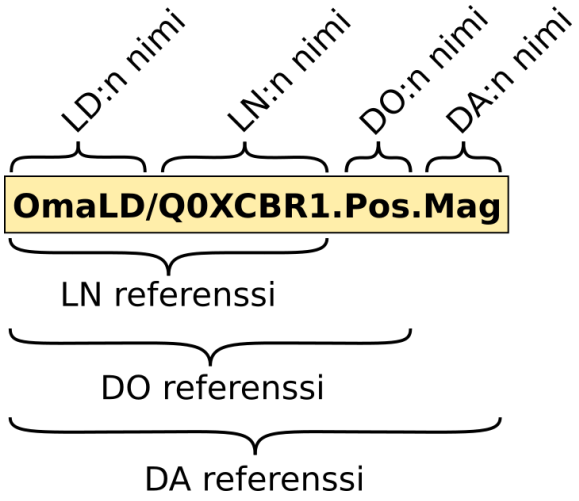
XCBR class				
Data object name	**Common data class**	**Explanation**	**T**	**M/O/C**
LNName		The name shall be composed of the class name, the LN-Prefix and LN-Instance-ID according to IEC 61850-7-2, Clause 22.		
Data objects				
Descriptions				
EEName	DPL	External equipment name plate		O
Status information				
EEHealth	ENS	External equipment health		O
LocKey	SPS	Local or remote key (local means without substation automation communication, hardwired direct control)		O
Loc	SPS	Local control behaviour		M
OpCnt	INS	Operation counter		M
CBOpCap	ENS	Circuit breaker operating capability		O
POWCap	ENS	Point on wave switching capability		O
MaxOpCap	INS	Circuit breaker operating capability when fully charged		O
Dsc	SPS	Discrepancy		O
Measured and metered values				
SumSwARs	BCR	Sum of switched amperes, resettable		O
Controls				
LocSta	SPC	Switching authority at station level		O
Pos	DPC	Switch position		M
BlkOpn	SPC	Block opening		M
BlkCls	SPC	Block closing		M
ChaMotEna	SPC	Charger motor enabled		O
Settings				
CBTmms	ING	Closing time of breaker		O

Kuva 3. *IEC 61850 -standardin katkaisijaluokan XCBR -määritys [7, s. 106].*

2.1.4 Attribuuttien viittaus hierarkiassa

IEC 61850 -standardi määrittää tarkasti kuinka hierarkian eri kohtia viitataan IED-laitteessa. Viitteitä käytetään kun IED-laitteelle tehdään standardin osan 7-2 ACSI-määritysten mukaisia palvelukutsuja. Esimerkiksi jonkin attribuutin arvon asettaminen (SetDataValues) tai lukeminen (GetDataValues). Viitteen avulla IED-laite tietää, mihin loogisen noodin instanssiin ja sen data attribuuttiin palvelypyyntö kohdennetaan. Kuvassa 4 on esitetty kuinka standardi määrittää viitteen muodostumisen loogisesta laitteesta data attribuuttiin asti [5, s. 93].

Viite muodostuu suoraan laitteessa olevien luokkien instanssien nimien ja hierarkian mukaan. Loogisen laitteen (LD) ja loogisen noodin (LN) erottimena käytetään kauttaviivaa, ja muiden osien erottimena käytetään pistettä. Loogisella laitteella voi olla käyttäjän oma määrittämä nimi, mutta kuitenkin alle 65 merkkiä. Loogisen laitteen nimeen standardi ei puutu. Loogisen noodin instanssin nimi koostuu alku-, keski- ja loppuosasta. Alkuosan käyttäjä voi itse päättää, kuvassa 4 "*Q0*". Voi sisältää numeroita ja kirjaimia, mutta täytyy alkaa kirjaimella. Keskiosan täytyy olla loogisen luokan nimi, josta instanssi on tehty. Tässä tapauksessa jo aikaisemmin mainittu katkaisijan luokka, XCBR. Tämä osuus on aina 4 kirjainta pitkä ja on aina isoilla kirjaimilla. Loppuosa on instanssin numeerinen arvo,



Kuva 4. *IEC 61850 -standardin määrittämä viitteen rakenne.*

joka ei sisällä kirjaimia. Loppuosan käyttäjä voi itse päättää, jonka ei tarvitse välttämättä olla juokseva numero. Alku- ja loppuosaan muotoon standardi ei anna määrityksiä. Alku- ja loppuosan yhteenlaskettu merkkien pituus täytyy olla alle 13 merkkiä. Data objektien (DO) ja attribuuttien (DA) niminä käytetään standardin määrittämiä nimiä, jotka määritetään niitä vastaavissa luokissa osissa 7-3 ja 7-4. Riippuen viittauksesta, näistä muodostuu loogisen noodin referenssi, data objektin referenssi ja data attribuutin referenssi. [6, s. 181–182] [5, s. 93–95]

Standardissa määritetään kaksi näkyvyysaluetta (engl. scope) viittaukselle, jotka ovat palvelin- ja looginen laite -näkyvyysalueet. Serverinäkyvyysalueelle viitataan ottamalla viittauksesta pois loogisen laitteen nimi. Eli kuvassa 4 viittaus tulisi muotoon "*/Q0XCBR1.Pos.Mag*". Edellemainittua viittausta käytetään silloin, kun loogisen noodin instanssi sijaitsee loogisen laitteen ulkopuolella, mutta kuitenkin palvelimelta. Looginen laite -näkyvyysalueessa viittaus sisältää loogisen laitteen nimen ennen kauttaviivaa, toisin kuin palvelin-näkyvyysalueessa. Esimerkiksi kuvassa 4 oleva viittaus "*OmaLD/Q0XCBR1.Pos.Mag*". Loogisen laitteen -näkyvyysaluetta käytetään silloin kun loogisen noodin instanssi sijaitsee loogisen laitteen sisällä sen hierarkiassa. Tässä työssä jatkossa käytetään pelkästään loogisen laitteen -näkyvyysaluetta. [6, s. 183]

Standardi määrittä maksimipituuksia viittauksille. Seuraavaksi kerrotut pituusmääritykset ovat voimassa kummallekin edelle mainitulle näkyvyysalueen viittaukselle. Ennen kauttaviivaa saa olla maksimissaan 64 merkkiä. Tämän jälkeen kauttaviiva, josta seuraa uudelleen maksimissaan 64 merkkiä. Eli koko viittauksen maksimipituus saa olla enintään 129 merkkiä, kauttaviiva mukaan lukien. [6, s. 183]

2.1.5 Attribuuttien funktionaalinen rajoitus ja niistä muodostetut datajoukot

Kirjoita tähän kuinka standardin mukaisia data settejä muodostetaan ja mitä ne ovat. Kirjoita tähän myös FCD ja FCDA lyhenteistä ja mitä ne tarkoittavat. Anna esimerkki kuinka data setit muodostuu vaikka kuvalla. Näitä tietoja tarvitaan kun käsitellään raportointilohkoja ja niiden tarkailevia data attribuutteja.

Kirjoita tähän myös funktionaalisista rajoitteista, mitä standardi määrittää data attribuuteille. Ja kuinka näitä käytetään FCD ja FCDA ryhmien muodostamiseen. Funktionaalisesta rajoitteesta ja FCD ja FCDA asioista on tietoa [6, s. 54–55].

Standardin yleiset luokat (CDC) määrittävät käytettävät data attribuutit. Luokat määrittää myös jokaiselle data attribuutille aikaisemmin mainittun funktionaalinen rajoitteen (engl. *functional constraint*, lyhennetään **FC**). Funktionaalinen rajoite kuvaa attribuutin käyttötarkoitusta ja sitä mitä palveluita attribuuttiin voidaan käyttää. Funktionaalinen rajoite voidaan ymmärtää niin sanottuna suodattimena data objektin data attribuuteille. Esimerkiksi kaikki attribuutit, jotka liittyvät laitteen tilaan (engl. status), niillä on funktionaalinen rajoite ST (standardissa engl. status information). Standardi määrittää paljon erilaisia funktionaalisia rajoitteita, jotka ovat kaikki kahden ison kirjaimen yhdistelmiä. Taulukossa 2 on esitetty joitain tärkeimpiä funktionaalisia rajoitteita. Funktionaalinen rajoite myös määrittä onko attribuutti kirjoitettava tai luettava.

Taulukko 2. *Osa IEC 61850 -standardin määrittämistä funktionaalisista rajoitteitteista (FC) [6, s. 54].*

Lyhenne	Selite	Luettava	Kirjoitettava
ST	Laitteen tilatieto (status)	Kyllä	Ei
MX	Mittaustieto (measurands)	Kyllä	Ei
CF	Laitteen asetusarvo (configuration)	Kyllä	Kyllä
DC	Selitystieto (description)	Kyllä	Kyllä

Funktionaalisia rajoitteita käytetään, kun IED-laitteeseen tehdään ACSI-palveluiden mukaisia kutsuja. Esimerkiksi jos halutaan lukea kuvassa 4 OmaLD/Q0XCBR1.Pos polussa olevan data objektin kaikki tilan arvot yhdellä kutsulla, käytettäisiin ST funktionaalista rajoitetta taulukosta 2. Kutsu jättää lukematta kaikki muut data objektin attribuutit.

Funktionaalista rajoitetta voidaan käyttää suodattamaan data attribuutteja data objektista ja niiden ali data objekteista. Toisin sanoen hierarkiassa referenssipisteestä alaspäin oleviin kentiin. Standardi määrittä lyhtenteen FCD (engl. Functional Constrained Data), jota käytetään silloin kun hierarkian ensimmäistä data objektia suodatetaan funktionaalisella rajoitteella. Aikaisemmin mainittu OmaLD/Q0XCBR1.Pos funktionaalisella rajoitteella ST, on FCD-suodatus. Tässä Pos on ensimmäinen data objekti Q0XCBR1 loogisen noodin jälkeen.

Kirjoita yllä olevaan kappaleeseen vielä FCDA määrityksestä ja sitten tämän jälkeen kuinka FCD ja FCDA määrityksiä käytetään datajoukkojen rakentamiseen IED-laitteeseen. Piirrä tähän myös kuva, josta tulee esille FCD:n ja FCDA:n erot ja kuinka se suodattaa.

2.1.6 Abstrakti kommunikointi ja ACSI

Voisiko tähän kirjoittaa miten mappaus johonkin tekniikkaan tapahtuu ja kuinka kommunikaatio sitten tapahtuu oikeasti. Samalla selittää ACSI-mallista jotain. Hyvä kuva standardissa löytyy tästä löytyy [5, s. 76]. Voisiko tämän heittää johonkin omana otsikkonaan?

Voisiko tähän myös laittaa kuva kuinka osat on pilkottu eri IED-laitteille ja niiden välinen kommunikoninti. Mallia [5, s. 31].

2.1.7 Viestien tilaus ja tilauksen konfigurointi

Kirjoita tähän IEC 61850 -standardin määrittästä abstraktista raportointimallista. Tätä raportointi mekanismia tullaan käyttämään raporttien tilauksessa ja sen konfigurointi täytyy ymmärtää toteuttettavan ohjelmiston kannalta. Käy läpi mitä asiakasohjelman täytyy tehdä ja mikä on tapahtumien järjestys jotta raportteja voidaan edes tilata. Hyvä kuva ja selitys missä järjestyksessä asiat tapahtuu asiakkaan ja palvelimen välillä. Ja yleistä tietoa raportoinnista löytyy [5, s. 40–44].

Kirjoita data attribuuttien liipaisimiin enemmän laadun muutoksesta mitä tarkoittaa ja viite sinne.

IEC 61850 -standardi määrittää erilaisia liipaisimia data attribuuteille, joita voidaan käyttää liipaisemaan jokin tapahtuma IED-laitteessa. Standardi määrittää seuraavia liipaisimia data attribuuteille:

- datan muutos (engl. data change, standardissa lyhenne *dchg*),
- laadun muutos (engl. quality change, standardissa lyhenne *qchg*), ja
- datan päivitys (engl. data update, standardissa lyhenne *dupd*).

Edellä mainituissa ero datan muutoksen ja päivityksen välillä on se, että datan päivitys liipaisee tapahtuman, vaikka attribuutin uusi arvo olisi sama. Datan muutos ei liipaise tapahtumaa, jos uusi arvo on sama kuin edellinen arvo. Laadun muutos tarkoittaa, että data attribuuttiin liitetty laatuarvo muuttuu. Laatuarvo kertoo, voiko attribuutin arvoon luottaa. [5, s. 90]

Standardi määrittää kaksi mahdollisesti liipaistavaa tapahtumaa IED-laitteessa, jotka ovat raportointi ja lokitus. Lokitus on IED-laiteessa tapahtuvien tapahtumien lokitusta myöhempää käyttöä ja tarkastelua varten. Esimerkiksi attribuutin arvon muutos. Raportointi on tapahtuma, jossa generoidaan viesti tapahtuman liipaisseista attribuuteista. Tämä viesti lähetetään niitä tilaaville asiakkaille. Jos tilaavaa asiakasta ei ole, viestiä ei generoida.

Standardi käyttää sanaa "raportti"(engl. ja standardissa report) näiden viestien kuvaamiseen. Kuitenkin tässä työssä on käytetty sanaa "viesti"tästä eteenpäin "raportin"sijaan. Tämä sen takia, koska suomenkielessä raportti-sana voi tarkoittaa lukijalle muuta merkitystä, kuin verkon yli asiakkaan ja palvelimen välistä viestiä. Tässä työssä keskitytään edelle mainittuihin viesteihin, ei lokitukseen. Ja lopullinen ohjelmisto nimenomaan käsitteli näitä viestejä.

Standardi määrittelee kaksi luokkaa viestien tilaamisen ja konfigurointiin. Luokat ovat puskuroitu viestintälohko (engl. *Buffered Report Control Block*, lyhennetään **BRCB**) ja ei puskuroitu lohko (engl. *Unbuffered Report Control Block*, lyhennetään **URCB**). Tekstissä kumpaakin luokkaan viitatessa käytetään lyhennettä **RCB**. Ainoa ero luokkien välillä on, että BRCB puskuroi viestejä jonkin aikaa yhteyden katkettua. Yhteyden palautuessa, se lähettää puskuroidut viestit järjestyksessä asiakkaalle. BRCB takaa viestien järjestyksen ja saatavuuden. URCB lähettää viestejä asiakkaalle ilman puskurointia. Yhteyden katketessa, viestit menetetään. IED-laitetta konfiguroitaessa, luokista tehdään instansseja asiakkaiden tarpeen mukaan. Standardi määrittää, että tilaavan asiakkaan on varattava yksi RCB-instanssi itselleen ja tänä aikana muut asiakkaat eivät voi kyseistä RCB:tä käyttää. Niinpä IED-laitteelle on määritettävä RCB-instansseja sen käyttötarkoitusten mukaan.

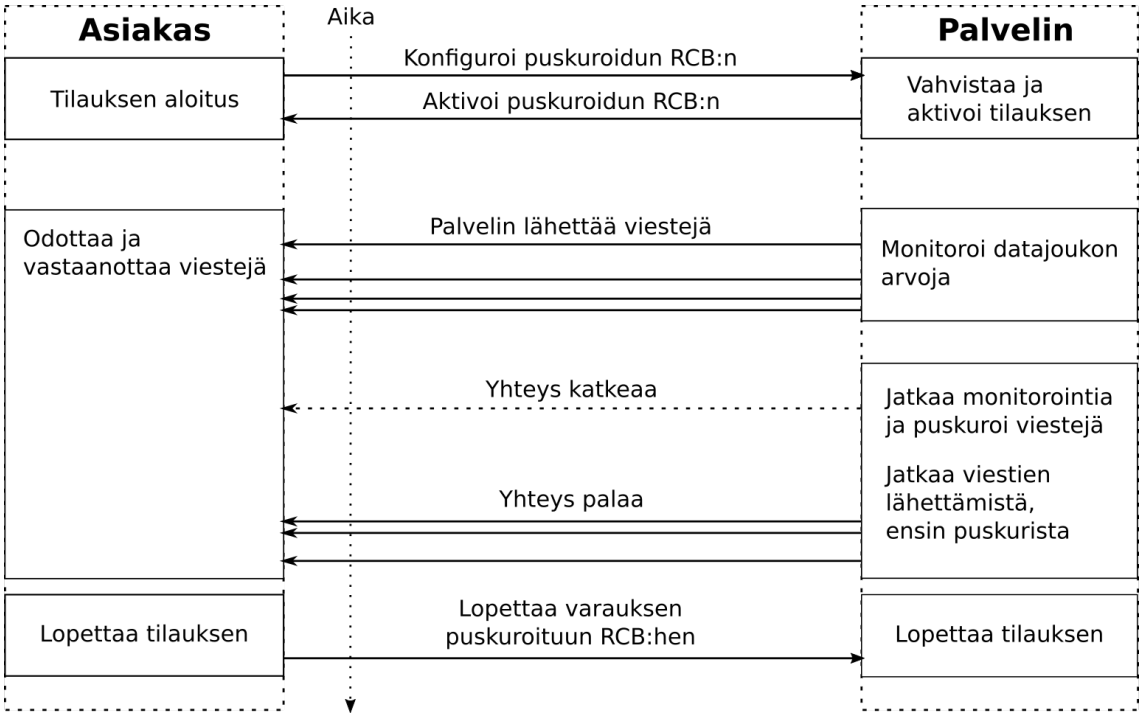
Jokainen RCB-instanssi kytketään johonkin muodostettuun datajoukkoon, jota se tarkkailee ja josta viestit generoidaan. Yhteen datajoukkoon voi olla kytkettynä monta RCB-instanssia. Jolloin yhden data attribuutin liipaistessa, jokainen siihen kytketty RCB generoi viestin asiakkaalle.

RCB-luokat sisältävät attribuutteja, joita asiakas konfiguroi ennen tilausta omien tarpeidensa mukaan. Tämän jälkeen asiakas varaa RCB:n kirjoittamalla konfiguroidut arvot ja asettamalla kentän RptEna arvoksi tosi (katso taulukko 3). Tämän jälkeen RCB on varattu kyseiselle asiakkaalle ja IED-laite aloittaa datajoukon attribuuttien tarkkailun. Asiakas jää odottamaan viestien tuloa palvelimelta ilman erillistä kyselyä. Jos konfiguroitu liipaisin liipaisee tapahtuman, RCB lähettää viestin asiakkaalle. Kuvassa 5 on esitetty yllämainittu prosessi asiakkaan ja palvelimen välillä käyttäen puskuroitua BRCB-luokkaa. Kuvassa yhteyden katketessa, palvelin puskuroi viestejä. Yhteyden palautuessa samalta asiakkaalta, palvelin lähettää viestit oikeassa järjestyksessä asiakkaalle. Tilaus lopetetaan asiakkaan pyynnöstä tai yhteyden ollessa poikki tarpeeksi kauan.

2.1.8 Raportointi-luokan määritys ja toiminta

Kirjoita tähän standardin määritämästä BRCB luokan rakenteesta ja mitä tietoa se sisältää. Mainitse että standardissa käytetään UTC-aikaa aikakentissä [5, s. 50].

BRCB-luokalla on erilaisia attribuutteja, joita asiakas voi kirjoittaa ja lukea ennen tilauksen aloittamista. Taulukossa 3 on esitetty standardin määrittämän BRCB-luokan attribuutit, attribuutin nimi englanniksi ja sen selite. Taulukossa ei ole esitetty attribuuttien tyyppejä, koska ne voi lukija tarvittaessa tarkemmin lukea standardin omasta määrityksestä. Ja



Kuva 5. *Puskuroitu viestien tilausprosessi asiakkaan ja palvelimen välillä.*

standardissa muutenkin kuvataan luokan eri attribuuttien toiminta paljon perusteellisemmin. Tässä työssä riittää että lukija ymmärtää luokan päätoiminnan hyvin. URCB-luokka on melkein samanalainen kuin taulukossa 3 määritetty BRCB-luokka. Tarkka määritys ja BRCB ja URCB luokkien erot löytyvät standardin osasta 7-2 [6, s. 93–118].

RCB-luokan TrgOps-attribuutti on binääritietue, jossa yksittäinen bitti ilmaisee mikä liipaisin voi aiheuttaa viestin lähettämisen. Asiakas voi päättää mitä liipaisimia haluaa käyttää. TrgOps sisältää seuraavat liipaisimet:

- datan muutos (engl. data change, standardissa lyhenne *dchg*),
- laadun muutos (engl. quality change, standardissa lyhenne *qchg*), ja
- datan päivitys (engl. data update, standardissa lyhenne *dupd*),
- yleinen kysely (enlg. *general-interrogation*, standardissa lyhenne GI), ja
- jatkuva viestintä väliajoin (engl. *intergrity*).

Kolme ensimmäistä dchg, qchg ja dupd ovat aikaisemmin määrittettyjen data attribuuttien liipaisimia. Asiakas voi tilata viestejä esimerkiksi vain data muutoksista ja ei muista. RCB-luokka määrittää data attribuuttien liipaisimien lisäksi vielä kaksi liipaisinta lisää, yleinen kysely ja jatkuva viestintä väliajoin. Yleinen kysely on viesti, johon RCB sisällyttää kaikki datajoukon attribuutit. Ja jonka asiakas voi liipaista asettamalla luokan attribuutin GI arvoksi tosi ja TrgOps attribuutissa liipaisin on päällä. Tällöin RCB käynnistää viestin generoinnin ja lähettää sen asiakkaalle. Jos liipaisin ei ole päällä TrgOps attribuutissa, ja GI arvoksi asetetaan tosi. RCB ei generoi viestiä. Viestin lähetyksen jälkeen RCB

Taulukko 3. *BRCB-luokan määritetyt attribuutit ja niiden selitteet [6, s. 94–103].*

Attribuutti	Englanniksi	Selite
BRCBName	BRCB name	Objektin nimi
BRCBRef	BRCB reference	Objektin referenssi
RptID	Report identifier	RCB-instanssin yksilöivä id lähetettyihin viesteihin, asiakas voi asettaa
RptEna	Report enable	Varaa RCB:n ja aloittaa viestien lähetyksen
DatSet	Data set reference	Tarkailtavan datajoukon referenssi
ConfRev	Configuration revision	Juokseva konfiguraation numerointi, muutos kasvattaa numerointia
OptFlds	Optional fields	Mitä optionaalisia kenttiä viestiin lisätään
BufTm	Buffer time	Puskurointiaika, ennen viestin lähetystä. Tänä aikana tapahtuvat liipaisut yhdistetään samaan viestiin
SqNum	Sequence number	Juokseva lähetetyn viestin numerointi
TrgOps	Trigger options	Millä liipaisimilla viesti lähetetään
IntgPd	Integrity period	Periodisen viestien väli millisekunteina, arvolla 0 ei käytössä
GI	General-interrogation	Käynnistää yleiskyselyn, joka sisältää kaikki datajoukon attribuutit seuraavaan viestiin
PurgeBuf	Purge buffer	Puhdistaa lähettämättömät viestit puskurista
EntryID	Entry identifier	Puskurissa olevan viimeisimmän viestin id. Arvo 0 tarkoittaa tyhjää puskuria
TimeOfEntry	Time of entry	Puskurissa olevan viimeisimmän viestin aikaleima
ResvTms	Reservation time	Varausaika millisekunteina, arvo -1 tarkoittaa konfiguraation aikaista varausta ja 0 että ei varausta
Owner	Owner	Yksilöi varaavan asiakkaan, yleensä IP-osoite tai IED-laitteen nimi. Arvo 0 että RCB on vapaa tai ei omistajaa

itse asettaa GI:n arvoksi epätosi. Jatkuva viestintä on viestin lähettäminen asiakkaalle tietyn väliajoin, johon sisältyy kaikki datajoukon attribuutit, kuten yleisessä kyselyssä. Toiminnon saa päälle kun asiakas asettaa RCB-luokassa attribuutit IntgPd arvoksi muu kuin 0, ja TrgOps attribuutin arvossa kyseinen liipaisin on päällä. Attribuutti IntgPd kertoo minkä väliajoin viesti generoidaan ja lähetetään asiakkaalle. Jos IntgPd arvo on muu kuin 0 ja TrgOps atribuutissa liipaisin ei ole päällä, ei viestiä generoida ja lähetetä asiakkaalle väliajoin.

Viestien tilaus aloitetaan kun asiakas kirjoittaa RptEna-attribuutin arvoksi tosi. Tilauksen aikana kirjoitus joihinkin attribuutteihin muuttuu, verrattuna ennen tilausta. Esimerkiksi yleisen kyselyn tekeminen on mahdollista tilauksen aikana kirjoittamalla GI:n arvoksi tosi. Tilauksen aikana kirjoittamalla TrgOps-attribuutin aiheuttaa puskurin tyhjentämisen. Ja Attribuutin OptFlds kirjoitus aiheuttaa epäonnistuneen vastauksen palvelimelta.

RCB-luokan attribuuttin OptFlds avulla asiakas voi asettaa mitä vaihtoehtoisia kenttiä viestiin sisällytetään. Attribuutin OptFlds on binääritietue, niin kuin ja TrgOps ja taulukossa 4 on esitetty sen asetettavat arvot [6, s. 98].

Taulukko 4. *RCB-luokan OptFlds attribuutin arvot ja niiden selitteet,*

Arvo	Selite
sequence-number	Jos tosi, sisällytä RCB-luokan attribuutti SqNum viestiin
report-time-stamp	Jos tosi, sisällytä RCB-luokan attribuutti TimeOfEntry viestiin
reason-for-inclusion	Jos tosi, sisällytä syy miksi arvo(t) sisällytettiin viestiin
data-set-name	Jos tosi, sisällytä RCB-luokan attribuutti DatSet viestiin
data-reference	Jos tosi, sisällytä datajoukon liipaisseen kohdan rakentamiseen käytetty FCD- tai FCDA-referenssi viestiin
buffer-overflow	Jos tosi, sisällytä viestiin tieto onko puskuri vuotanut yli kentällä BufOvfl (engl. buffer overflow)
entryID	Jos tosi, sisällytä RCB-luokan attribuutti EntryID viestiin
conf-revision	Jos tosi, sisällytä RCB-luokan attribuutti ConfRev viestiin

Kuinka attribuutit vaikuttavat viestin rakenteeseen ja mitä syitä arvon tai arvojen sisältymiseen viestissä voi olla, käsitellään seuraavassa kohdassa.

2.1.9 Viestin rakenne ja muoto

Kuvassa 6 on esitetty standardin määrittämän viestin rakenne ja kuinka optionaaliset kentät vaikuttavat viestin sisältöön [6, s. 104]. Kuvasta voi helposti nähdä mitä kohtia optionaaliset kentät viestiin lisäävät.

Viestin kenttiä SqNun, SubSqNum ja MoreSegmentsFollow käytetään kertomaan asiakkaalle, jos päätason viesti on liian pitkä ja se on pilkottu alaosiin. Kenttä SqNum on RCB-instanssin samanniminen kenttä ja on juokseva numerointi päätason viesteille. Kenttä SubSqNum on juokseva numerointi alkaen 0, jos päätason viesti, eli saman SqNum arvon sisältävä viesti on pilkottu osiin. Kentän MoreSegmentsFollow ollessa tosi asiakas tietää että päätason viesti on pilkottu osiin ja seuraava osa on odotettavissa palvelimelta. Kun viestin kaikki osat on lähetetty, palvelin asettaa viimeisessä viestissä kentän MoreSegmentsFollow arvoksi epätosi ja seuraavassa päätason viestissä SubSqNum kentän arvoksi 0. [6, s. 105–106]

Puskuroidun BRCB-instanssin puskurin täyttyessä viesteistä, esimerkiksi laiterajoitteesta johtuen. Asettaa RCB-instanssi seuraavaan viestiin kentän BufOvlf arvoksi tosi. Tästä kentästä asiakas voi päätellä onko tapahtunut tiedon menetystä. Kenttä sisällytetään viestiin vain jos RCB-instanssin OptFlds-attribuutissa on buffer-overflow bitti asetettu arvoksi tosi. [6, s. 106–107]

Tärkein rakenne viestistä on ymmärtää kuinka liipaissut datajoukon alko viestiin on lisätty. Yksi viesti voi sisältää 1:stä n:ään kappaletta alkioita. Tämä arvo riippuu onko RCB-instanssilla käytössä puskurointiaika BufTm. Tämän ajan sisällä liipaiseet datajoukon alkiot sisällytetään samaan viestiin. Jokainen sisällytetty alkio voi sisältää kentät DataRef tai ReasonCode. Jokaiselle alkiolle pakollinen tieto on Value. Tärkeä tieto Value kentästä on ymmärtää, että se voi sisältää yhden tai monta data attribuutin arvoa. Tämä riippuu

Viestin rakenteellinen sisältö		
Parametrin nimi	**Englanniksi**	**Selitys**
RptID	Report identifier	RCB-instanssin yksilöivä id.
OptFlds	Optional fields	Mitä optionaalisia kenttiä viestiin on sisällytetty
Jos sequence-number = tosi		
SqNum	Sequence number	Juokseva lähetetyn viestin numerointi
SubSqNum	Sub sequence number	Pilkotun viestin juokseva alinumerointi
MoreSegmentsFollow	More segments follow	Tosi jos samalla juoksevalla päänumerolla saapuu vielä lisää viestejä
Jos data-set-name = tosi		
DatSet	Data set	Tarkailtavan datajoukon referenssi
Jos buffer-overflow = tosi		
BufOvfl	Buffer overflow	Jos arvo on tosi, on bufferoidut viestit vuotaneet yli
Jos conf-revision = tosi		
ConfRev	Configure revision	Juokseva konfiguraation numerointi
Viestin data		
Jos report-time-stamp = tosi		
TimeOfEntry	Time of entry	Aikaleima milloin viesti generoitiin
Jos entryID = tosi		
EntryID	Entry id	Viestin yksilöivä numero
Liipaissut datajoukon alkio [1..n]		
Jos data-reference = tosi		
DataRef	Data reference	Liipaisseen datajoukon alkion FCD tai FCDA referenssi
Value	Value	Sisältää arvon tai arvot liipaisseesta datajoukon alkiosta
Jos reason-for-inclusion = tosi		
ReasonCode	Reason code	Syykoodi miksi tämä datajoukon kohta on sisällytetty viestiin

***Kuva 6**. Standardin määrittämä lähetetyn viestin rakenne.*

onko datajoukon liippaissut alkio FCD vai FCDA referenssi. Viittauksen ollessa FCDA-referenssi, joka sisältää vain yhden data-attribuutin. Sisältää Value-kenttä vain kyseisen data attribuutin arvon. Jos viittaus on FCD tai FCDA-referenssi, joka sisältää monta data attribuuttia. Sisältää Value-kenttä kaikki kyseiset arvot, vaikka niistä olisi liipaissut vain yksi attribuutti. Kuvassa X on esitetty malli kuinka sama liipaisu aiheuttaa eri viestin generoinnin kahteen eri RCB-instanssiin eri attribuuteilla Value-kentään. [5, s. 107–108]

Piirrä tähän kuva ottaen mallia [6, s. 108]. Ja selitä ylläolevaan viitaten kuinka Value kenttä sisältää eri arvoja FCD ja FCDA referenssistä riippuen.

Jokaisessa viestin datajoukon alkiossa oleva vaihtoehtoinen kenttä ReasonCode kertoo miksi alkio on sisällytetty viestiin. Kentä kertoo mikä RCB-instanssin TrgOps-attribuutilla asetetuista liipaisimista liipaisi tapahtuman ja aiheutti alkion sisällytyksen viestiin. Kentän arvot ovat suoraan verrattavissa TrgOps-attribuutin arvoihin. [6, s. 28–29]

2.2 Abstraktimallin sovitus MMS-protokollaan

Kirjoita kuinka ylempi ACSI sovitetaan MMS-protokollan palveluiksi ja tietotyypeiksi standardin IEC 61850-8-1 osuuden mukaan. Tähän myös miten raportointi toimii MMS-protokollan päällä.

2.2.1 MMS-protokolla

Selitä lyhyesti mikä on MMS-protokolla ja vähän sen tietotyypeistä. Tämän tarkoitus on pohjustaa tulevaa IEC 61850 abstraktien olioiden (ACSI) sovitusta tämän protokollan päälle.

2.3 Advanced Message Queuing Protocol

Kirjoita tähän AMQP määrittävästä standardista, mikä sen tarkoitus on ja mihin sitä voidaan käyttää.

2.3.1 Viestien välitysmekanismit

Mitä mekanismeja AMQP tarjoaa viestien välittämiseen osapuolille. Näitä on jono, reititys suoraan osapuolien välillä ja viestin julkaisu ja tilaaminen.

2.3.2 Tilaus ja julkaisu -mallin osat

Kirjoita tähän AMQP tarjoamista viestien julkaisu ja tilaus -mallin osista osapuolten kesken. Kerro mitä eri osat tekevät ja mikä niiden tehtävä viestien välittämisessä on. Englanniksi osia ovat esim. exchange, queue, publisher ja consumer.

3. ALKUTILANNE

Pohjusta miksi suunniteltava ohjelmisto tarvitaan toteuttaa yritykseen johon työn teen. Alustava suunniteltu ohjelmiston toteutus olisi tilata IEC 61850 -standardin määrittämiä raportteja ja muokata ne uuteen muotoon ja julkaista ne eteenpäin tilaavalle ohjelmalle käyttäen AMQP-standardin määrittämää viestintää. Jonon tilaava asiakasohjelmisto voi olla mikä tahansa muu ohjelmisto. Viestien lopullinen muoto voisi olla JSON.

Nykyisin sähköasemilla älykkäillä elektronisilla laitteilla (engl. *Intelligent Electronic Devices*, **IED**) asemilla voidaan toteuttaa tuhansia eri datapisteitä, jotka kuvaavat aseman toiminnallisuutta ja konfiguroitavuutta. Tämän konfiguroitavuuden ansiosta IED:tä voidaan asemalla käyttää erilaisina sähkölaitteina, kuten sulakkeina tai muuntajina. IEC 61850 -standardin abstraktit datamallit määrittävät IED:n datapisteiden rakenteet, muodot ja tyypit. Standardin mukaan erillisistä datapisteistä voidaan muodostaa haluttuja datajoukkoja (engl. *data set*). Datajoukkot ovat helppo tapa kuvata halutut tai tärkeät datapisteet yhdeksi yhteinäiseksi joukoksi. [10].

Asiakas-palvelin-arkkitehtuurissa asiakkaan on mahdollista tilata datajoukkojen sen hetkisiä arvoja IEC 61850 -standardin määrittäminä raportteina konfiguroitavilla parametreilla, jotka konfiguroidaan ennen tilausta. Arkkitehtuurissa asiakas tekee tilauksen palvelimelle (tässä tapauksessa IED), jonka jälkeen palvelin lähettää raportteja asiakkaalle automaattisesti, jonkin asiakkaan konfiguroiman ehdon täyttyessä. Standardi määrittää kuinka raporteja voidaan esim. välittää TCP/IP-protokollan avulla. Yksi raporti sisältää mm. tietojoukon sen hetkisiä arvoja ja syyn raportin lähetykseen (esim. arvon muuttuminen). Palvelin ylläpitää tilausta kunnes asiakas lopettaa tilauksen tai yhteys osapuolten välillä katkeaa. Asiakas tilaa raportit konfiguroimalla palvelimella olevan erillisen raportointilohkon (engl. *Report Control Block*, **RCB**). Lohkolla voi konfiguroida mm. raporttien sisältämiä vaihtoehtoisia kenttiä ja erilaisia liipaisimia raporttien generointiin. Standardi määrittää että yhdellä RCB:llä voi olla vai yksi tietojoukko ja yksi tietojoukko voi olla viitattuna monessa eri RCB:ssä. Yhdessä IED:ssä voi olla määrittetynä monta RCB:tä. Yhtä tilaavaa asiakasta kohden on yksi RCB instanssi. [6, s. 91–130].

Tulevissa kappaleissa pohjustetaan työn alussa olemassa olevan ohjelmiston arkkitehtuuria, mitkä olivat sen komponentit ja niiden toiminnallisuus. Tämän jälkeen pohditaan toteutuksen ongelmia, ja mitä työssä pyritään ratkaisemaan uudella toteutuksella. Asetettujen tutkimuskysymysten ja ongelmien kautta pyritään löytämään uudelle ohjelmiston arkkitehtuurille pohjaa ja ratkaisua siihen liittyviin päätöksiin.

3.1 Kokonaiskuva

Kirjoita tähän osioon kokonaiskuva alkutilanteesta missä oltiin ennen työn aloittamista. Selvennä kuvilla alkutilanteen arkkitehtuuria.

Työn aloitusvaiheessa oli jo toteutettuna ohjelmisto raporttien tilaukseen ja käsittelyyn. Tämä toteutus oli puutteellinen, ei helposti skaalautuva, ja huono suorituskyvyltään. Alkuperäinen ohjelmisto oli lähellä enemmän ensimmäistä prototyyppiä ennen todellista toteutusta. Työn tarkoituksena oli suunnitella ja toteuttaa uusi toteutus, joka ratkaisisi entisen ongelmakohdat.

Alkuperäisessä toteutuksessa asiakasohjelmisto oli toteutettu Ruby-ohjelmointikielellä. IEC 61850 -standardin määrittämien palveluiden ja tietorakenteiden toteutukseen käytettiin avoimen lähdekoodin libIEC61850-kirjastoa[1]. Kirjasto on ohjelmoitu C-kielellä ja sen avulla voidaan toteuttaa kumpikin palvelin- ja asiakasohjelmisto. Tässä toteutuksessa tarvittiin vain asiakasohjelman osuutta Ruby-osuuden toteutukseen. Kirjasto abstraktoi standardin määrittämiä palveluita ja tietorakenteita ohjelmoijalle helpoiksi funktioiksi ja C-kielen rakenteiksi. Normaalisti C-koodin funktioiden kutsuminen Rubysta suoraan ei ole mahdollista ilman erillistä liitosta. Seurauksena Rubyyn oli tehty laajennos libIEC61850-kirjastoon käyttäen Rubylle saatavaa ruby-ffi -kirjastoa[2] (engl. *Foreign Function Interface*, **FFI**). Liitoksen avulla libIEC61850-kirjasto voi hoitaa standardin vaatiman matalan tason toiminnan ja Ruby-ohjelmisto voi keskittyä vaadittuun toiminnallisuuteen.

Kirjasto toteuttaa raporttien vastaanoton palvelimelta erillisellä säikeellä. Säie käynnistetään kun asiakasohjelma asettaa funktion takaisinkutsuntaa varten raportin saapuessa ja aloittaa tilauksen. Asetettua funktiota kutsutaan asynkronisesti erillisestä säikeestä raportin saapuessa asiakkaalle. Takaisinkutsun suorituksen jälkeen, suoritus palaa takaisin säikeeseen.

3.2 Ratkaistavat ongelmat

Kirjoita tähän mitä ongelmia edellisen toteutuksen kanssa on ja mitä yritään ratkoa. Mainitse suorituskykyongelmista Rubylla ja libiec61850-kirjastoa käyttäen.

Työn alussa olevan ohjelmiston ongelmia oli mm. ei helposti skaalautuvuus, huono suorityskyky raporttien määrän ollessa suuri, eikä ohjelmisto tukenut kaikkia standardin määrittämiä toiminnallisuuksia. Ohjelmistoa voisi enemmän pitää ensimmäisen toteuksen prototyyppinä. Ohjelman suoritusalustana käytettiin Linuxia.

Ohjelmisto pystyi tilaamaan ja vastaanottamaan raportteja yhdeltä IED:ltä ja siinä monelta määritellyltä RCB:ltä. Ohjelmisto prosessoi ja tallensi raportteja tietokantaan muuta

[1] http://libiec61850.com
[2] https://github.com/ffi/ffi

käyttöä varten. Tilanteessa, jossa raportteja tilaavassa järjestelmässä on monta osaa, jotka kaikki tarvitsevat raporttien tietoja reaaliajassa. Joutuvat eri osat tässä tilanteessa kyselemään tietoja tietokannasta, ilman erillistä tietoa niiden saapumisesta. Tämä aiheuttaa turhaa kuormaa tietokannalle ja tietojen saaminen reaaliajassa ei ole mahdollista. Myöskin jos komponentti tarvitsee tietyn tyypin raportteja, ei kaikkea tietoa, ongelma on sama.

Ohjelmiston suorituskyky paikoin raporttien määrän ollessa suuri aiheutti ongelmia. Syynä Rubyn toteutuksessa oli oletustulkissa (*CRuby*) oleva globaali lukitus (engl. *Global Interpreter Lock*, **GIL**). Vaikka Rubyn säie on oma käyttöjärjestelmän tarjoama säie, GIL estää säikeiden yhtäaikaisen suorituksen ja vain yksi säie on suorituksessa kerrallaan [12, s. 131–133]. Linux-pohjaisella käyttöjärjestelmällä libIEC61850-kirjaston laitteistoabstraktiokerros (engl. *Hardware Abstraction Layer*, **HAL**) käyttää POSIX-säikeitä [13]. Linux-käyttöjärjestelmän säikeet ovat suorituksessa yhtä aikaa ja moniytimisellä prosessoreilla asioita tapahtuu samalla ajan hetkellä. Nyt raportin saapuessa, C-prosessin säikeen suoritus kutsuu takaisinkutsuntaan asetettua funktiota, joka on implementoitu Rubyn puolella. On funktion suoritus GILin alaista suoritusta. Ruby-prosessin myös suorittaessa muuta toimintaa takaisinkutsujen välissä, on Rubyn suorituskyky ohjelmiston pullonkaulana raporttien määrän ollessa tiheää.

Kirjoita tähän vielä ongelmasta kun tilataan monta RCB:tä. Raporttien tullessa Rubyn puolelle, ei Rubyn muu koodi saa tilattua loppuja RCB:tä kirjaston lukitusten takia. Ja yhteys aikakatkeaa tämän takia. Selitä lukituksista tarkemmin ja myös liitä pätkiä libIEC61850-kirjaston koodista. Syyn selityksen voi siirtää muualle. Kirjoittaa vain että on ongelma, ja selvitys miksi, muualla.

3.3 Tutkimuskysymykset

Esitä tässä työlle asetettuja tutkimuskysymyksiä. Näitä voisi olla esim. seuraavat:

- *Mikä on syynä huonoon suorituskykyyn alkutilanteen toteutuksella?*
- *Kuinka suorituskyky paremmaksi verrattuna nykyiseen toteutukseen?*
- *Mitkä ohjelmiston arkkitehtuurin suunnittelumallit (design patterns) olisivat sopivia tämän kaltaisen ongelman ratkaisemiseen? Mitä niistä pitäisi käyttää ja mitä ei?*
- *Mikä olisi sopiva lopullisen prosessoidun tiedon muoto?*
- *Kuinka järjestelmä hajautetaan niin että tiedon siirto eri osapuolten välillä on mahdollista ja joustavaa (push vs pull, message queue jne.)?*

4. SUUNNITTELU

Kirjoittaa tähän kuinka toteutettava arkkitehtuuri suunniteltiin ja kuinka päätöksiin päädytiin. Kirjoitusta myös miten tilattuja raportteja käsitellään ja kuinka niitä julkaistaan eteenpäin. Tarkoituksena olisi saada raportit nykyaikaiseen JSON muotoon.

4.1 Suorituskyvyn parantaminen

Miksi entisen toteutuksen suorituskyky on huono ja mitä voitaisiin tehdä sen parantamiseksi. Kirjoita vaikutuksista tähän ja mihin tuloksiin päädyttiin.

4.2 Järjestelmän hajautus

Lähde erilaisista hajautukista (pull vs pull, message queue) ja päätä mikä sopii tähän toteutukseen parhaiten ja miksi.

4.3 Ohjelmiston parametrisointi

Kirjoita mitä asiakasohjelman pitää tehdä jotta raportit saadaan tilattua ja mitä parametrejä ohjelmisto tarvitsee toimiakseen. Kuinka käyttäjä kontrolloi ohjelman eri asetuksia?

4.4 Arkkitehtuurin suunnittelu

Määritä ohjelman tarkempaa arkkitehtuuria mitä voidaan käyttää asetettujen ja yllämainittujen asioiden saavuttamiseen ja tarkentamiseen. Mitä jos käyttäjä tilaa monta viestiblokkia, niin missä järjestykssä asiat tehdään jne.

4.5 Prosessoidun viestin muoto

Kirjoita tähän mihin muotoon viestit lopussa tallennetaan esim. JSON. Miksi tähän valintaan päädyttiin. Kerro myös kuinka raportin alkuperäistä rakennetta muokattiin uuteen muotoon sopivaksi.

5. TOTEUTUS

Kirjoita tähän osioon siitä kuinka suunniteltu arkkitehtuuri toteutettiin ja millä tekniikoilla. Tämä osio käyttää lyhyitä koodiesimerkkejä hyväkseen selittämään lukijalle kuinka toteutus tehtiin, jotta lukija voisi itse toteuttaa samanlaisen ohjelmiston.

5.1 Ohjelmiston toteutuksen valinta

Kirjoita tähän miksi päädyttiin tietynlaiseen ohjelmiston toteuttamiseen. Työssä on mietitty komentorivipohjaista toteutusta. Lisäksi mille alustalle ohjelmisto suunnitellaan Windows vai Linux.

5.2 Kielen valinta

Kirjoita tähän mikä kieli valittiin toteutuksen tekemiseen ja miksi tämä. Alustava suunnitelma on toteuttaa C-kielellä.

5.3 RabbitMQ

Kirjoita tähän RabbitMQ toteutuksesta. Kirjasto toteuttaa AMQP-standardin määrittämiä eri viestintämalleja. Kerro kuinka sitä hyödynnetään tässä työssä ja vähän sen että mitä vaatii.

5.4 Käytettävät kirjastot

Kirjoita tähän erilaisista kirjastoista mitä toteutukseen valittiin ja miksi. Alaotsikoita voi lisätä jos toteutukseen tarvitaan muita kirjastoja.

5.4.1 libiec61850

IEC 61850 -standardin toteuttava C-kirjasto joka tekee raskaan työn standardin määrittämien palveluiden toteuttamiseen ja muodostamiseen. Kirjasto tarjoaa rajapinnat serveri- ja asiakasohjelmiston toteuttamiseen, mutta vain asiakasohjelmiston rajapintoja käytetään. Kirjasto tarjoaa myös rajapinnat haluttujen raporttien tilaamista varten. Kirjaston nettisivu täältä: http://libiec61850.com/libiec61850/.

5.4.2 rabbitmq-c

RabbitMQ:n rajapinnan toteuttava kirjasto C-kielen ohjelmille. Kirjastolla voidaan toteuttaa julkaisevia ja tilaavia ohjelmistoja. Kirjastosta käytetään julkaisevan puolen toteutusta. Kirjasto löytyy täältä: https://github.com/alanxz/rabbitmq-c.

5.4.3 JSON-formatointi

Joku kirjasto JSON formatointiin C-kielelle. Näkyy olevan parikin vaihtoehtoa. Perustele tähän valinta ja miksi.

5.5 Jatkokehitys

Kirjoita tähän ideoita mitä jää jatkokehitykseen ja mitä ohjelmistossa on puutteita tai mitä jäi tekemättä.

6. ARVIOINTI

Kirjoitta tähän arviota työn tuloksista.

7. TULOKSET

Kirjoita tähän lopputuloksen analysoinnista ja peilaa saatuja tuloksia työlle alussa asetettuihin kysymyksiin. Mitä jäi saavuttamatta, mitä saavutettiin ja miten hyvin? Mitä olisi voinut parantaa? Voi jakaa aliotsikoihin jos tarvetta.

8. YHTEENVETO

Kirjoita tähän ensin arviointi ja yhteenveto työstä ja sen lopputuloksista. Mitä hyötyjä työnantaja työstä saa ja jatkokehitysideoita. Mitä työssä meni hyvin ja mitä olisi voinut tehdä toisin?

Kommentteja työtä aloittaessa:

- *Olisiko hyvä, että lähdet työssäsi erilaisista hajautus paradigmoista (push vs pull; message queue), perustelet valintasi ja sitten menet suunnitteluun ja toteutukseen?*
- *Ja olisi hyvä, että työ perustelee miksi tuota MQ arkkitehtuuria yleensä (ja rabbitMQ:ta) käytetään.*

Things to do now:

- *Laittaa aihe hyväksyntään.*
- *Lähde kirjoittamaan teoriaa ja ennen sitä yleistä tasoa missä ollaan. Yleinen korkea taso sen takia, että lukija ymmärtää mistä edes on kyse. Pidä koko ajan kirjoittaessa mielessä top-down lähestymistapa! Erittäin tärkeä!!!*
- *Loppu otsikoida niin että ensin on tulokset, niiden arviointi ja yhteenveto mainitussa järjestyksessä.*
- *Kirjoittaessa miettiä asioita mistä kirjoitetaan ja pitää kontekstista kiinni.*
- *Pidä lauseet simppelineinä ja helppolukuisina! Älä turhaan vaikeuta hommaa lukijalle ja se ei tuo työhön yhtään mitään lisäarvoa! Todella tärkeä asia ajatella! Jos lause käsittää monta asiaa, pilko se pienempiin erillisiin lauseisiin.*
- *Muihinkin lähteisiin voi viitata kuin tieteellisiin. Toki yritä löytää tieteellisiä julkaisuja mahdollisuuksien mukaan. Osoittaa että olet perehtynyt asiaan paremmin.*
- *Kun kirjoitat asiaa esim. että entisessä ohjelmassa oli ongelma että ei skaalaudu helposti tai on huono suorityskyky. Kerro mistä johtopäätös tulee. Tämä ei ole lukijalle selvää tietoa.*
- *Teorien ja yleisen osuuden kirjoittamisen jälkeen, sovi palaveri Karin kanssa.*
- *Työn otsikko on hyvä, ei tarvitse olla erikseen "ohjelmallisesti"sanaa.*
- *Työn päätason otsikoita laittaa enemmän kuvaavimmiksi kuin "Alkutilanne"ja "Teoria".*
- *Käytä työssä viesti sanaa raportin sijaan. Tuo lukijalle esille että se tarkoittaa standardin mukaisia raportteja.*

Huomioituja asioita toisten dipoissa:

- *Tärkeät sanat esitellään tekstissä ensimmäisen kerran kursiivilla painottamisen takia. Tämän jälkeen ei enää samaa sanaa kursivoida.*
- *Todella paljon erilaisia lähteitä käytetty! Blogiposteja, kirjoja, ja tapahtumien kirjoituksia (IEEE). On myös nettisivuja käytetty lähteenä kun mainitaan esim. Git ja jotain muita sivuja. Nämä tietysti voi olla myös alaliitteenä sivulla.*
- *Tosi hyvin kirjoitettu! Todella selkeää tekstiä ja etenee hyvin ja on lukijalle ystävällinen.*

- *Johdanto on pilkottu otsikoihin työn alkutilanteen selvittämiseksi hyvin ennen teoriaa. Ja teoriaosuus alkaa joustavasti johdannon jälkeen järkevästi.*
- *Kun listataan tekstiä, sana on ensin* kursiivilla *ja on selitetty asiaa. Kohta loppuu puolipisteeseen (;). Tämän jälkeen jatkuu pienellä seuraava aihe ja päättyy myös puolipisteeseen. Viimeinen kohta alkaa myös pienellä, mutta päättyy pisteeseen normaalisti. Seuraava kappale alkaa normaalisti. Listassa lauseet muokkautuvat yhteen esim. käyttäen ja sanaa.*
- *Kysymys teoriassa mikä työssä on jäljellä oli kirjoitettu* kursiivilla*.*

Palautetta Hannulta: Tiivistelmään ottaa mallia tutki ja kirjoita kirjasta. Siinä on hyvin mainittu mitä tiivistelmän pitää sisältää ja pysyä aiheessa. Lainaa tämä kirja kirjastosta. Älä jaarittele itsetäänselvyyksiä tiivistelmässä. Esim. kaksi ekaa lausetta on tuhlattu jo saman asian sanomiseen. Saako "sähköntuotantolaitoksista, sähkölinjoista ja sähköasemista"yhdistettyä jotenkin, että ei toista samaa sanaa kolme kertaa? Avainsanoja käyttää tiivistelmässä. Johdannosta pois turha puhuminen itsestäänselvyyksistä. Esim. "Nykypäivänä sähköverkot ovat iso yhteiskuntaa ja sen sujuvaa toimivuutta. Ilman sähköä ei moni asia nykypäivänä toimisi niinkuin se on. Sähköä tarvitaan joka paikassa ja tietotekniikan lisääntyessä vieläkin enemmän."nämä lauseet ovan vähän turhia. Mene suoraan aiheeseen.

LÄHTEET

[1] C. Brunner, IEC 61850 for power system communication, teoksessa: 2008 IEEE/-PES Transmission and Distribution Conference and Exposition, April, 2008, s. 1–6.

[2] B. E. M. Camachi, O. Chenaru, L. Ichim, D. Popescu, A practical approach to IEC 61850 standard for automation, protection and control of substations, teoksessa: 2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), June, 2017, s. 1–6.

[3] IEC 61850-1 Communication networks and systems for power utility automation – Part 1: Introduction and overview, International Electrotechnical Commission, International Standard, Mar. 2013, 73 p. Saatavissa (viitattu 15.6.2018): https://webstore.iec.ch/publication/6007

[4] IEC 61850-6 Communication networks and systems for power utility automation – Part 6: Configuration description language for communication in electrical substations related to IEDs, International Electrotechnical Commission, International Standard, Dec. 2009, 215 p. Saatavissa: https://webstore.iec.ch/publication/6013

[5] IEC 61850-7-1 Communication networks and systems in substations - Part 7-1: Basic communication structure for substation and feeder equipment - Principles and models, International Electrotechnical Commission, International Standard, July 2003, 110 p. Saatavissa (viitattu 16.5.2018): https://webstore.iec.ch/publication/20077

[6] IEC 61850-7-2 Communication networks and systems for power utility automation - Part 7-2: Basic information and communication structure - Abstract communication service interface (ACSI), International Electrotechnical Commission, International Standard, Aug. 2010, 213 p. Saatavissa (viitattu 16.5.2018): https://webstore.iec.ch/publication/6015

[7] IEC 61850-7-4 Communication networks and systems for power utility automation - Part 7-4: Basic communication structure - Compatible logical node classes and data object classes, International Standard, Mar. 2010, 179 p. Saatavissa (viitattu 16.5.2018): https://webstore.iec.ch/publication/6017

[8] IEC 61850:2018 SER Series, International Electrotechnical Commission, verkkosivu. Saatavissa (viitattu 9.6.2018): https://webstore.iec.ch/publication/6028

[9] K. Kaneda, S. Tamura, N. Fujiyama, Y. Arata, H. Ito, IEC61850 based Substation Automation System, teoksessa: 2008 Joint International Conference on Power System Technology and IEEE Power India Conference, Oct, 2008, s. 1–8.

[10] R. E. Mackiewicz, Overview of IEC 61850 and Benefits, teoksessa: 2006 IEEE PES Power Systems Conference and Exposition, Oct, 2006, s. 623–630.

[11] New documents by IEC TC 57. Saatavissa (viitattu 9.6.2018): http://digitalsubstation.com/en/2016/12/24/new-documents-by-iec-tc-57/

[12] R. Odaira, J.G. Castanos, H. Tomari, Eliminating Global Interpreter Locks in Ruby Through Hardware Transactional Memory, SIGPLAN Not., Vol. 49, Iss. 8, Feb. 2014, pp. 131–142. Saatavissa (viitattu 16.5.2018): http://doi.acm.org/10.1145/2692916.2555247

[13] Official repository for libIEC61850, the open-source library for the IEC 61850 protocols http://libiec61850.com/libiec61850, GitHub verkkosivu. Saatavissa (viitattu 17.5.2018): https://github.com/mz-automation/libiec61850

[14] B.D. Stockton, Design Guide for Rural Substations, United States Department of Agriculture, June 2001, 763 p. Saatavissa (viitattu 25.5.2018): https://www.rd.usda.gov/files/UEP_Bulletin_1724E-300.pdf