TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

VILLE JOKELA

SÄHKÖASEMADATAN LAAJAMITTAINEN ANALYSOINTI

Diplomityö

Tarkastaja: Professori Hannu-Matti Järvinen
Tarkastaja ja aihe hyväksytty Tieto- ja sähkötekniikan tiedekunta-neuvoston kokouksessa 3. syyskuuta 2014.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma
JOKELA, VILLE: Sähköasemadatan laajamittainen analysointi
Diplomityö, 62 sivua
Marraskuu 2014
Pääaine: Pervasive Systems
Tarkastaja: Professori Hannu-Matti Järvinen
Avainsanat: automaattinen analysointi, häiriötallenne, konfigurointi, sähköasemadata

Teknologian jatkuva kehittyminen ja laitteiden digitalisoituminen ovat lisänneet tuotettavan datan määrää. Sama on havaittavissa myös sähköasemalla, jossa nykyaikaiset suojalaitteet kykenevät tuottamaan niin normaalitilanteeseen liittyvää prosessidataa kuin vikatilannekohtaisia häiriötallenteitakin. Kerättävästä datasta saadaan merkittävää liiketoiminnallista hyötyä, kun sitä jäsennellään ja prosessoidaan automaattisesti. Samalla avautuu uusia liiketoimintamahdollisuuksia, kun analyysipalveluita voidaan myydä esimerkiksi pilvipalveluiden muodossa.

Tämä opinnäytetyö liittyy ABB:llä kehitettyyn OffSide-ohjelmistoalustaan, jolla sähköasemadatalle voidaan ajaa erityisiä analyysiohjelmia. Työn päätavoitteena oli selvittää, miten sähköasemadataa saataisiin analysoitua mahdollisimman kattavasti kyseisellä alustalla. Analysoitavasta datasta keskityttiin sähköasemalta saataviin häiriötallenteisiin. Tähän liittyen työn tavoitteisiin kuului myös ohjelmistoalustaan liittyvän konfigurointityökalun vaatimuksien kerääminen, työkalun toteuttaminen sekä testaaminen. Opinnäytetyö toteutettiin osana CLEEN Oy:n hallinnoimaa Älykkäät sähköverkot ja energiamarkkinat -tutkimusohjelmaa (SGEM) ja Suomen valtion innovaatiorahoituskeskus Tekesin rahoituksella.

Kirjallisuusosiossa esitellään syyt, miksi häiriötallenteiden analysointi kannattaa automatisoida sekä miten sähköasemadatan analysointi sopii myös tulevaisuuden älykkäisiin sähköasemiin. Kirjallisuusosiossa käydään myös läpi sähköasemadataan liittyvät standardit IEC 61850 ja IEEE C37.111-1999 (COMTRADE). Tutkimusosassa puolestaan esitellään OffSide-ohjelmistoalusta sekä siihen läheisesti liittyvä PCM600-konfigurointityökalu sekä RTDB-tietokannan sisältävä cpmPlus-tuotannonhallintaohjelmisto. Tutkimusosassa käydään läpi myös sähköasemadatan analysointiin ja sen visualisointiin liittyvä käytännön pilottihanke, jonka avulla konfigurointityökalulle määriteltiin vaatimukset.

Tutkimustuloksena esitellään konfigurointityökalun toteutus, joka liittyy osaksi OffSiden Connectivity Package -laajennusta. Tuloksissa kuvataan myös OffSiden ohjelmistoon tehdyt muut muutokset, joiden avulla konfigurointi oli mahdollista. Tuloksia arvioitiin testaamalla järjestelmää käytännössä. Järjestelmän testaaminen osoitti sähköasemadatan analysoinnin luomat mahdollisuudet käyttökelpoisiksi. Toteutuksen yhteydessä havaittiin kuitenkin joitain rajoituksia ohjelmistoalustassa sekä merkittävät määrät jatkokehitysideoita.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Signal Processing and Communications Engineering
JOKELA, VILLE: Broad Analysis of Data from Electrical Substation
Master of Science Thesis, 62 pages
November 2014
Major: Pervasive Systems
Examiner: Professor Hannu-Matti Järvinen
Keywords: Automated analysis, Configuration, Disturbance record, Electrical substation data

Continuous development of technology and digitalization of devices have increased the amount of available data. This same progress is happening in electrical substation, where modern Intelligent Electronic Devices can produce process data related to normal operations and disturbance records. Collected data can provide significant economic benefits when it is structured and processed automatically. There can also be new business opportunities when results of analysis can be provided, e.g., as a cloud service.

This thesis is related to ABB's OffSide software platform, which is capable of running analysis applications for electrical substation data. The main objective of this thesis was to figure out how data from electrical substation could be analyzed with OffSide as broadly as possible. Related to this, gathering requirements for a configuration tool, developing that tool and testing the tool were also objectives of the thesis. The focus of analysis was on disturbance records. This work was carried out in the Smart Grids and Energy Markets (SGEM) research program coordinated by CLEEN Ltd. with funding from the Finnish Funding Agency for Technology and Innovation, Tekes.

The reasons why analysis of disturbance records should be automated and why analysis of electrical substation data is important part of future smart substations are described in the literature part of the thesis. The literature part also presents two standards related to electrical substation data: IEC 61850 and IEEE C37.111-1999 (COMTRADE). The practical part presents OffSide platform and two related software: PCM600 configuration tool and cpmPlus collaborative production management software which includes RTDB history database. A pilot project related to analysis of electrical substation data and visualization of analysis results is also described. In that part requirements for the configuration tool were gathered.

As the result of the thesis the implementation of the configuration tool, which is part of OffSide's Connectivity Package extension, is presented. All the changes in OffSide's software which made the configuration possible are described together with it. The results were evaluated by testing the system in practice. The testing indicated that possibilities created by analyzing electrical substation data are usable. However, some limitations in the software platform together with vast amount of further development ideas were found.

ALKUSANAT

Tämä diplomityö tehtiin ABB Oy:n Medium Voltage Products -yksikölle Tampereella. Haluan kiittää jokaista henkilöä, joka oli jollain tavalla mukana projektissa. Erityisesti haluan kiittää Erkka Kettusta ja Jani Valtaria työn ohjaamisesta koko projektin ajan. Opinnäytetyön tarkastaja professori Hannu-Matti Järvinen Tampereen teknillisestä yliopistosta ansaitsee myös kiitokset. Lisäksi kiitokset kuuluvat läheisilleni, jotka ovat tukeneet minua niin opintojen kuin diplomityön tekemisenkin aikana.

Tampereella 18. marraskuuta 2014

_____

Ville Jokela

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

ACT	Application Configuration Tool, sovelluksen konfigurointityökalu. PCM:n työkalu, jolla ohjelmia voidaan luoda ja muokata graafisesti.
ACSI	Abstract Communication Service Interface, abstrakti kommunikaatiopalvelurajapinta. IEC 61850:n standardoima kommunikaatiorajapinta.
API	Application Programming Interface. Ohjelmointirajapinta, joka määrittelee, miten ohjelmistokomponenttien tulisi vaikuttaa keskenään.
BAT	.BAT on tiedostopääte komentojonoille, joita voidaan ajaa komentorivin kautta Windowsissa.
CDS	Configuration Data Server, konfiguraatiodatapalvelin. PCM:n työkalu, joka säilyttää tiedot liittyen IED:n funktiomalliin.
ClioNet	ABB:n kehittämä suojareleille tarkoitettu tiedonjako- ja varmuuskopiointipalvelu.
COMTRADE	Common format for Transient Data Exchange for power systems. IEEE:n standardi häiriötallenteiden tiedostoformaateille.
ConnPack	Connectivity Package, liitettävyyspaketti. PCM600:n laajennus, jolla mallinnetaan laitteen ominaisuuksia ja toimintaa.
cpmPlus	ABB:llä kehitetty tuotannonhallintaohjelmisto, joka sisältää RTDB-tietokannan.
CRW	Common Read/Write tool, PCM:n työkalu, joka hoitaa tietojen kirjoittamisen PCM:n ja IED:n välillä.
DLL	Dynamic-link library. Microsoftin kehittämä tiedostotyyppi jaetuille ohjelmakirjastoille.
GUI	Graphical User Interface, graafinen käyttöliittymä.
IED	Intelligent Electrical Device, sähköverkon suojalaite.
JavaScript	Web-ympäristössä käytettävä dynaaminen komentosarjakieli.
MVVM	Model View ViewModel. Ohjelmistosunnittelumalli, joka mahdollistaa graafisen käyttöliittymän kehittämisen erillään varsinaisesta datamallista.
.NET	Microsoftin kehittämä ohjelmistokomponenttikirjasto.
OffSide	ABB:llä kehitetty sovellusalusta, jolla voidaan suorittaa analyysiohjelmia sähköasemadatalle.
OPC	Open Platform Communications. Standardi avoimelle tiedonsiirrolle teollisuuden automaatiosovelluksissa.

OPC AE	OPC Alarms and Events. Ryhmä standardeja hälytysten ja tapahtumien välittämiseen automaatiosovelluksissa.
OPC DA	OPC Data Access. Ryhmitelmä standardeja reaaliaikaisen datan välittämiseen automaatiosovelluksissa.
OPC HDA	OPC Historical Data Access. Ryhmitelmä standardeja historiadatan välittämiseen automaatiosovelluksissa.
OPC UA	OPC Unified Architecture. Teollisuuden laitteiden välinen kommunikaatioprotokolla.
Perl	Tulkattava skriptimäinen ohjelmointikieli.
PCM600	ABB:llä kehitetty työkalu suojareleiden konfiguroimiseen.
PDS	Parameter Data Server, parametridatapalvelin. PCM:n työkalu, jonka vastuulla on IED:n tyyppidatan sekä parametrien hallinta.
PST	Parameter Setting Tool, parametrien asettelutyökalu. PCM:n työkalu, jolla laitteiden ja ohjelmien parametreja voidaan asetella.
Regexp	Regular expression, säännöllinen lauseke. Säännölliset lausekkeet ovat kuvauksia merkkijonojen rakenteista.
RTDB	Real Time Database. ABB:n kehittämä relaatiotietokanta, joka on suunniteltu teollisuusprosessien tiedon hallintaan ja historiatietojen tallentamiseen.
SCADA	Supervisory Control and Data Acquisition. Valvomojärjestelmä, jossa on graafinen käyttöliittymä automaatiojärjestelmiin.
SGEM	Smart Grids and Energy Markets, älykkäät sähköverkot ja energiamarkkinat. Suomalainen tutkimusohjelma, jonka tavoitteena on kehittää kansainvälisiä älysähköverkkoratkaisuja.
Singleton	Ohjelmistojen suunnittelumalli, jonka mukaisesta luokasta on saatavilla aina vaan yksi instanssi.
SQL	Structured Query Language, standardoitu kyselykieli, jolla relaatiotietokantaa voidaan käsitellä.
VtrinLib	CpmPlus-tuotannonhallintaohjelmiston tarjoama abstraktiokerros, joka tarjoaa olioperustaisen pääsyn datalähteeseen, kuten RTDB-tietokantaan.
Web socket	Protokolla samanaikaisen kommunikaation mahdollistamiseksi molempiin suuntiin yhdessä TCP-yhteydessä.
WPF	Windows Presentation Foundation. Windows Vistan ja sitä uudempien Windows-versioiden graafinen rajapinta .NET-sovelluskehyksessä.
XML	Extensible Markup Language, rakenteellinen merkintäkieli.

1 JOHDANTO

Teknologian jatkuva kehittyminen ja laitteiden digitalisoituminen ovat lisänneet tuotettavan datan määrää. Sama on havaittavissa myös sähköasemalla, jossa nykyaikaiset suojalaitteet kykenevät tuottamaan niin normaalitilanteeseen liittyvää prosessidataa kuin vikatilannekohtaisia häiriötallenteitakin. Kerättävästä datasta saadaan merkittävää liiketoiminnallista hyötyä, kun sitä jäsennellään ja prosessoidaan automaattisesti. Samalla avautuu uusia liiketoimintamahdollisuuksia, kun analyysipalveluita voidaan myydä sähköyhtiöille esimerkiksi pilvipalveluiden muodossa.

Tämä opinnäytetyö liittyy ABB:llä kehitettyyn ohjelmistoalustaan, jolla sähköasemadatalle voidaan ajaa erityisiä analyysiohjelmia. Työn päätavoitteena on selvittää, miten sähköasemadataa saadaan analysoitua mahdollisimman kattavasti kyseisellä alustalla. Tähän liittyen työn tavoitteisiin kuuluu myös ohjelmistoalustaan liittyvän konfigurointityökalun vaatimuksien kerääminen Elenia Oy:n kanssa toteutetussa pilottihankkeessa, työkalun toteuttaminen, työkalun testaaminen realistisessa käyttötilanteessa sekä ohjelmistoalustan laajentaminen tukemaan konfigurointiprosessia. Näiden lisäksi tavoitteena on selvittää, kuinka hyvin analyysiohjelmien ajoalusta skaalautuu analyysiohjelmien määrän kasvaessa. Analysoitavasta datasta käytännön osuudessa keskitytään sähköasemalta saataviin COMTRADE-häiriötallenteisiin.

Luku 2 käsittelee, miksi sähköasemalta saatavaa dataa kannattaa analysoida ja mihin analysoinnin tuloksia voidaan hyödyntää. Luvussa 3 esitellään työn kannalta oleelliset standardit IEC 61850:n ja IEEE C37.11:n (COMTRADE). Luku 4 kuvaa ABB:llä kehitetyn ohjelmistoalustan, jolla sähköasemadatan analyysiohjelmia voidaan suorittaa. Luvussa kuvataan myös alustaan liittyvä analyysiohjelmien konfigurointityökalu sekä alustan käyttämä tietokanta. Luvussa 5 kerrotaan sähköasemadatan analysointiin liittyvästä pilottihankkeesta, jossa kerättiin vaatimukset toteutettavalle konfigurointityökalulle. Varsinainen konfigurointityökalun toteutus tukikomponentteineen esitellään luvussa 6. Luvussa arvioidaan myös toteutuksen onnistumista käytännössä suoritetun testauksen perusteella. Lopulta johtopäätökset ja työn aikana havaitut jatkokehitysideat tuodaan esiin luvussa 7.

2 TARVE SÄHKÖASEMADATAN ANALYSOINNILLE

Sähköasemalla olevat älykkäät elektroniset laitteet (Intelligent Electronic Device, IED) ovat laitteita, joilla on monipuolisia elektronisia suojaustoimintoja, kehittyneitä paikallisia kontrollilogiikoita, kyky monitorointiin sekä kommunikaatioon (Strauss, 2003). Tässä diplomityössä IED, suojalaite sekä suojarele tarkoittavat samaa asiaa. IED:t tuottavat suuria määriä dataa, jota voidaan automaattisesti hyödyntää. Datan analysoinnilla voidaan esimerkiksi yrittää ymmärtää häiriöiden syitä sekä selvittää laitteiden toimintaa tarkemmin. Kerättyä dataa voidaan säilyttää sähköasemalla olevan asematietokoneen tietokannassa tai sitä voidaan kerätä myös yhteiseen tietokantaan yhtiön sisällä. (Kezunovic et al, 2009)

Tämä luku kuvaa yleisesti tarpeen sähköasemadatan analysoinnille. Kohdassa 2.1 käydään läpi syyt, miksi juuri vika- ja häiriödatan analysointi kannattaa automatisoida. Kohta 2.2 esittelee, miten muutos kohti älykkäitä sähköverkkoja lisää vaatimuksia datan analysoinnille sähköasematasolla. Lopulta kohdassa 2.3. tuodaan esiin mahdollisuudet sähköasemadatan analysoinnin tulosten hyödyntämiseksi.

2.1 Miksi vika- ja häiriödatan analysointi kannattaa automatisoida

Tarve vika- ja häiriödatan analysoinnille kasvaa samalla, kun sähkönjakelujärjestelmän kompleksisuus kasvaa. Automatisoimalla analysointia päästään luotettavammin ja nopeammin ymmärtämään häiriöiden ja vikojen syitä, mikä on tärkeää sähkönjakelujärjestelmän suorituskyvyn parantamiseksi.

Tämän kohdan sisältö perustuu lähteeseen (Cigre Working Group B5.20, 2010).

Järjestelmän luotettavuuden parantaminen

Sähkönjakelujärjestelmässä olevan viallisen komponentin tai osan nopea ja tarkka havaitseminen voi vähentää katkoksen kestoa. Samalla järjestelmän haavoittuvuus katkoksen aikana vähenee. Käytännön esimerkkinä vian etäisyyden paikallistaminen sähkölinjalta mahdollistaa vian nopeamman korjaamisen ja täten lyhemmän katkoksen sähkönjakelussa.

Sähköasemalta saatavaa dataa voidaan käyttää myös sähköaseman tilan arvioimisen parantamiseen. SCADA-järjestelmä (Supervisory Control and Data Acquisition) on valvomo-ohjelmisto, jossa on graafinen käyttöliittymä automaatiojärjestelmiin. Aikai-

semmin järjestelmän operaattorit tekivät kaikki hallintaan liittyvät päätökset SCADA-järjestelmästä saatavan operaatiodatan perusteella. Nykyään päätöksenteossa voidaankin hyödyntää esimerkiksi suojareleleiden tuottamaa dataa, mikä parantaa järjestelmän luotettavuutta.

Työntekijöiden tuottavuuden parantaminen

Vika- ja häiriödataa analysoidessa saattaa olla tarve käsitellä suurta määrää informaatiota. Vaikka osa vikatilanteista on ymmärrettävissä vain ihmisten asiantuntemuksella, automatisoitu analysointi voi tehdä merkittävän parannuksen suurta datamäärää prosessoidessa. Automatisoinnin tärkeänä etuna on mahdollisuus säästää työntekijöiden aikaa hoitamalla datan hankinnan ja muut toistuvat tehtävät. Näin työntekijät voivat keskittyä tärkeimpiin tapahtumiin. Toistuvien tehtävien automatisointi johtaa myös parempaan ja luotettavampaan diagnosointiin, sillä ihmiset ovat taipuvaisia virheisiin suoritettaessa toistuvia tehtäviä.

Nopeampi reagointiaika

Yleisesti automaattinen vika- ja häiriötilanteiden analyysi auttaa löytämään vikojen syyt, sijainnit ja piirteet nopeammin ja tarkemmin. Automatisointia rajoittaa tietoverkkojen nopeus ja prosessointitehon määrä, joista molemmat kehittyvät varsin nopeasti uusien teknologioiden myötä. Lisäksi automatisointi auttaa löytämään orastavat viat laitteista ennen niiden kehittymistä varsinaisiksi vioiksi.

Nopeampi reagointiaika luo automatisointia hyödyntävälle yritykselle suoranaista kilpailuetua vähentämällä katkosaikaa. Yhteiskunnan sähköriippuvuuden kasvaessa nopea palautuminen voi vaikuttaa merkittävästi suuren yleisön mielikuvaan sähkönjakeluyhtiöstä. Erityisesti suuren luokan katkokset kiinnittävät huomion sähkönjakeluyhtiöihin. Mikäli automatisointia ei hyödynnetä, voi yhtiö näyttää haluttomalta reagoimaan katkoksiin. Lisäksi nopea reagointi häiriöihin vähentää lakisääteisten korvausten maksutarvetta, joka voi pitkäkestoisten vikojen sattuessa nousta merkittäväksi taloudelliseksi rasitteeksi. Esimerkiksi vuonna 2013 sähköverkkoyhtiö Elenia maksoi korvauksia sähkökatkoista noin 70 000 asiakkaalleen (Yle, 2013).

Tehokkaampi datankäsittely

Automaattisen häiriö- ja vika-analyysin ottaminen käyttöön edellyttää automatisointiin liittyvän datan parempaa organisointia. Automatisointi vaatii, että jokainen tiedosto, tietokanta ja tietokannan osa ilmaisee selkeästi, mistä data on peräisin ja minkä tyyppistä dataa se sisältää. Datan organisointi auttaa myös löytämään järjestelmän konfigurointiin ja suunnitteluun liittyviä virheitä, kun järjestelmä on käytävä läpi automatisointia varten. Tärkein seuraus tehokkaammasta datankäsittelystä on tarkempien ja luotettavampien toimintojen tekeminen välttäen samalla ihmisperäisiä virheitä.

Tietomäärän kasvattaminen historiadataan perustuen

Sähkönjakelujärjestelmän kattava operointi edellyttää historiadataan perustuvan tilastodatan määrittelemisen, keräämisen ja pitämisen ajan tasalla. Vika- ja häiriötilanteiden automaattinen analysointijärjestelmä voi auttaa järjestelmän operointiin liittyvän statistiikan keräämistä ja ylläpitämistä. Esimerkkinä tilastodatan keräämisestä on pitkäkestoinen vika-analyysi, jossa useiden tapahtumien dataa käytetään muodostamaan kokonaiskuva järjestelmän haavoittuvuuksista. Näin voidaan selvittää, mitkä verkon osat ovat taipuvaisempia vikoihin ja minkä laitteiden kunto on heikentymässä. Lisäksi verkon laitteiden huolto ja korvaaminen uusilla voidaan suunnitella tehokkaammin, mikäli laitteiden kunnosta on statistiikkaa tarjolla.

Lainsäädännöllisen vaatimusten täyttäminen

Sähköyhtiölle asetetut lainsäädännölliset vaatimukset tietojen saamiseksi ovat lisääntymässä. Suomessa sähköverkonhaltijoiden tulee ilmoittaa vuosittain Energiavirastolle tunnusluvut, jotka kuvaavat verkkotoiminnan laajuutta, taloutta, kannattavuutta, hintatasoa, tehokkuutta sekä laatua (Energiavirasto, 2012). Esimerkkinä vaadituista tunnusluvuista on häiriöiden lukumäärä jännitetasoittain. Tietomäärän kasvaessa on käytännössä mahdotonta toimittaa kaikki vaadittavat tiedot manuaalisesti. Automatisoinnin avulla vaaditut tiedot saadaan toimitettua vähentäen samalla ihmisten työmäärää. Tämän lisäksi toimitettavaan tietoon voidaan kohdistaa suodattamista, jolla erotetaan epäoleelliset tapahtumat oikeista virhetilanteista. Myös yhdessä toimivien yritysten välistä toimintaa voidaan tehostaa käyttämällä automaattista analysointia, joka saattaa tulla valvonnan alaiseksi tiedon välittämisen tehostamiseksi.

Investoinnin tuoton kasvattaminen

Kun automatisoinnin hyödyt ja mahdollisuudet ovat otettu huomioon, on tärkeää huomioida myös investoinnista aiheutuvat kustannukset. Yleensä siirtyminen vika- ja häiriötallenteiden automatisoituun analysointiin tehdään vaiheittain, ja jokaisessa vaiheessa saavutettuja hyötyjä arvioidaan, jotta tarve tuleville investoinneille selviää. Vaiheittaista automatisointia voi hyödyttää esimerkiksi seuraavat asiat:

- Laitteiden kustannusten pieneneminen johtuen teknologisesta kehityksestä.
- Ikääntyvien laitteiden korvaaminen moderneimmilla laitteilla.
- Uusien sähköasemien varustaminen nykyaikaisilla, kuten IEC 61850 -standardin mukaisilla, laitteilla.
- Lainsäädännölliset vaatimukset, jotka ohjaavat päivittämään laitteita ja menetelmiä.

Lisäksi investoinneissa on tärkeää ottaa huomioon kerätty data laitteiden kunnosta, jotta investointien tarve ja huoltojen ajankohta voidaan optimoida.

Eri henkilöstöryhmien yhteistyön koordinointi

Automatisoinnin tuloksia voidaan hyödyntää yrityksen sisällä useassa eri liiketoiminta-alueessa. Näiden liiketoiminta-alueiden henkilöstöryhmien tulee toimia yhteistyössä,

jotta järjestelmän vaatimuksia määritellessä voidaan ottaa huomioon ryhmäkohtaiset tavoitteet. Yhteistyö ryhmien välillä mahdollistaa liiketoimintamallien päivittämisen ja datan hyödyntämisen parantamisen koko yrityksen laajuudella. Koska yhteistyö tällä tasolla ei ole kovin yleinen käytäntö, voidaan automatisoituja ratkaisuja pitää yhteistyötä kannustavina.

2.2 Kohti älykkäitä sähköverkkoja

Siirtyminen kohti älykkäitä sähköverkkoja luo muutospaineita koko sähkönjakelujärjestelmään. Sähköasematasolla saatavissa olevan datan ja sen prosessoinnin merkitys tulee kasvamaan. Samalla prosessoitua dataa voidaan hyödyntää valvomotasolla. Tässä kohdassa kuvataan muutoksen tuomia vaatimuksia sekä niihin ratkaisuksi esitettyä konseptia.

Tämän kohdan sisältö perustuu lähteeseen (Valtari & Verho, 2011).

2.2.1 Älykkäiden sähköverkkojen tuomat uudet vaatimukset sähköasematasolla

Jakeluverkon kehittyneempi kontrollointi

Yleisiä aiheita älykkäiden sähköverkkojen visioissa ovat itsestään parantuvat verkot sekä vian jälkeinen tehon palauttaminen. Jakeluverkossa tapahtuvan vian ilmaantuessa sähköaseman tulisi automaattisesti paikantaa ja eristää vika, jotta sähkönjakelua voidaan jatkaa verkon terveessä osassa. Tämä tilanne edellyttää osan tällä hetkellä verkon valvontakeskuksessa suoritettavista kehittyneemmistä kontrollointioperaatioista siirtämistä sähköasematasolle.

Automaattinen adaptaatio muutoksiin topologiassa, tuotannossa ja kulutuksessa

Vikatilanteiden lisäksi nopeaa ja automaattista reagointia tarvitaan myös normaalitilanteessa tapahtuviin muutoksiin. Verkossa olevien aktiivisten resurssien määrä voi vaihdella runsaasti, mikä voi vaatia kuormaa tasapainottavia toimintoja myös sähköasematasolla. Näiden lisäksi muutokset topologiassa ja resurssien määrässä edellyttävät myös suojaus- ja valvontatoimenpiteiden automaattista adaptaatiota.

Tarkempi ja selektiivisempi verkon suojaus

Yhteiskunnan sähköriippuvuuden kasvaessa vaatimukset keskeytyksettömälle sähkönjakelulle tulevat yhä tiukemmiksi. Samalla useiden maiden sähkönjakeluun liittyvä lainsäädäntö liikkuu kohti keskeytyksetöntä sähkönjakelua. Keskeytykset sähkönjakelussa tulee tallentaa tilastollista analyysia varten ja asiakkaita täytyy kompensoida liian pitkistä keskeytyksistä. Tämä lisää tarvetta olemassa olevien suojausfunktioiden parantamiselle ja uusien, tarkempien suojausfunktioiden kehittämiselle.

Laitteiden kunnonvalvonta ja laitteiden hallinta

Älykkäät sähköverkot edellyttävät kaikkien verkon resurssien optimaalista hyödyntämistä. Käytännössä jokaista verkon komponenttia tulee jatkuvasti valvoa, jotta komponenttien kunto ja kunnon edellyttämät huoltotoimenpiteet voidaan tarkasti tunnistaa. Ennen kuin laitteiden kunnonvalvontaan perustuvia huoltotoimenpiteitä voidaan kunnolla hyödyntää, tulee sähköasemalla olla mahdollista suorittaa kattavaa datan keräämistä ja prosessoimista.

Eri valmistajia tukeva alusta avoimilla rajapinnoilla ja isoilla tietomäärillä

Verkkojen kompleksisuuden kasvaessa kasvaa myös sähköasemilla liikkuvan datan määrä. Samalla useampien kiinnostuneiden osapuolien täytyy päästä käsiksi tähän dataan, sillä yhtiöt ovat yhä enemmän kiinnostuneita ulkoistamaan osia olemassa olevista palveluistaan. Ulkoistaminen kolmansille osapuolille edellyttää siis selkeitä ja avoimia rajapintoja sähköaseman prosessidataan.

Kun toiminnallisuus sähköasemalla kasvaa, kasvaa myös sähköasemalla suoritettavien toimintojen tarjoajien määrä. Tämä tarkoittaa, että avoimien prosessidatarajapintojen lisäksi avoimia rajapintoja tarvitaan myös dataa hyödyntäville ohjelmistoalustoille. Tulevaisuudessa samalla alustalla voidaan ajaa usean eri valmistajan kehittämiä sovelluksia, niin kuin nyt tehdään vähemmän kriittisissä ympäristöissä, kuten mobiililaitteissa.

Tietoturvallinen jakeluverkon palomuuri

Kun sähkönjakeluverkossa siirrytään hyödyntämään kehittyneempiä kommunikaatiomenetelmiä, tulee tietoturvasta oleellinen osa koko järjestelmän turvallisuutta. Yksittäinen tietoturvallinen tuote ei ole riittävä, sillä potentiaalisia haavoittuvuuksia voi nousta esiin turvattomasta integraatiosta olemassa oleviin infrastruktuureihin. Vaikka sähköasema voi muodostaa erillisen turvatun saarekkeen sähkönjakelulle, täytyy sen myös tarjota täysin varma informaatiopalomuuri sähköaseman kanssa kommunikoiville osapuolille.

Matalat elinkaarikustannukset

Edellä mainitut vaatimukset voivat muuttua ja uusia vaatimuksia voi tulla kasvavalla tahdilla. Yritykset eivät kuitenkaan halua kärsiä jatkuvista ja hintavista koko järjestelmää koskettavista päivitysprosesseista, vaikka tarve uusille ominaisuuksille olisikin ilmeinen. Hankinta- ja asennuskustannukset ovat selkeä osa sähköaseman laitteiden elinkaarikustannuksia, mutta tulevaisuudessa sähkönjakelujärjestelmien muuttuessa yhä dynaamisimmiksi kasvaa myös ylläpitokustannuksen merkitys. Tämä luo tarvetta helposti ja edullisesti tehtäville päivityksille.

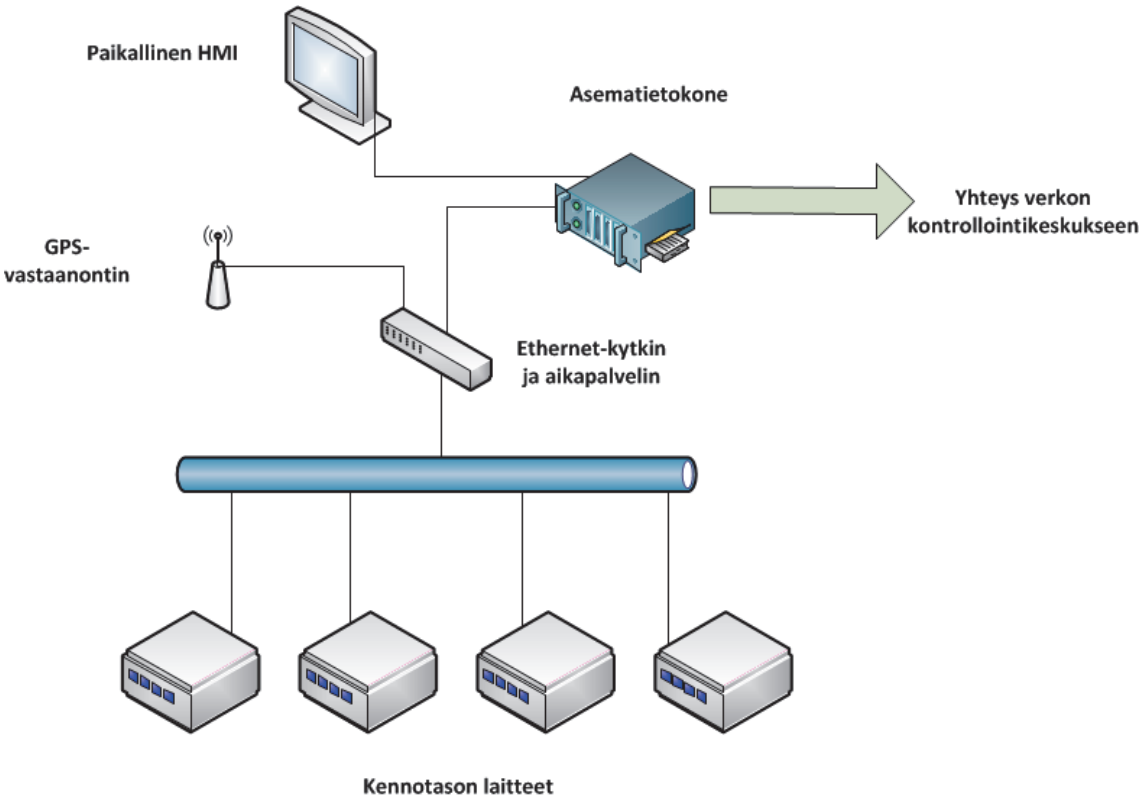
Elinkaarikustannuksin tulee vaikuttamaan myös jakeluverkon luotettavuuden kasvattaminen, jonka todellisia kustannuksia on vaikea arvioida. Asiakkaiden sähkönjakelukatkoksista johtuvien maksujen lisäksi sähkönlaadun merkitys kasvattaa merkitystään.

Lähitulevaisuudessa toimitetun sähköenergian laadulla voikin olla merkitystä siitä maksettuun hintaan.

2.2.2 Tulevaisuuden älykkäät sähköasemat

Jotta älykkäiden sähköverkkojen tuomat vaatimukset saataisiin täytettyä, on tulevaisuuden sähköasemille esitetty kolmea erityyppistä ratkaisua. Ensimmäisessä kaikki toiminnallisuus on keskitetty korkean tason suojalaitteisiin. Toinen vaihtoehto keskittää koko toiminnallisuuden asematietokoneeseen. Kolmantena vaihtoehtona on asematietokoneen ja suojalaitteiden yhdistäminen, johon tämä työ osittain liittyy.

Asematietokoneen ja suojalaitteiden yhdistäminen perustuu keskitettyihin suojaus- ja kontrollitoimintoihin. Nämä täydentävät kennotason suoja- ja kontrollilaitteiden toimintoja korvaamatta niitä kuitenkaan kokonaan. Kennotason suoja- ja kontrollilaitteet nähdään edelleen sähköaseman tukirankana. Ne ovat vastuussa perustason aikakriittisistä suojaustoiminnoista kommunikoiden keskitetyn asematietokoneen kanssa. Havainnollistava kuva esitetystä järjestelmästä löytyy kuvasta 2.1.



Kuva 2.1 *Esitetty älykkään sähköaseman rakenne. Mukailtu lähteestä (Valtari & Verho, 2011).*

Asematietokone

Asematietokoneen vastuulla ovat kaikki kehittyneemmät toiminnot. Koska kennotasolla olevat suojalaitteet huolehtivat ensisijaisista suojaustoiminnoista, voidaan asematieto-

koneeseen tehdä muutoksia ilman, että heikennetään verkon turvallisuutta. Täten mahdollistetaan vaivattomat ja nopeat päivitykset.

Asematietokoneessa olevat toiminnot voidaan jakaa kahteen eri kategoriaan: aikakriittisiin ja ajasta tiukasti riippumattomiin toimintoihin. Aikakriittiset toiminnot vaikuttavat suoraan verkon turvallisuuteen, ja ne vaativat reaaliaikaista prosessidataa. Näillä toiminnoilla on aina tiukat turvallisuus- ja luotettavuusvaatimukset, joten niitä ei tulisi avata monille kiinnostuneille osapuolille.

Ajasta riippumattomat toiminnot voivat operoida historiadatalla. Nämä toiminnot vaikuttavat verkon turvallisuuteen vain välillisesti esimerkiksi paikantamalla vian sijainnin. Myös avoimia rajapintoja voidaan tarjota kolmansille osapuolille ja usean valmistajan väliset ohjelmistoalustat ovat mahdollisia. Kohdassa 2.1 mainittu vika- ja häiriötallenteiden automaattinen analysointi kuuluu tähän kategoriaan.

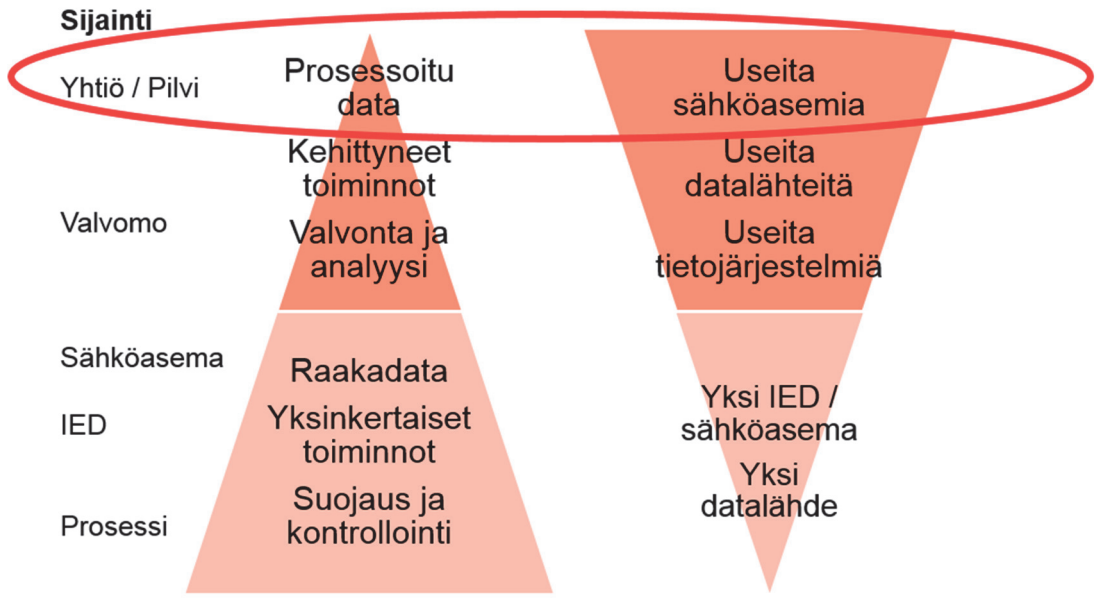
Kommunikaatio

Sähköasemastandardi IEC 61850:n kasvava käyttöönotto on tuonut standardisoidun Ethernet-pohjaisen kommunikaation sähköasemille. Sen mukainen asemaväylä sallii IED-laitteiden yhdistämisen kytkimen kautta ilman horisontaalisia fyysisiä kytkentöjä laitteiden välillä. Standardin mukainen prosessiväylä puolestaan tuo sähköaseman mittalaitteiden informaation saataville muille aseman laitteille standardissa muodossa. Sähköaseman kommunikaatio nojaakin tulevaisuudessa vahvasti tähän standardiin.

2.3 Analysoidun datan hyödyntäminen

Pilvipalvelut ovat kasvattaneet suosiotaan yritysmaailmassa, eivätkä sähköyhtiötkään tee poikkeusta niiden hyödyntämisessä. Tavallisten toimistosovelluksien lisäksi yhtiöt tutkivat sovellusaluekohtaisten ohjelmien ja toimintojen siirtämistä pilveen. Pilvipalveluista onkin tullut merkittävät osa siirtymistä älykkäisiin sähköverkkoihin. (Tweed, 2012)

Kehittyneempään vika- ja häiriötallenteiden analyysiin kykenevien sähköasemien määrän kasvaessa kasvaa myös analysoidun datan määrä. Analysoitua dataa tulee myös korkeamman tason analysointifunktioista. Analysoitua dataa voidaankin hyödyntää siirtämällä tulokset pilveen. Tämä tarjoaa ABB:n kaltaisille sähköasemalaitteiden valmistajille uusia liiketoimintamahdollisuuksia, kun laitteiden lisäksi asiakkaille voidaan myydä datan analysointia palveluna. Kuva datan prosessoinnista ja prosessoitavan datan lähteistä löytyy kuvasta 2.2. (ABB, 2013d)



Kuva 2.2 *Datan prosessoinnin tasot. Mukailtu lähteestä (ABB, 2013d).*

Pilvipalvelussa tarjottavia sähköasemalta saatavan datan analyysituloksia voidaan yhdistää avoimeen dataan, jota nykyään on yhä enemmän saatavilla. Esimerkiksi Ilmatieteen laitos on avannut oman verkkopalvelunsa, jossa sen tuottamaa tietoaineistoa voi hyödyntää (Ilmatieteenlaitos, 2014). Avoin data voi auttaa löytämään yhteyksiä vikojen ja niiden olosuhteiden välillä. Säädatasta kiinnostavia tietoja vian tapahtumahetkeltä ovat esimerkiksi tuulen nopeus ja suunta, sademäärä sekä ukkonen. Maaperädatasta voidaan puolestaan etsiä korrelaatiota vikapaikkojen ja maaperän välillä. (ABB, 2013d)

Visualisoinnin merkitys korostuu prosessoidun datamäärän kasvaessa. Prosessoitua tietoa käytetään apuna päätöksenteossa, joten visualisoinnin tulee keskittää käyttäjän huomio oleellisiin asioihin. Tärkeä tieto voi jäädä huomioimatta, mikäli se hukkuu tietomassan joukkoon. Parhaassa tapauksessa visualisointi auttaa löytämään jotain uutta visualisoitavan datan sisäisistä korrelaatioista ja suhteista, mikä on tärkeä osa menestyksekästä päätöksentekoa. (Steele, 2012)

3 STANDARDIT

Sähköasemalta saatavaan dataan liittyy merkittävästi kaksi standardia: IEC 61850 ja IEEE C37.111-1999 (COMTRADE). Tässä luvussa kyseiset standardit esitellään työhön soveltuvin osin.

3.1 IEC 61850

Vuosien saatossa tapahtunut sähköverkon suojalaitteiden teknologinen kehittyminen on lisännyt sähköasemalla liikkuvan informaation määrää. Informaation välittämiseen eri valmistajat ovat kehittäneet omia protokolliaan, joiden välinen yhteistoiminta on ollut haastavaa ja kallista. Tämä loi selkeän tarpeen yhtenäisille kommunikointimenetelmille, joilla eri valmistajien suojalaitteet voivat toimia yhdessä. Tarpeen täyttämiseksi lähdettiin kehittämään IEC 61850 -standardia.

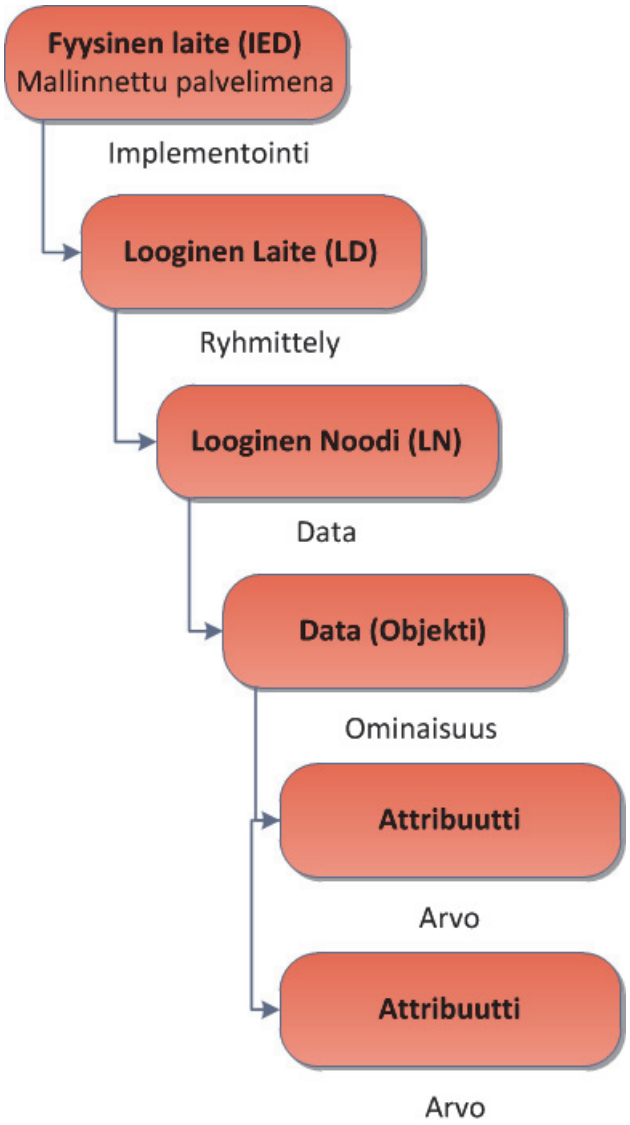
IEC 61850 -standardin tavoitteena on luoda kommunikaatiostandardi, joka tukee funktioita sähköaseman sisäisistä aina koko sähköverkon kattaviin funktioihin asti. Standardin tarkoituksena ei kuitenkaan ole funktioiden ominaisuuksien tai määrän rajoittaminen, vaan se pyrkii mahdollistamaan uusien teknologisten ratkaisujen adaptoimisen. Koska erityisesti kommunikaatioteknologiasta uusia innovaatioita tehdään nopealla tahdilla, standardissa oleva datamalli on eriytetty käytetyistä kommunikaatioprotokollista laittamalla ne abstraktien palvelujen ja objektien taakse. Tämä mahdollistaa protokollista riippumattomien ohjelmien kehittämisen ja täten teknologian kehittymisen hyödyntämisen. (IEC 61850-1, 2013)

3.1.1 Datan mallintaminen

IEC 61850 -standardin datamalli perustuu datan jakamiseen kahteen päätasoon:

- *Fyysisten laitteiden* jakamiseen *loogisiin laitteisiin*
- Loogisten laitteiden jakamiseen *loogisiin noodeihin*, *dataobjekteihin* ja *data-attribuutteihin*. (IEC 61850-1, 2013)

Standardin mukaista datan mallintamista on havainnollistettu kuvassa 3.1.



Kuva 3.1 *IEC 61850:n datan mallintaminen. Mukailtu lähteestä (IEC 61850-1, 2013).*

Funktiot

Suojalaitteiden toiminnot määritellään standardissa funktioiden avulla. Funktioiden määrä voi vaihdella laitteittain, ja yksittäinen funktio voi olla hajautettu useamman keskenään kommunikoivan loogisen laitteen välille. Funktiot ovat myös jaettavissa kolmeen eri tasoon: prosessi-, sähköasema- ja kennotasoon. (IEC 61850-1, 2013)

Looginen noodi (logical node, LN)

Looginen noodi määritellään standardissa funktion osana, joka toimii loogisen kokonaisuuden datasäiliönä. Yksittäinen funktio voi koostua useammasta eri loogisesta noodista, mutta yksittäinen looginen noodi kuuluu aina yhteen fyysiseen laitteeseen. Loogiset noodit voivat kommunikoida keskenään lähettämällä ja vastaanottamalla dataa, jolloin vastaanottavan noodin tulee pystyä tarkistamaan datan oikeellisuus ja laatu. Standardi määrittelee lähetettävälle datalle myös laatuattribuutit tiedonsiirtovirheistä selviytymiseksi. (IEC 61850-5, 2013)

Standardissa on määriteltynä 92 valmista loogista noodia. Loogiset noodit voidaan jakaa eri ryhmiin niiden toimintojen perusteella. Yksittäisen loogisen noodin nimi koostuu ryhmätunnuksesta sekä standardoidusta nimestä. Esimerkiksi piirin katkaisijan (cirquit breaker) loogisen noodin nimeksi standardi määrittelee XCBR:n. Mikäli samassa loogisessa laitteessa on useita instansseja samasta loogisesta noodista, on nimeen lisättävä instanssinumero. (IEC 61850-7-1, 2011; IEC 61850-7-4, 2010)

Dataobjektit ja -attribuutit

Looginen noodi koostuu määritellyistä dataobjekteista. Dataobjektit ovat standardin mukaisten yhteisten dataluokkien (Common Data Classes, CDC) instansseja. Yhteiset dataluokat määrittelevät datan varsinaisen tyypin. Yksittäisen dataluokan käyttöä voidaan rajoittaa funktionaalisilla rajoitteilla (Functional Constraints, FC). (IEC 61850-7-4, 2010)

Varsinainen data sijaitsee dataobjektin attribuuteissa. IEC 61850 määrittelee dataluokille vähintään kolme data-attribuuttia:

- arvo
- laatu
- aika.

Pakollisten attribuuttien lisäksi luokalla voi olla optionaalisia ja ehdollisia attribuutteja. (IEC 61850-7-1, 2011)

Looginen laite (logical device, LD)

Looginen laite ryhmittelee samankaltaiset loogiset noodit yhteen. Näiden lisäksi looginen laite voi sisältää erilaisia palveluita, jotka mahdollistavat esimerkiksi kommunikaation ja asetteluiden ryhmittelyn. Looginen laite kuulu aina yksittäiseen fyysiseen laitteeseen, ja niitä voi olla useampia samassa fyysisessä laitteessa. Standardin mukaan jokaisessa loogisessa laitteessa on vähintään loogisen laitteen yleisiä tietoja sisältävä looginen noodi LLN0 sekä fyysistä laitetta kuvaava LPHD-noodi. (IEC 61850-7-1, 2011)

3.1.2 Kommunikaatio

IEC 61850:n standardoima kommunikaatio on abstrahoitu *abstraktin kommunikaatiopalvelurajapinnan taakse* (ACSI, Abstract communication service interface). Se sisältää datamallin mukaisten luokkien määrittelyt. Näiden lisäksi siinä määritellään *palvelin*, joka edustaa ulospäin näkyvää toimintaa fyysisestä laitteesta. Datamallin ohella se sisältää seuraavat, dataan kohdistuvat palvelut määrittelevät, mallit:

- *Datajoukko* mahdollistaa dataobjektien ja attribuuttien ryhmittelyn.
- *Korvaaminen* tukee prosessiarvon korvaamista toisella.
- *Asetteluryhmien kontrollointi* määrittelee yksittäisten asetteluiden ja asetteluryhmien muokkaamisen ja vaihtamisen.

- *Raporttien kontrollointi* kuvailee, miten raportteja voidaan generoida ja lokitietoja voidaan kerätä.
- *GSE-lohkojen (generic substation event) kontrollointilohkot* tukevat nopeiden järjestelmänlaajuisten viestien jakelua.
- *Näytearvojen kontrollilohkot* näytearvojen nopeaan välittämiseen.
- *Kontrolli* kuvailee kontrolloivat palvelut, kuten laitteet.
- *Aika ja ajan synkronointi* tarjoavat aikaan liittyvät palvelut järjestelmälle ja laitteille.
- *Tiedostojärjestelmä* määrittelee suurten datalohkojen, kuten ohjelmien, vaihtamisen.
- *Seuranta* tarjoaa rajapinnan palveluiden diagnosointiin. (IEC 61850-7-2, 2010)

Abstraktin kommunikaatiorajapinnan ja varsinaisten käytettyjen kommunikaatioprotokollien välillä tarvitaan tietenkin mallintaminen, jotta järjestelmä voi toimia. Nykyinen standardi määrittelee mallintamiset MMS:ään (Manufacturing Message Specification, ISO 9506) ja Ethernet-kehyksiin (ISO/IEC 8802-3). Aikakriittistä tiedonvälitystä varten standardissa on esitelty *GOOSE*-viestit (generic object oriented substation event), joilla järjestelmässä voidaan välittää tietoa käyttäen Ethernetin linkkikerrosta ja broadcast-paketteja. Tilan muuttumista varten standardissa on esitelty *GSSE*-viestit (generic substation state event), joilla tilan muuttuminen voidaan ilmaista välittämällä bittipareja. (IEC 61850-7-2, 2010; IEC 61850-8-1, 2011)

3.1.3 Konfiguraatio

IEC 61850 -perustaisen kommunikaation konfiguroimiseksi standardissa on kehitetty erillinen *sähköaseman konfiguraation kuvauskieli* (Substation Configuration description Language, SCL). Se on XML-pohjainen kieli, joka kuvailee kaiken sähköasemalla tapahtuvan informaation vaihtamisen. XML (Extensible Markup Language) puolestaan on rakenteellinen merkintäkieli. SCL mahdollistaa myös tiedon välittämisen eri valmistajien konfigurointityökalujen välillä sekä takaisinpäin yhteensopivuuden laitteiden ja konfigurointityökalujen eri versioiden välillä.

Standardi määrittelee konfigurointiprosessiin *järjestelmän konfigurointityökalun* ja *IED-konfigurointityökalun*, joilla varsinainen konfigurointi suoritetaan. Nämä mallinnetaan erillisinä työkaluina, jotka voivat käytännön toteutuksessa sijaita samassa sovelluksessa. Työkaluilla voidaan muokata kuutta erityyppistä SCL-kuvaustiedostoa:

- ICD-tiedosto (IED Capability Description) kuvaa IED-laitteen toiminnot ja ominaisuudet. Sitä käytetään tiedon välittämiseen IED-konfigurointityökalun ja järjestelmän konfigurointityökalun välillä.
- IID-tiedosto (Instantiated IED Description) sisältää yhden IED:n tiedot, jotka välitetään IED-konfigurointityökalusta järjestelmän konfigurointityökaluun.

- SSD-tiedosto (System Specification Description) kuvaa sähköaseman rakenteen ja vaaditut loogiset noodit. Sitä käytetään järjestelmän konfigurointityökalussa.
- SCD-tiedosto (Substation Configuration Description) sisältää järjestelmän yleisrakenteen määrittelyn. Sitä käytetään tiedon välittämiseen järjestelmän konfigurointityökalusta IED-konfigurointityökaluille.
- CID-tiedosto (Configured IED Description) sisältää jo konfiguroidun IED-instanssin tiedot. Sillä välitetään informaatio IED-konfigurointityökalusta varsinaiselle IED-laitteelle.
- SED-tiedostolla (System Exchange Description) välitetään tietoa kahden eri projektin välillä. (IEC 61850-6, 2009)

3.2 IEEE C37.111-1999 (COMTRADE)

Vika- ja transienttidatan tallentamiseen ja testaamiseen käytettyjen digitaalisten laitteiden nopea kehittymistahti ja implementointi ovat luoneet tarpeen standardille dataformaatille sähköteollisuudessa. Vikadataa käytetään eri laitteiden toimesta tehostamaan ja automatisoimaan analyysia, testausta, kehitystä ja simulaatiota vika- ja häiriöolosuhteissa ja niihin liittyvissä suojatoimenpiteissä sähkönjakelujärjestelmissä. Koska jokainen datan lähde saattaa käyttää erilaista valmistajakohtaista formaattia, yhteinen standardiformaatti on välttämätön datan hyödyntämisen tehostamiseksi eri sovellusten välillä. Tätä tarkoitusta varten kehitettiin IEEE:n standardi COMTRADE (Common Format for Transient Data Exchange), joka määrittelee fyysisessä mediassa, kuten kiintolevyllä, olevien tiedostojen formaatin. (Ryan & Shank, 1999)

Standardi määrittelee neljä eri tiedostotyyppiä, joista jokaisessa on eri tason informaatiota. Yhteen tapahtumaan liittyvien tiedostojen nimien ilman tiedostopäätteitä tulee olla sama, enintään kahdeksan merkkiä pitkä sana ilman pisteitä ja välejä. Eri tiedostotyypit tiedostopäätteineen ovat:

a) header eli otsikko (.HDR)
b) configuration eli konfiguraatio (.CFG)
c) data (.DAT)
d) information eli informaatio (.INF).

Koska otsikko- ja informaatiotiedostot ovat optionaalisia, niitä ei esitellä tässä työssä.

Kohdan sisältö perustuu lähteeseen (IEEE C37.111-1999, 1999).

3.2.1 Konfiguraatiotiedosto

Konfiguraatiotiedosto on ASCII-merkistöinen tekstitiedosto, joka mahdollistaa datatiedoston tulkitsemisen. Konfiguraatiotiedostossa tulee olla seuraavat tiedot vastaavassa järjestyksessä:

a) aseman nimi, tallennusvälineen tunnus ja COMTRADE-standardin vuosi
b) kanavien lukumäärä ja tyyppi
c) kanavien nimet, yksiköt ja muunnoskertoimet

d) verkkotaajuus
e) näytteenottotaajuudet ja näytteiden lukumäärät per näytteenottotaajuus
f) ensimmäisen datapisteen päivämäärä ja kellonaika
g) ensimmäisen liipaisupisteen päivämäärä ja kellonaika
h) datatiedoston tyyppi
i) aikaleiman kerroin.

Listauksessa 3.1 on esimerkki oikealta sähköasemalla toimivasta releestä saadusta konfiguraatiotiedostosta.

```
StationName,10.15.7.1,1999
72,8A,64D
1,IL1,A,,A,1.17188,0,0,-32767,32767,300,5,P
2,IL2,B,,A,1.17188,0,0,-32767,32767,300,5,P
3,IL3,C,,A,1.17188,0,0,-32767,32767,300,5,P
4,Io,N,,A,0.273438,0,0,-32767,32767,70,1,P
5,Uo,N,,kV,0.0451055,0,0,-32767,32767,11.547,0.1,P
6,U1,A,,kV,0.078125,0,0,-32767,32767,20,0.1,P
7,U2,B,,kV,0.078125,0,0,-32767,32767,20,0.1,P
8,U3,C,,kV,0.078125,0,0,-32767,32767,20,0.1,P
1,DPHLPDOC1_START,,,0
2,DPHLPDOC2_START,,,0
…
…
64,Unused BI,,,0
50
1
1600,3200
08/09/2011,11:54:11.751587
08/09/2011,11:54:12.751587
BINARY
1
```

Listaus 3.1 *Esimerkki COMTRADE-konfiguraatiotiedostosta.*

Listauksessa 3.1 ensimmäinen rivi ilmaisee aseman nimen, IED:n tunnuksena olevan IP-osoitteen sekä COMTRADE-standardin version. Toinen rivi kertoo mitattujen kanavien kokonaismäärän, 72, sekä niiden jaon analogisiin ja digitaalisiin kanaviin. Seuraavana tulevat rivit kuvailevat analogisten ja digitaalisten kanavien tiedot siinä järjestyksessä. Näiden jälkeen konfiguraatiotiedostossa tulevat verkkotaajuus, näytteenottotaajuus ja näytteiden lukumäärä, ensimmäisen datapisteen sekä liipaisupisteen päivämäärät ja ajat, datatiedoston formaatti sekä aikaleiman kerroin. Standardin mukaan nimet, ku-

ten aseman nimi ja kanavan nimi, voivat olla myös tyhjiä. Tällöin kuitenkin on erotettava rivin osiot toisistaan käyttäen pilkkuja, kuten tehdään nimen ollessa määriteltynä.

3.2.2 Datatiedosto

Datatiedosto sisältää varsinaiset mitatut arvot. Tiedoston tulee vastata täysin konfiguraatiotiedoston määrittelemää formaattia. Datatiedoston tyyppi voi olla joko binääri tai ASCII.

ASCII-tyyppinen tiedosto on jaettu riveihin ja sarakkeisiin. Rivien määrä ja täten myös tiedoston koko riippuu häiriötallenteen kestosta. Yksittäinen rivi sisältää seuraavat sarakkeet pilkulla erotettuna:

a) näytteen numero
b) kyseisen näytteen aikaleima
c) analogiakanavien datojen arvot kyseisellä aikaleimalla
d) binäärikanavien datojen arvot kyseisellä aikaleimalla.

Aikaleiman muoto on pp/kk/vvvv,tt:mm:ss.ssssss, missä:

- pp on päivä välillä 1–31
- kk on kuukausi välillä 1–12
- vvvv on vuosi välillä 1900–9999
- tt kuvaa tunnit välillä 00–23
- mm kuvaa minuutit välillä 00–59 ja
- ss.ssssss kuvaa sekunnit välillä 00.000000–59.999999.

Viimeisen binäärikanavan jälkeen tulee rivinvaihtomerkki (CR/LF) sekä viimeisen näytteen ollessa kyseessä ASCII-tiedoston loppumerkki (IA HEX). Esimerkki ASCII-datatiedoston yhdestä rivistä on listauksessa 3.2. Siinä on kuusi analogikanavaa ja kuusi binäärikanavaa.

```
5,667,-760,1274, 72, 61, -140,  -502, 0,0,0,0,1,1 <CR/LF>
```

Listaus 3.2 *Yksittäinen rivi ASCII-muotoisesta COMTRADE-häiriötallenteesta.*

Binääritiedostossa data on yhtäjaksoisena virtana ilman mitään välimerkkejä, rivinvaihtomerkkejä tai tiedoston loppumerkkiä. Tiedostossa esitetään järjestyksessä näytteen numero, aikaleima, jokaisen analogiakanavan arvo sekä binäärikanavien arvot ryhmiteltynä. Datan tulkinta perustuu täysin sen sijaintiin tiedostossa, joten puuttuva tai korruptoitunut osio saattaa tehdä kokonaisen binääritiedoston käyttökelvottomaksi. Tarkemmin eriteltynä yksittäinen näyte esitetään seuraavasti:

a) Näytteen numero ja aikaleima esitetään molemmat etumerkittöminä neljän tavun binääreinä alkaen vähiten merkitsevästä tavusta.
b) Analogiakanavien näytteet esitetään kahden tavun binääreinä kahden komplementti–muodossa alkaen vähiten merkitsevästä tavusta. Heksadesimaalimuodossa oleva 8000 on varattu puuttuvan datan indikoimiseen.

c) Binäärikanavien näytteet 16 kanavan ryhmissä, joissa yhtä ryhmää vastaa kaksi tavua alkaen vähiten merkitsevästä tavusta. Kahden tavun vähiten merkitsevä bitti vastaa 16 kanavan ryhmän pienintä kanavanumeroa. Puutuvan datan indikoimiseen ei ole varattu erityistä merkkiä, mutta tiedoston eheyden vuoksi tavut tulee täydentää kokonaiseksi joko 0- tai 1-bitillä.

Listausta 3.2. vastaava esimerkki binääritiedostosta on esiteltynä listauksessa 3.3.

```
05 00 00 00 9B 02 00 00 08 FD FA 04 48 00 3D 00 74 FF 0A FE 30
00
```

Listaus 3.3 *Yksittäinen rivi binäärimuotoisesta COMTRADE-häiriötallenteesta.*

Listaus 3.3 on heksadesimaalimuodossa mukavuussyistä.

4 NYKYINEN JÄRJESTELMÄ JA VALMIIT KOMPONENTIT

ABB:n sisäisessä tutkimusprojektissa on kehitetty OffSide-niminen laskentajärjestelmä, jonka avulla sähköasemadatalle voidaan ajaa analyysiohjelmia. Laskentajärjestelmä hyödyntää cpmPlus-tuotannonhallintaohjelmiston tietokantaa datan prosessointiin ja PCM600-konfigurointityökalua ajettavien analyysiohjelmien konfigurointiin. Tämä luku kuvaa laskentajärjestelmän ja siihen liittyvät valmiit komponentit ja sovellukset soveltuvin osin.

4.1 CpmPlus-tuotannonhallintaohjelmisto

CpmPlus on ABB:lla kehitetty yhteistyön mahdollistava tuotannonhallintaohjelmisto (Collaborative Production Management, CPM), jota voidaan käyttää tuotteiden ja järjestelmien rakentamiseen valmistavassa prosessiteollisuudessa sekä palvelua tarjoavissa tahoissa, kuten sähköyhtiöissä. CpmPlus koostuu itsenäisistä ohjelmistomoduuleista, joita yhdistämällä saadaan aikaiseksi hyvin skaalautuva ohjelmistoalusta. Yksi näistä teknologiamoduuleista on RTDB-tietokanta (Real Time Database), joka on relaatiotietokanta historiatiedon säilyttämiseen. CpmPlusin arkkitehtuurin puolestaan kokoaa yhteen datan abstraktioon tehty rajapinta nimeltä VtrinLib. VtrinLib on samalla pääasiallinen tapa RTDB:n käsittelyyn.

Tämän kohdan sisältö perustuu lähteisiin (ABB, 2013a; ABB, 2013b).

4.1.1 CpmPlusin arkkitehtuuri

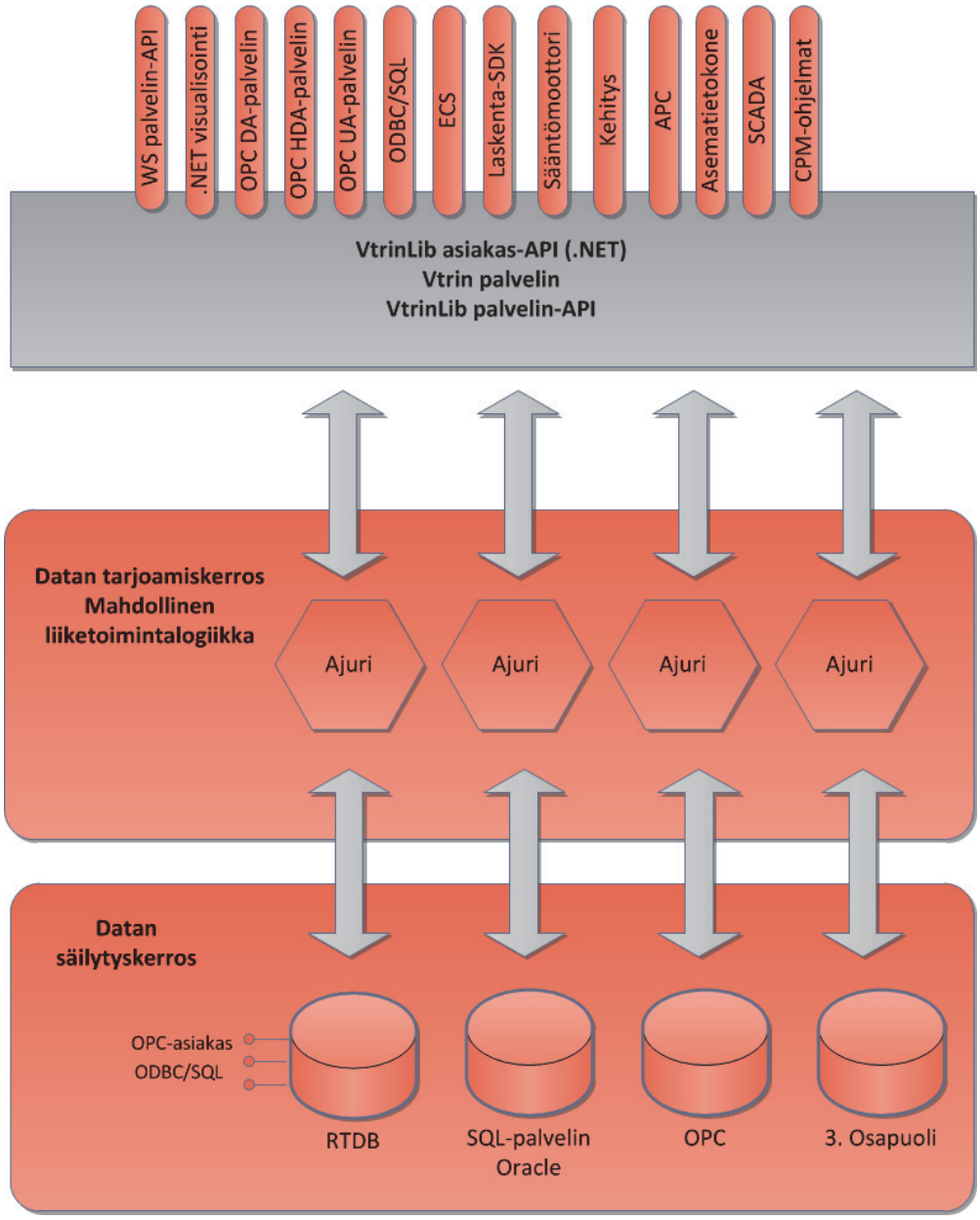
CpmPlusin arkkitehtuuri perustuu palveluiden integrointiin, mikä saadaan aikaiseksi käyttämällä yhtenäistä datan abstraktiorajapintaa eli VtrinLibiä. Datan abstraktiorajapinnan tarkoitus on:

- Muuttaa eri lähteistä tulevat tietorakenteet yhtenäisen oliomallin mukaisiksi.
- Eristää datan esittäminen sen säilömisestä ja ohjelmien liiketoimintalogiikasta.
- Tarjota hyvin määritelty malli ohjelmakohtaisten tietorakenteiden, liiketoimintalogiikoiden ja esityskerrosten lisäämiseen.
- Tarjota geneerinen kehitysrajapinta ohjelmadatalle.
- Tarjota laajennettava tapa lisätä järjestelmärajapintoja ilman muutoksia väliker-roksen rajapintoihin.
- Tarjota mahdollisuus hierarkkisen järjestelmän hajauttamiseen.

- Edistää alustan tietoturvaa.

VtrinLib on kehitetty Microsoftin .NET-sovelluskehystä käyttäen, ja se on asiakas–palvelin-mallin mukainen. Se piilottaa varsinaisen sisäisen tietovaraston, kuten esimerkiksi RTDB:n. Asiakas–palvelin-malli puolestaan mahdollistaa samanaikaisen yhteyden useaan eri datalähteeseen. Se myös vähentää tietokantojen kuormaa käyttämällä välimuistia sekä asiakkaan että palvelimen päässä lisäten samalla kokonaisuuden suorituskykyä.

Datan abstraktiorajapinta sisältää tuen käyttäjien autentikoinnille ja tietoturvalliseen datankäsittelyyn. Sen vastuulla on myös datan muutoksista aiheutuvien tapahtumien käsittely. Lisäksi se mahdollistaa datan suodattamisen. Yleiskuva cpmPlusin arkkitehtuurista löytyy kuvasta 4.1.



Kuva 4.1 *CpmPlus-ohjelmiston yleisarkkitehtuuri. Mukailtu lähteestä (ABB, 2013a).*

Kuvassa mainittu OPC (Open Platform Communications) on standardi avoimelle tiedonsiirrolle teollisuuden automaatiosovelluksissa (OPC Foundation, 2014).

VtrinLibin lisäksi cpmPlus tarjoaa seuraavat julkiset rajapinnat:

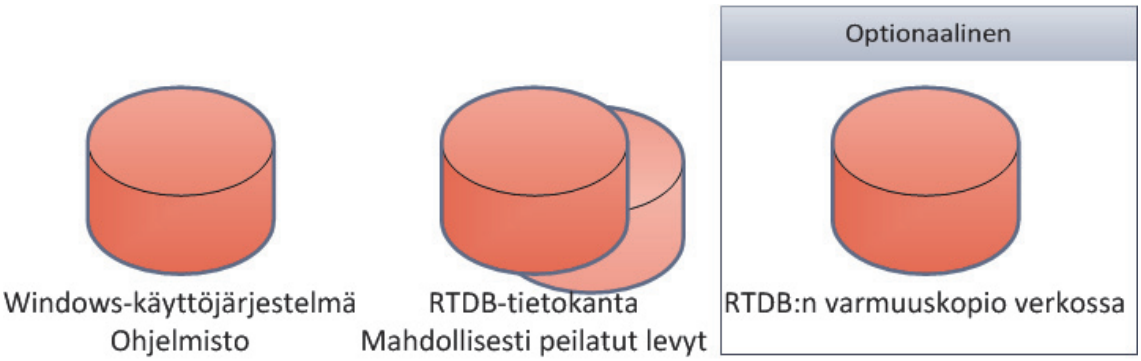
- OPC DA/HDA/AE- ja UA-asiakkaat datan hankintaan kontrollijärjestelmistä historiatietokantaan.
- OPC DA/HDA -palvelimet tarjoamaan pääsyn reaaliaika- sekä historiadataan.
- OPC UA -palvelimen tarjoamaan standardin, turvallisen ja alustasta riippumattoman pääsyn datan abstraktiokerroksessa olevaan dataan.
- JavaScript APIn alustasta riippumattomien näkymien tekoon nettiselaimille
- Web socket -palvelinrajapinnan datan abstraktiokerrokseen.

OPC:n määritelmistä automaatiosovelluksissa AE (Alarms and Events) sisältää ryhmän standardeja hälytysten ja tapahtumien välittämiseen, DA (Data Access) reaaliaikaisen datan välittämiseen, HDA (Historical Data Access) historiadatan välittämiseen ja UA (Unified Architecture) teollisuuden laitteiden välisen kommunikointiin. JavaScript API puolestaan on ohjelmointirajapinta Web-ympäristössä käytettävälle dynaamiselle komentosarjakielelle, kun taas Web socket on protokolla samanaikaisen kommunikaation mahdollistamiseksi molempiin suuntiin yhdessä TCP-yhteydessä.

Tietovarastoa voidaan käsitellä suojaan matalan tason rajapinnan, kuten SQL-tyyppisen (Structured Query Language) relaatiotietokannan kyselykielen, kautta. Tätä ei kuitenkaan suositella, jotta vältytään sitomasta cpmPlus-alusta ja sitä hyödyntävän sovelluksen sisäiset rakenteet liian tiukasti toisiinsa. Syynä tähän on, että liian tiukka sitoutuminen toisiinsa johtaisi tiukkoihin riippuvuuksiin eri sovellusversioiden välillä.

4.1.2 RTDB-historiatietokanta

RTDB on relaatiotietokanta, joka on suunniteltu ja optimoitu teollisuusprosessien tiedon hallintaan ja kattavan historiatiedon tallentamiseen. Se on kehitetty Windows-käyttöjärjestelmälle, jossa sitä suoritetaan taustalla useampana eri Windows-palveluna. Fyysisesti RTDB-tietokanta voi sijaita missä tahansa yksittäisessä Windowsin kansiossa, mutta tyypillisesti tietokanta ja siihen liittyvä ohjelmisto on jaoteltu eri kiintolevyille. Havainnollistava kuva RTDB:n peruskokoonpanosta löytyy kuvasta 4.2.

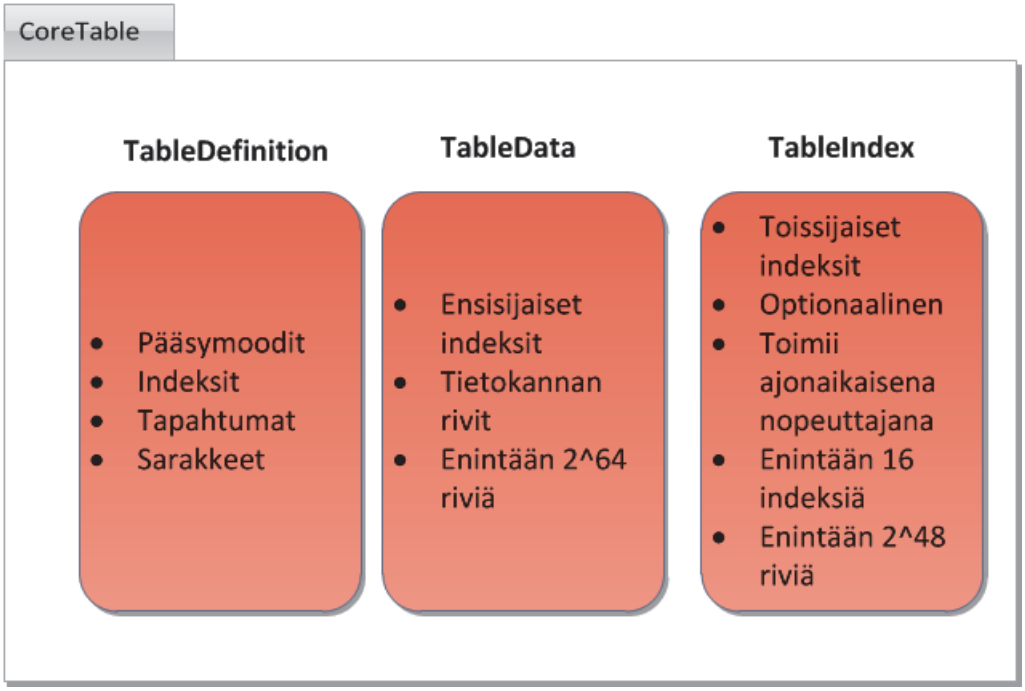


Kuva 4.2 *RTDB:n peruskokoonpano. Mukailtu lähteestä (ABB, 2013a).*

RTDB-tietokanta koostuu useista tauluista (CoreTable). Yksittäinen taulu puolestaan koostuu kahdesta pakollisesta ja yhdestä optionaalisesta fyysisestä tiedostosta:

- TableDefinition-tiedosto sisältää taulun metadatan, kuten sarakkeiden nimet ja tyypit.
- TableData-tiedosto sisältää varsinaisen riveistä koostuvan taulun sekä mahdolliset ensisijaiset indeksit.
- Optionaalinen TableIndex-tiedosto sisältää toissijaiset indeksit, mikäli taulussa sellaisia on.

Havainnollistava kuva taulun rakenteesta löytyy kuvasta 4.3.



Kuva 4.3 *RTDB-tietokannan taulun koostuminen. Mukailtu lähteestä (ABB, 2013a).*

Varsinainen prosessidata tallennetaan RTDB:ssä *muuttujiin* (variable). Yksittäisen muuttujan tyyppi voi olla teksti, 64-bittinen bittisarja, 64-bittinen kokonaisluku tai 64-bittinen liukuluku. Jokainen muuttuja identifioidaan enintään 32-merkkiä pitkällä unicode-muotoisella yksikäsitteisellä nimellä RTDB:n versiossa 4.2, jota OffSide käyttää. Uudemmissa versioissa muuttujan nimen suurinta mahdollista pituutta on kasvatettu 254 merkkiin asti.

Muuttujiin tallennettavat arvot ovat aikaleimasta ja senhetkisestä arvosta muodostettuja pareja. RTDB tukee aikaleimojen tarkkuutta aina 100 nanosekuntiin asti, mikä riittää varsin hyvin sähköasemalta saatuihin mittauksiin. Koska RTDB on suunniteltu prosessidatalle, se olettaa tallennettavien arvojen tulevan aikajärjestyksessä. Mikäli muuttujaan yritetään lisätä arvoa jo tallennettujen aikaleimojen väliin, rikkoutuu muuttujan historiakirjanpito.

Yksittäisen muuttujan tiedot muodostuvat useammasta valmiista taulusta. Näitä ovat muun muassa historiataulut sekä nykyisten arvojen taulu. Loppukäyttäjälle muuttujat

näyttävät kuitenkin olevan vain yksittäisessä taulussa. Tämä ratkaisu mahdollistaa tietokannan optimoinnin esimerkiksi vanhojen arvojen kompressoinnilla. Kompressointi ja historiatietojen säilytyksen asetukset ovat käyttäjän määriteltävissä.

Muuttujien lisäksi RTDB tukee käyttäjän määrittelemiä tauluja. Nämä ovat muodoltaan samanlaisia kuin RTDB:n valmiit olevat taulut. Käyttäjän määrittelemiä tauluja hyödynnetään esimerkiksi pilottihankkeen analyysiohjelmien tulosten säilömiseen (kohta 5.1).

Reaalimaailman laitteiden mallintamiseksi RTDB:ssä on tarjolla *laitemalli* (equipment model). Laitemalli on hierarkkinen rakenne, jolla voidaan kuvata loogisten yksiköiden, kuten erilaisten tuotantolaitteiden, ominaisuudet. Laitemallin avulla voi myös määritellä erilaisia hälytyksiä ja tapahtumia liittyen laitteisiin.

4.1.3 VtrinLib-abstraktiokerros

Datan abstraktiokerros eli VtrinLib tarjoaa olioperustaisen pääsyn mihin tahansa datanlähteeseen, johon on kehitetty sopiva ajuri. Tällä hetkellä ajureita löytyy RTDB:hen, SQL-pohjaisiin tietokantoihin, OPC-datalähteisiin, Oracleen sekä tekstitiedostoihin. VtrinLibin tärkeimpiä ominaisuuksia yhtenäisen rajapinnan lisäksi ovat datan varastointi välimuistiin, oliomalli, SQL-tyyppisten kyselyjen mahdollistava kyselymoottori, tietoturvallinen etäyhteys TCP:n tai HTTP:n yli, autentikointi, pääsyn kontrollointi sekä tapahtumat.

VtrinLibissä datan käsittely tehdään luokkainstanssien kautta. Uutta instanssia lisättäessä luokkainstanssivälimuistiin tehdään lisäyskomento ja olemassa olevaa instanssia päivitettäessä kyseiselle instanssille tehdään päivityskomento. Molemmista komennoista saadaan paluuarvoksi väliaikainen luokkainstanssiobjekti, johon halutut muutokset ja lisäykset tehdään. Tämän on kutsuttava commit-komentoa, jotta muutokset jäävät voimaan. Samaten lisäyksen ja päivityksen voi perua yksinkertaisesti hylkäämällä väliaikaisen instanssin.

VtrinLib tarjoaa mahdollisuuden tapahtumien kuunteluun. Mikäli operaatio tehdään VtrinLibin kautta, generoituu tapahtumasta ilmoitus aina automaattisesti. Datanlähteeseen tehtyjen ulkoisten muutoksien aiheuttamien tapahtumien täytyy generoitua kyseiselle datanlähteelle tehdyn ajurin kautta. Tapahtumia voidaan kuunnella:

- Luokkainstanssikohtaisesti, eli kun yksittäinen instanssi muuttuu.
- Luokkainstanssivälimuistikohtaisesti, eli kun jokin luokan instansseista muuttuu.
- Ajurikohtaisesti, eli kun minkä tahansa luokan mikä tahansa instanssi muuttuu.

Pääsyn kontrollointi puolestaan hoidetaan kolmella tasolla:

- luokkataso
- ominaisuustaso
- instanssitaso.

Luokkatason voidaan ajatella vastaavan tietokannan tauluja, ominaisuustason sarakkeita ja instanssitason rivejä. Jokainen taso perii pääsyn kontrollointi-informaation ylemmältä

tasolta oletuksena. Mikäli objektille on määritelty omistaja (parent), pääsyn kontrollointi peritään omistajalta. Pääsyn kontrolloinnin määrittelyyn on myös kolme tapaa:

- listat pääsyn kontrollointiin
- Unix-tyylinen jaottelu omistajiin-ryhmiin-muihin
- liput (flags).

Tiedonsiirron turvallisuuteen on tarjolla 2048-bittinen RSA-avaintenvaihto ja 128-bittinen AES-salaus. Mies välissä -hyökkäyksen estämiseksi voidaan käyttää ennalta jaettuja ja välimuistissa olevia avaimia. Käyttäjän todentaminen puolestaan perustuu käyttöjärjestelmän toteutukseen, joten Windowsissa ja Linuxissa on omat toteutuksensa. Lisäksi Windows-versio tukee käyttäjien valtuutusten (credentials) käyttämistä ja tallentamista.

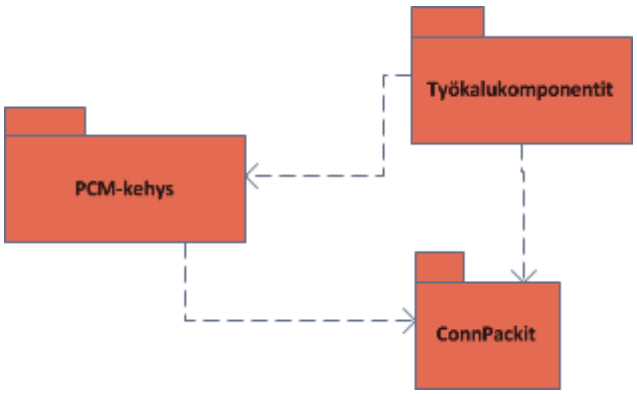
4.2 PCM600-konfigurointityökalu

PCM600 on ABB:n kehittämä konfigurointityökalu älykkäiden sähkölaitteiden konfiguroimiseen. Se tarjoaa yhtenäisen tavan käsitellä useita erityyppisiä suojalaitteita. Työkalua voidaan käyttää suojalaitteen koko elinkaaren aikana, eli konfiguroimisen lisäksi sillä voidaan esimerkiksi seurata suojalaitteen toimintaa ja kerätä suojalaitteen häiriötallenteita. Varsinainen konfigurointi voidaan suorittaa suoraan kytkemällä PCM600:n sisältävä tietokone suojalaitteeseen tai etänä Ethernet-yhteyden yli.

Tämän kohdan sisältö perustuu lähteisiin (ABB, 2010; ABB, 2013c).

4.2.1 PCM600-työkalun arkkitehtuuri

PCM600-työkalun arkkitehtuuri jakautuu kolmeen pääelementtiin: *PCM-kehykseen* (PCMFrame), *työkalukomponentteihin* (Tools) ja *liitettävyyspaketteihin* (ConnPack). Työkalun korkean tason arkkitehtuuri on kuvattu kuvassa 4.4.



Kuva 4.4 *PCM600-työkalun arkkitehtuuri. Mukailtu lähteestä (ABB, 2010).*

PCM-kehys on sovelluskehys, joka sisältää ydintoiminnallisuuden ja palvelut. Samalla se mahdollistaa siihen perustuville sovelluksille yhtenäisen ulkoasun ja yhtenäiset rajapinnat uusien toimintojen toteuttamiseksi.

Työkalukomponentit ovat PCM-kehyksen lataamia liitännäisiä, jotka lisäävät toimintoja PCM-kehykseen perustuvaan ohjelmistoon. PCM600 jaottelee käyttämänsä työkalukomponentit kolmeen kategoriaan: järjestelmä-, objekti- ja palvelutyökaluihin. Järjestelmätyökalut operoivat PCM600:n järjestelmädatan, kuten projektien ja käyttäjätunnusten, kanssa. Objektityökalut operoivat jonkin objektin, kuten IED:n, kanssa. Palvelutyökaluja puolestaan ajetaan taustalla ilman työkaluun liittyvää käyttöliittymää. Yksittäinen työkalu voi kuitenkin kuulua useampaan eri kategoriaan.

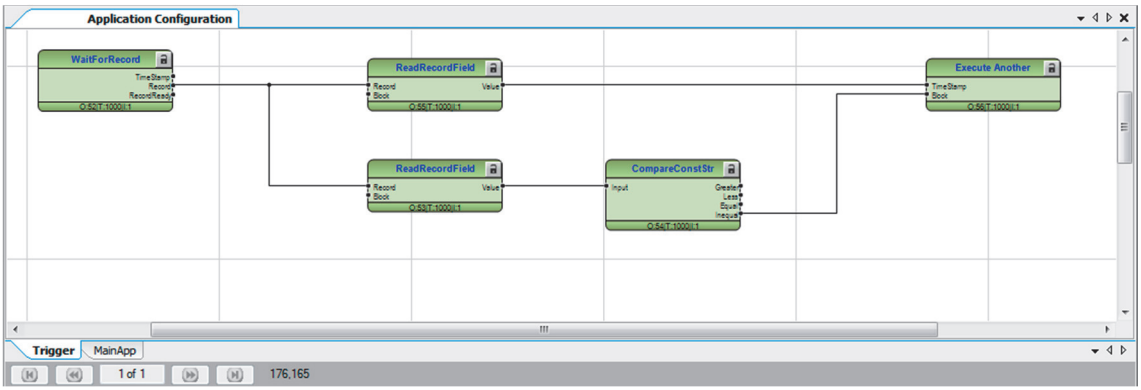
Liitettävyyspaketit mahdollistavat kommunikoinnin laitteiden ja PCM-kehykseen pohjautuvan ohjelmiston välillä. Ne voidaan jakaa kahteen kategoriaan: kommunikaatioliitettävyyspaketteihin ja IED-liitettävyyspaketteihin. Kommunikaatioliitettävyyspaketti sisältää tyypillisesti tarvittavat objektit yhden kommunikaatioprotokollan mahdollistamiseksi. IED-liitettävyyspaketti puolestaan sisältää objektityypit, jotka kapseloivat suojalaitteeseen tai suojalaiteperheeseen liittyvät datat. Liitettävyyspaketin ytimenä toimii *objektityyppi*, joka toimii välittäjäkomponenttina PCM:n ja liitettävyyspaketin välillä. Objektityypin tulee toteuttaa haluttu määrä työkalukomponenttien rajapintoja, joista PCM600 osaa näyttää toteutetut työkalut konfigurointityökalussa. Suojalaitteen konfiguroitavat parametrit, suojausohjelmien toimilohkot ja lohkojen väliset kytkennät mallinnetaan liitettävyyspaketissa *tyyppidatana*. Tyyppidata on datamalli, jota käytetään työkaluissa näkyvän *instanssidatan* generoimiseen.

4.2.2 Työkalut

PCM600:ssa on kiinteänä osana työkaluja, joilla suojalaitteita voidaan konfiguroida. Työkalut saavat tiedot konfiguroitavan laitteen ominaisuuksista suoraan ConnPack-laajennukselta. Nämä tiedot tuodaan käyttäjän muokattavaksi työkalujen avulla. Tässä kohdassa esitetään tämän työn kannalta olennaisimmat työkalut.

Sovelluksen konfigurointityökalu

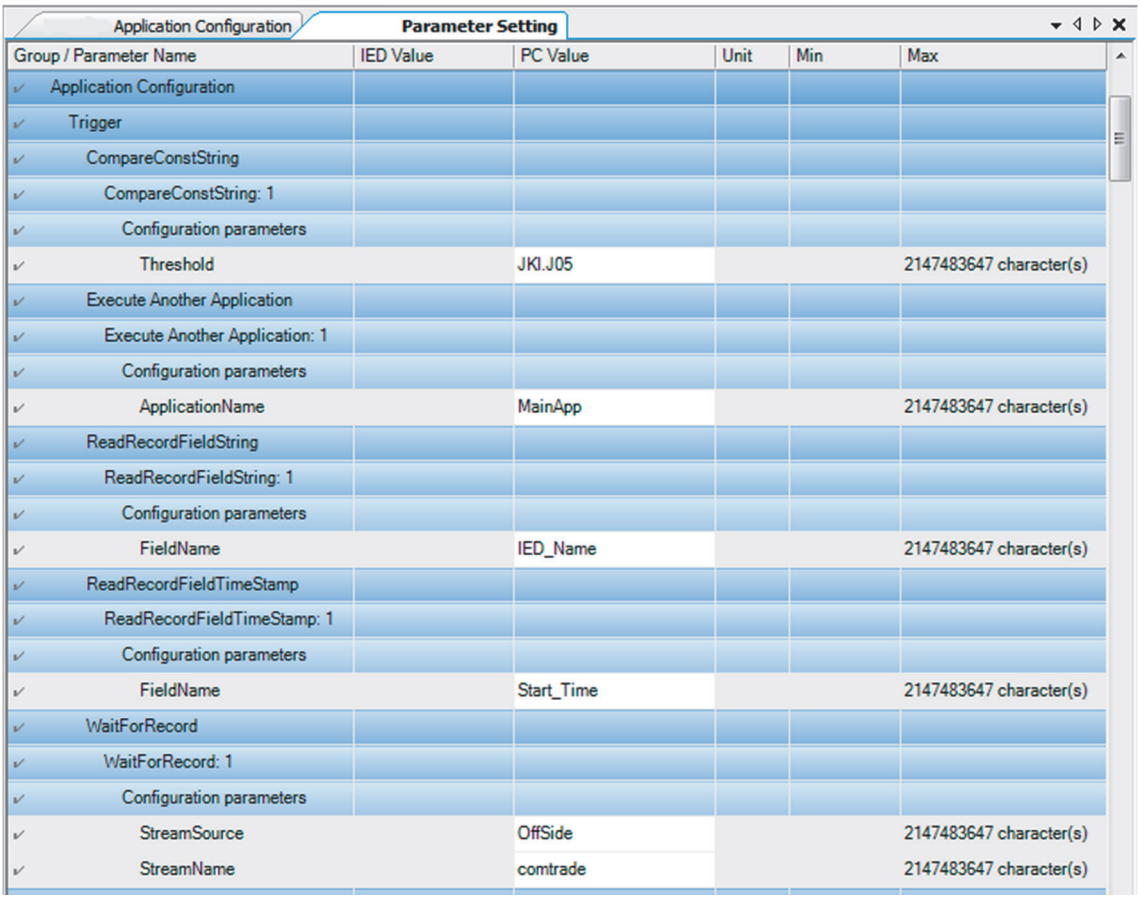
Sovelluksen konfigurointityökalu (Application Configuration Tool, ACT) tarjoaa graafisen tavan sovellusten luomiseen ja muokkaamiseen. Sovelluksien tekeminen tapahtuu yhdistelemällä suojalaitteen tukemia toimilohkoja toisiinsa. Kuvassa 4.5 on esimerkki OffSide-laskentajärjestelmälle kehitetyn analyysiohjelman osasta.



Kuva 4.5 *Esimerkki ohjelman rakenteesta sovelluksen konfigurointityökalussa.*

Parametrien asettelutyökalu

Parametrien asettelutyökalua (Parameter Setting Tool, PST) käytetään suojalaitteen parametrien visualisointiin ja konfiguroimiseen. Sillä asetellaan myös sovelluksen konfigurointityökalulla luodut ohjelmat. Kuvan 4.5 mukaisen ohjelman parametrit parametrien asettelutyökalussa löytyvät kuvasta 4.6.



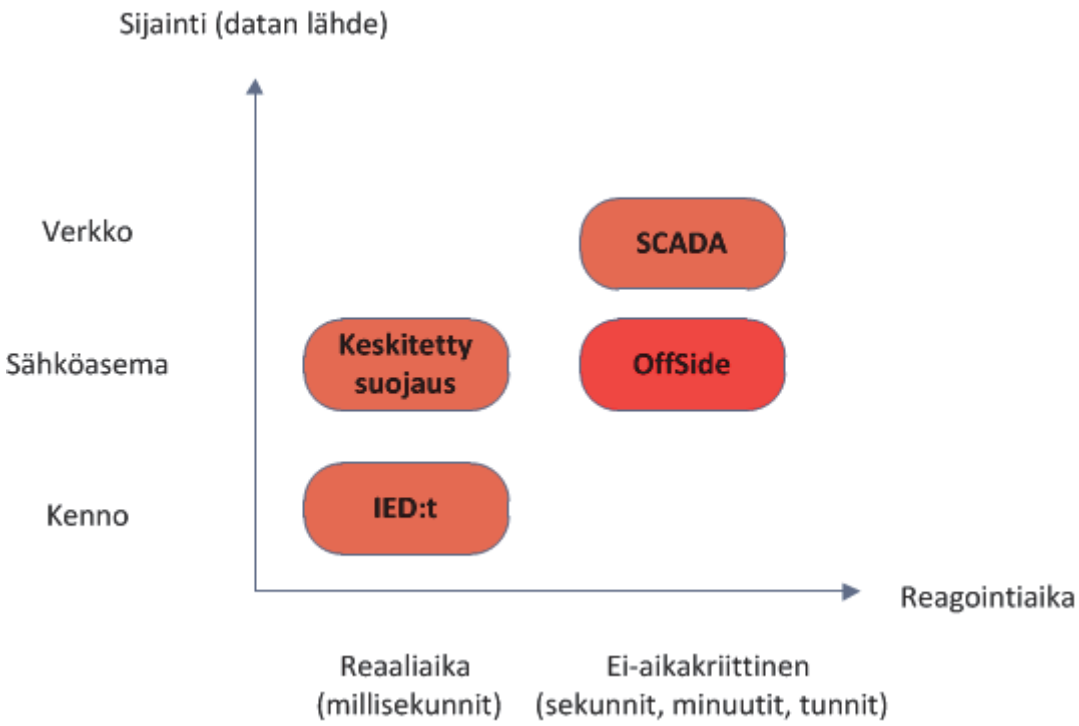
Kuva 4.6 *Esimerkki ohjelman parametreista parametrien asettelutyökalussa.*

Konfiguraation luku- ja kirjoitustyökalu

Konfiguraation luku- ja kirjoitustyökalulla (Common Read and Write, CRW) kirjoitetaan ACT:llä ja PST:llä viimeistellyt ohjelmat suojalaitteeseen. Se myös tarjoaa mahdollisuuden olemassa olevan konfiguraation lukemiseen suojalaitteesta PCM600-konfigurointityökaluun. CRW tarkistaa konfiguraation eheyden ennen kirjoittamista laitteeseen ja ennen konfiguraation tuomista PCM600:aan. Konfiguraation luku- ja kirjoitustyökalu saa kirjoitettavan datansa PCM600:an *konfiguraatio-* ja *parametridatapalvelimilta*. Konfiguraatiodatapalvelin (Configuration Data Server, CDS) on PCM:n työkalu, joka säilyttää tiedot liittyen IED:n funktiomalliin ja parametridatapalvelin (Parameter Data Server, PDS) on PCM:n työkalu, jonka vastuulla on IED:n tyyppidatan sekä parametrien hallinta.

4.3 OffSide-laskentaympäristö

OffSide on ABB:llä kehitetty sovelluskehys, joka kehitettiin analyysiohjelmien suorittamiseen sähköasematasolla. Analyysiohjelmat eivät OffSiden tapauksessa saa vaatia tiukkaa reaaliaikaista millisekuntien luokkaa olevaa vasteaikaa, vaan niiden vasteajat voivat vaihdella sekunneista aina useisiin tunteihin. OffSiden sijainnin ja aikakriittisyyden suhdetta on havainnollistettu kuvassa 4.7.



Kuva 4.7 *OffSiden sijainti suhteutettuna reagointiaikaan. Mukailtu lähteestä (ABB, 2012).*

Vaikka OffSide on alun perin suunniteltu sähköasematasolle, hyödynnetään sitä tässä työssä myös korkeammalla tasolla. Käytännössä tämä tarkoittaa, että saatavilla on samaan aikaan prosessoitavaa dataa useammalta eri sähköasemalta.

Tämän kohdan sisältö perustuu lähteisiin (ABB, 2012; ABB, 2013e).

4.3.1 Yleiskuva arkkitehtuurista

OffSiden arkkitehtuuri rakentuu viiden suunnitteluperiaatteen pohjalle. Nämä ovat sovelluskehyksenä oleminen, laajennettavuus, valmiiden komponenttien ja ympäristöjen hyödyntäminen, nykyaikaisten teknologioiden käyttäminen sekä käyttökokemuksen yhdenmukaisuus olemassa olevien suojausohjelmien määrittelyn ja suorittamiseen tarkoitettujen ympäristöjen kanssa.

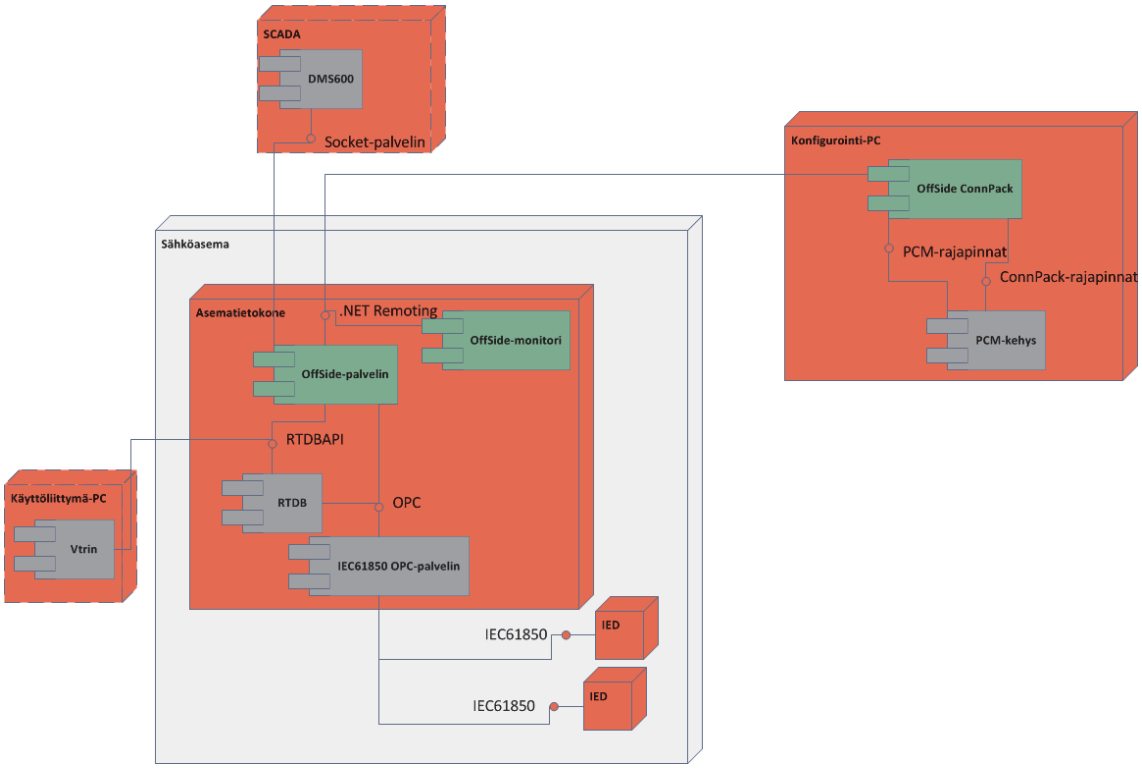
OffSide on kehitetty Microsoftin .NET-kehystä apuna käyttäen, ja se on implementoitu lähestulkoon kokonaan C#:lla. ABB:llä kehitetyistä sovelluskehyksistä ja komponenteista hyödynnetään sähköasematietokonetta laitteistoalustana, sähköasematietokoneen

ohjelmistoinfrastruktuuria erinäisten toimintojen, kuten OPC-palvelimen ja kommunikaation, toteuttamiseen, RTDB:tä historiatietokantana, releiden toimilohkoja funktiolohkojen pohjina sekä PCM600-työkalua OffSidessa ajettavien analyysiohjelmien kehittämiseen.

OffSiden pääkomponenttina on OffSide Server, joka on sähköasematietokoneella jatkuvasti suoritettava Windows-palvelu. Nimestään huolimatta kyseessä ei ole varsinainen palvelin, vaan enemmän käyttäjien määrittelemien analyysiohjelmien ajamiseen tarkoitettu ympäristö. Ohjelmat saavat prosessoitavan datansa samalla sähköasematietokoneella olevasta RTDB:stä. Tietokannassa oleva data voidaan visualisoida RTDB:n Vtrin-kirjaston avulla. RTDB:hen puolestaan data voi tulla IED:ltä OPC-palvelinten välityksellä. Myös suora yhteys OPC-palvelimen ja OffSiden välillä on mahdollinen.

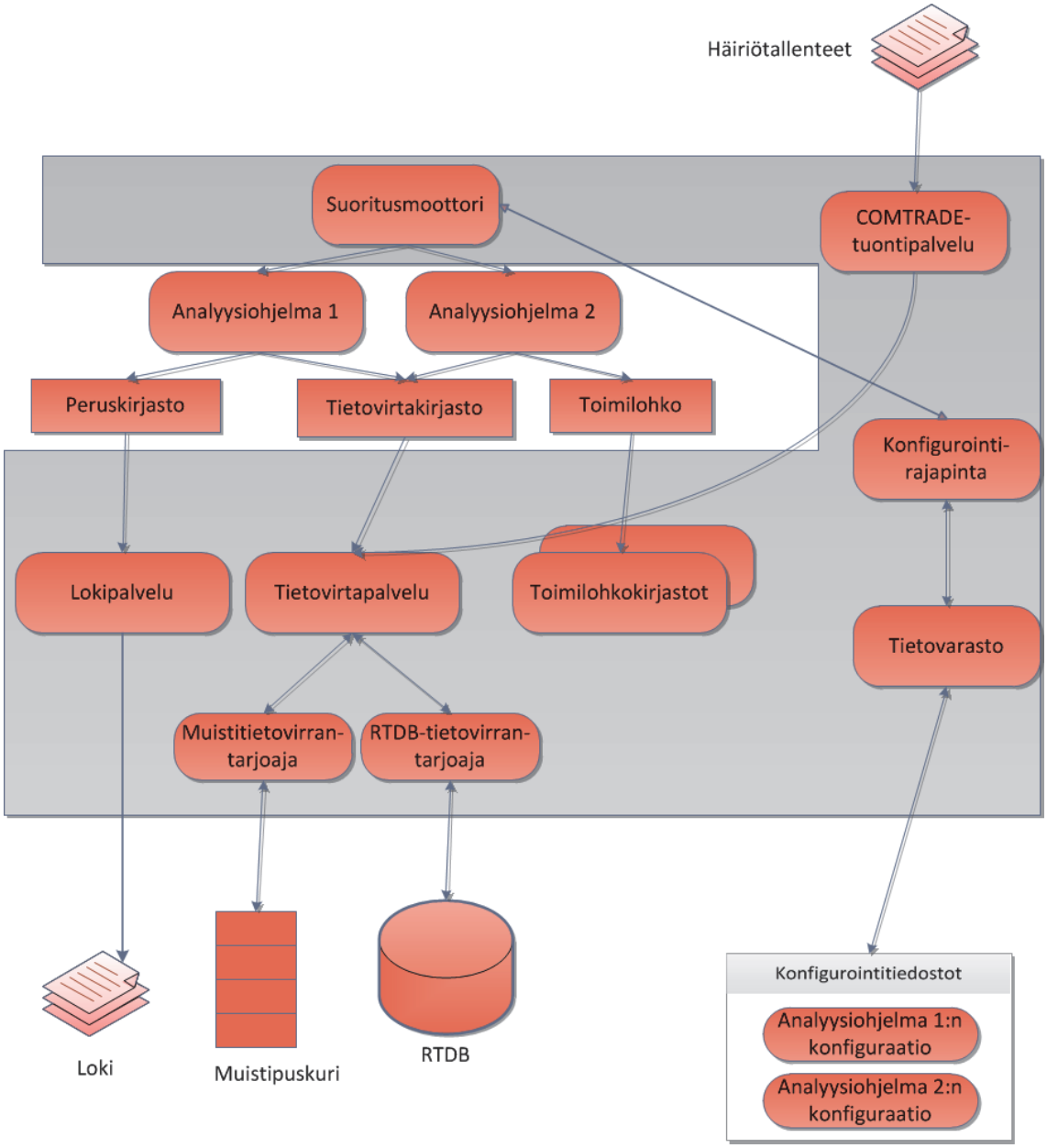
OffSide-palvelin toteuttaa .NET Remoting –rajapinnan, mikä mahdollistaa järjestelmän konfiguroimisen ja monitoroinnin. Tätä rajapintaa hyödyntävä monitorointisovellus on asennettavissa asematietokoneelle järjestelmän tilan tarkkailua varten. OffSide voi myös ottaa yhteyden muihin järjestelmiin, kuten SCADAan, esimerkiksi socket-pohjaisella kommunikaatiolla.

OffSidessä ajettavien analyysiohjelmien konfiguroiminen tehdään tyypillisesti toiselta tietokoneelta käyttäen PCM600:aa. Konfiguroimiseen käytettävälle tietokoneelle tulee tällöin asentaa OffSide Connectivity Package, jotta PCM600 tulee tietoiseksi OffSidesta. Edellä mainitut osat järjestelmästä löytyvät korkean tason sijoittelukaaviosta kuvasta 4.8.



Kuva 4.8 *OffSiden sijoittelukaavio. Mukailtu lähteestä (ABB, 2012).*

Analyysiohjelmien suhdetta suoritusympäristöön on havainnollistettu puolestaan kuvassa 4.9.



Kuva 4.9 *Analyysiohjelmien suhde suoritusympäristöön. Mukailtu lähteestä (ABB, 2012).*

Suurin osa kuvassa 4.9 näkyvistä komponenteista on selitetty seuraavissa alikohdissa.

4.3.2 Yleiset komponentit

RTDBAPI mahdollistaa pääsyn RTDB-tietokantaan. Se hyödyntää ABB:n sisäistä Vtrin-kirjastoa, joka on RTDB:n virallinen rajapinta. RTDBAPI tarjoaa kuitenkin OffSiden käyttöön paremmin sopivat operaatiot ja abstraktiot. API koostuu kolmesta luokasta:

- RTDB-luokka tarjoaa yhteyden varsinaiseen tietokantaan.
- Variable-luokka tarjoaa pääsyn yhteen muuttujaan.
- Table-luokka tarjoaa pääsyn RTDB:n yhteen tauluun.

COMTRADEAPI on matalan tason rajapinta COMTRADE-häiriötallenteen lukemiseen ja jäsentämiseen. Jäsentäminen tapahtuu noudattaen IEEE C37.111-1999 –standardia. COMTRADEAPI on toteutettu C++:lla, mutta se sisältää COMTRADE-nimisen käärijäluokan, joka tarjoaa pääsyn rajapintaan OffSiden C#-koodista.

IED-kohtaisten COMTRADE-kanavien numeroiden ja tunnisteiden kuvausten määrittelemiseksi OffSidessa on määritelty *COMTRADEChannelConfiguration*-nimellä oleva XML-tiedosto. Se sisältää IED:n nimen ja sitä vastaavien COMTRADE-tallenteiden analogi- ja binäärikanavien kanavanumerot ja niitä vastaavat tunnisteet. Esimerkki tällaisesta XML-tiedostosta on listauksessa 4.1.

```
<?xml version="1.0" encoding="utf-8"?>
<COMTRADEChannelConfiguration
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <IED Name="JKI.J05">
    <AnalogChannel Id="IL1" Index="1" />
    <AnalogChannel Id="IL2" Index="2" />
    <AnalogChannel Id="IL3" Index="3" />
    <BinaryChannel Id="DPHLPDOC_ST" Index="1" />
    <BinaryChannel Id="DPHLPDOC_OP" Index="15" />
  </IED>
</COMTRADEChannelConfiguration>
```

Listaus 4.1 *Esimerkki COMTRADE-kanavakonfiguraatiosta.*

Listauksessa 4.1. on yksi ”JKI.J05”-niminen IED, jolle on kuvattu kolme analogikanavaa ja kaksi binäärikanavaa. Ensimmäisestä analogikanavasta löytyy tunniste ”IL1”, toisesta ”IL2” ja niin edelleen. ” DPHLPDOC_ST” löytyy puolestaan binäärikanavasta 1 ja “DPHLPDOC_OP” binäärikanavasta numero 15.

Liitännäisten lataamiseen OffSidessa on *liitännäismanageri* (plugin manager), joka on singleton-sunnittelumallin (Martin, 2012) mukainen luokka. Singleton-suunnittelumallin mukaisesta luokasta on saatavilla aina vaan yksi instanssi. Liitännäismanageria voidaan käyttää .NET-kokoonpanotiedostojen automaattiseen löytämiseen ja lataamiseen sekä ladattujen kokoonpanotiedostojen mukaisiin tyyppien käyttämiseen. Liitännäistyyppi ladataan, mikäli:

- se on DLL-tyyppisessä .NET-kokoonpanotiedostossa
- samaa nimeä omaavaa kokoonpanotiedostoa ei ole ladattu aikaisemmin
- sen kokoonpanotiedostossa on liitännäisattribuutti määriteltynä
- se on määritelty julkiseksi ja
- kyseessä on luokka, joka ei ole abstrakti.

DLL-tiedostotyyppi (Dynamic-link library) on Microsoftin kehittämä tiedostotyyppi jaetuille ohjelmakirjastoille.

Työläiset (workers) ovat komponentteja, jotka prosessoivat tietoa omissa säikeissään. Niitä käyttämällä saadaan kasvatettua järjestelmän suorituskykyä. Jokaisella työläisinstanssilla on referenssi funktiokahvaan, jolla työläinen voidaan käynnistää ja pysäyttää. Työläisiä voi olla lukuisia erityyppisiä, mutta OffSiden tapauksessa käytetyimmät jakautuvat kahteen kategoriaan:

- *jaksotyöläinen* (periodic worker) suorittaa operaation aina tietyn ajanhetken välein
- *jonotyöläisellä* (queue worker) on jonossa kohteita, jolle suoritetaan toimenpide jonologiikan mukaisessa järjestyksessä.

4.3.3 Palvelin

Koko OffSiden toiminnallisuus on kapseloitu yhteen *palvelinobjektiluokkaan* (ServerObject). Tämän luokan vastuulla on analyysiohjelmien määrittelyjen lataaminen sekä palveluinstanssien ja analyysiohjelmainstanssien hallitseminen. Varsinainen palvelinobjekti on singleton-sunnittelumallin mukainen, eli saatavilla on aina vaan yksi instanssi staattisen instanssiominaisuuden kautta. Luokkaan voi ottaa yhteyttä .NETin etärajapintojen kautta.

Ylimmällä tasolla palvelinobjektin tarjoamat kaksi merkittävintä metodia ovat käynnistäminen ja pysäyttäminen. Käynnistämisen vastuulla on liitännäisten lataaminen, etäpalveluiden vaadittavan infrastruktuurin alustaminen, tilastojen ja tietovarastoihin pääsyn alustaminen, analyysiohjelmien ja niihin liittyvien asetusten lataaminen sekä palveluiden ja ohjelmien käynnistäminen. Nimensä mukaisesti pysäyttämisen vastuulla on käänteisesti käynnistämiseen nähden ajettavien ohjelmien ja palveluiden pysäyttäminen sekä käynnistämisestä syntyneiden jälkien siivoaminen.

Palvelinobjektia voidaan suorittaa kahdella eri tavalla: Windows-palveluna tai konsoliohjemana. Windows-palvelu on tuotantokäyttöön tarkoitettu tapa, ja se mahdollistaa OffSiden käynnistämisen automaattisesti asematietokoneen käynnistyessä. Konsoliohjelma puolestaan on tarkoitettu kehitys- ja ongelmanratkaisukäyttöön, sillä se mahdollistaa pienimuotoisen interaktion käyttäjän kanssa.

4.3.4 Palvelut

OffSiden tapauksessa *palvelu* (service) on liitännäinen, joka toteuttaa jonkun analyysiohjelmiin suoraan liittymättömän toiminnon. Varsinaista rajoitusta palvelun tarjoamalle toiminnolle ei ole, mutta tyypillisesti se on toiminto, jota suoritetaan jatkuvasti, joka tukee jollain tavalla funktiolohkoja tai joka tukee koko järjestelmää. Palveluita voi olla kahta eri tyyppiä:

- *Globaali palvelu* on nimensä mukaisesti saavutettavissa kaikista palvelimen osista, sillä siitä on suoritettavana aina yksi instanssi globaalisti.
- *Ohjelmakohtaisesta palvelusta* luodaan instanssi erikseen jokaista ladattua analyysiohjelmaa kohtaan. Tästä syystä kutakin palvelua kohtaan on pääsy vain kyseisen ohjelman funktiolohkoista.

Palvelun tulisi olla aina julkisen rajapinnan takana muokattavuuden helpottamiseksi. Rajapinnan toteuttava palvelu on yksittäinen .NET-luokka, jonka tulee toteuttaa myös palvelimen alustamiseen ja jälkien puhdistamiseen vaadittavat metodit. Koska palvelu on aina liitännäinen, ladataan se automaattisesti palvelimen käyttöön, mikäli sen toteuttava .NET-kokoonpanotiedosto on saatavilla oikeasta kansiosta. Vastaavasti palvelu voidaan poistaa käytöstä yksinkertaisesti poistamalla kyseinen DLL-tiedosto samasta kansiosta.

OffSide sisältää valmiina muutamia yleishyödyllisiä peruspalveluita. *Lokipalvelua* (log service) voidaan nimensä mukaisesti käyttää eritasoisten lokiviestien kirjoittamiseen ja *aikapalvelu* (time service) tarjoaa pääsyn reaaliaikaiseen kellonaikaan. *Työläispalvelu* (worker service) puolestaan kontrolloi tehtäviä suorittavia työläisiä.

Osalla OffSiden palveluista on tarve päästä käsiksi OPC-kohteisiin. Tätä tarkoitusta varten OffSidessa on erillinen *OPC-kuuntelupalvelu* (OPC Listener service). Se tarjoaa metodit OPC-kohteiden muutoksien kuunteluun. Mikäli kuunneltavan kohteen arvo muuttuu, OPC-palvelin lähettää ilmoituksen kuuntelupalvelulle. Tällöin kuuntelupalvelu kutsuu kuuntelijan asettamaa käsittelijämetodia, jolla haluttu toimintalogiikka toteutetaan.

Jotta COMTRADE-häiriötallenteita voidaan prosessoida analyysiohjelmissa, täytyy ne tuoda ensin RTDB-tietokantaan. Tämän hoitamiseksi OffSide käyttää *COMTRADE-tuontipalvelua* (COMTRADE import service). Se on XML-tiedostolla konfiguroitava globaali palvelu, joka kuuntelee asetettuja OPC-tapahtumia ja tarkkailee asetettuja kansioita. Valmiit häiriötallenteet jäsennetään COMTRADEAPIa käyttäen, jonka jälkeen ne viedään RTDB-tietokantaan IED-kohtaisessa järjestyksessä vanhimmasta alkaen.

COMTRADE-tuontipalvelun XML-tiedosto sisältää yleiset parametrit, jotka kertovat onko palvelu asetettu käynnistyväksi, kuinka usein jaksotyöläinen tarkastaa OPC-ilmoitusten perusteella prosessoitavat tiedostot, mikä on ajan katkaisuraja jaksotyöläiselle sekä missä formaatissa RTDB-muuttujien nimet muodostetaan. Yleisten parametrien lisäksi tiedosto sisältää OPC-palvelimen URL:n sekä IED-nimet ja niitä vastaavat tarkkailtavat kansiot ja kuunneltavien OPC-kohteiden nimet. COMTRADE-tuontipalvelun XML-tiedostoa on havainnollistettu listauksessa 4.2, jossa on yksi esimerkki-IED nimeltä ”JKI.J05”.

```
<?xml version="1.0" encoding="UTF-8"?>
<COMTRADEImportConfiguration
  xsi:noNamespaceSchemaLocation="COMTRADEImportServiceConfiguration.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  Enabled="true" Timeout="20000" TestInterval="1000" StreamNameFormat="{1}">
  <OPCServer URL="opcda://OPC.DaWrapper.1">
    <IED Name="JKI.J05" LocalFolder="C:\COMTRADE\JKI\J05"
      OPCItemRecordingStarted="Generic/bool01"/>
  </OPCServer>
</COMTRADEImportConfiguration>
```

Listaus 4.2 *Esimerkki COMTRADE-tuontipalvelun konfiguraatiosta.*

4.3.5 Analyysiohjelmat ja funktiolohkot

OffSiden päätarkoitus on ajaa analyysiohjelmia, jotka prosessoivat tietoa sähköasemalta. Analyysiohjelmien muokattavuus halutaan pitää mahdollisimman suurena, ja toisaalta ohjelmien kehittäminen halutaan pitää mahdollisimman helppona. Näistä syistä on päädytty ohjelmien rakentamiseen *funktiolohkoista*.

Yksittäinen funktiolohko toteuttaa jonkin yksinkertaisen toiminnon. Funktiolohkojen toimintaa voidaan säätää muuttamalla niiden parametreja. Varsinainen toteutus puolestaan on tyypillisesti yksittäinen .NET-luokka, joka toteuttaa vaaditut yksinkertaiset rajapinnat. Toteuttava luokka voidaan kirjoittaa suoraan esimerkiksi C#:lla tai se voi olla suojareleen toimilohkon päälle automaattisesti generoitu .NET-kääre.

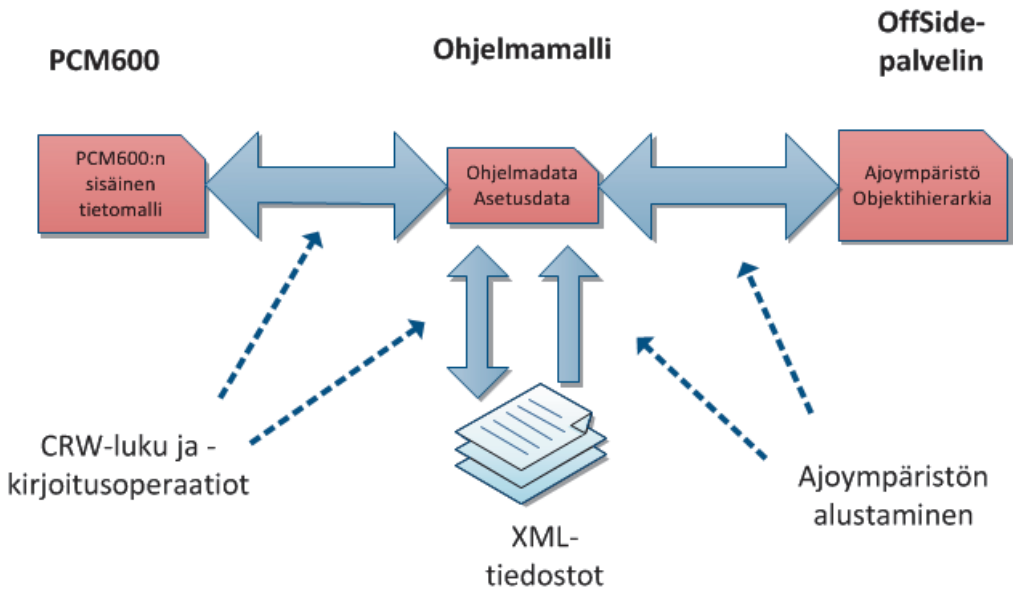
OffSidessa funktiolohkot yhdistetään funktiolohkokirjastoihin. Kaikki funktiolohkokirjastot ovat liitännäisiä, joten ne ladataan automaattisesti DLL-tiedoston ollessa oikeassa paikassa palvelimen käynnistyessä. OffSidessa on valmiina lukuisia perusfunktiolohkoja. Liitännäisyyden ansiosta funktiolohkokirjastoja voidaan kehittää ja lisätä OffSideen täysin riippumatta varsinaisen ajojärjestelmän kehityksestä. Täten myös kolmannet osapuolet voivat laajentaa funktiolohkojen tarjontaa.

Varsinaiset analyysiohjelmat rakennetaan valitsemalla halutut funktiolohkot ja määrittelemällä informaation kulku niiden välillä yhdistämällä lohkojen sisään- ja ulostuloja toisiinsa. Suoritusympäristö suorittaa lohkot niiden välisten yhteyksien mukaisessa järjestyksessä, jotta kunkin lohkon sisääntulolla on jo arvo evaluoituna. Tästä johtuen takaisinkytkennät eivät ole mahdollisia analyysiohjelmissa.

Analyysiohjelma ei käynnisty itsestään, vaan sen tulee sisältää vähintään yksi käynnistävä funktiolohko. Käynnistävä funktiolohko odottaa jonkin ehdon täyttymistä. Ehtona voi olla esimerkiksi yksittäisen signaalin arvon muuttuminen tai laskurin täyttyminen. Ehdon täyttymisen jälkeen lohko käynnistää ohjelman suorittamisen. On myös mahdollista käynnistää jokin toinen analyysiohjelma ehdon täyttyessä. Näin analyysiohjelmia yhdistelemällä voidaan rakentaa kompleksisempaa logiikkaa analyyseja varten.

4.3.6 Ohjelmamalli (application model)

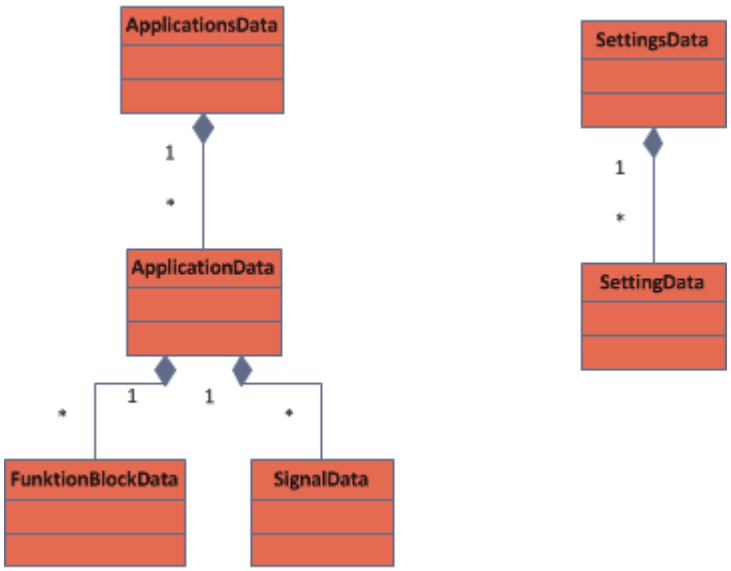
OffSidessa ajettavien analyysiohjelmien rakenne on määritelty *ohjelmamallissa* (application model). Sitä käytetään määriteltyjen analyysiohjelmien, niiden funktiolohkoinstanssien, signaalien ja parametrien arvojen siirtämiseen ja säilyttämiseen. Ohjelmamalli luodaan ConnPackissä, kun PCM600:lla määritellyt CDS-instanssidata ja PDS-parametrit muokataan ja lähetetään IED-kirjoitusoperaatiossa OffSidelle. OffSide-palvelimen vastuulla on olioinstanssien luominen ja hierarkian rakentaminen XML-tiedostojen pohjalta. PCM600:lla voidaan myös lukea ohjelmamalli OffSidestä IED-lukuoperaatiolla. Ohjelmamallin toimintaa käytännössä on havainnollistettu kuvassa 4.10.



Kuva 4.10 *Ohjelmamallin toiminta. Mukailtu lähteestä (ABB, 2012).*

Ohjelmamalli voidaan nähdä riippuvuuksia vähentävänä muurina, koska se on riippumaton OffSiden olioinstanssien toteutuksesta ja PCM600:n rajapinnoista.

Luokkakaavio ohjelmamallista on kuvassa 4.11.



Kuva 4.11 *Ohjelmamallin luokkakaavio. Mukailtu lähteestä (ABB, 2012).*

Ohjelmamallin siirtely ja varastointi sarjallisessa muodossa aiheuttaa rajoitteita mallin käytölle. Ensimmäinen haaste tulee graafimaisen mallin esittämisestä XML-puuna. Toinen haaste aiheutuu oliokuvauksien viittauksien ja parametrien arvojen säilyttämisestä tekstimuodossa. Näistä syistä johtuen olioiden luonnin yhteydessä viittausten ja parametrien arvojen todelliset tyypit on selvitettävä. Ohjelmamallin olioiden onkin tarjottava

selvitysmetodi (resolve), jotta luodut instanssit saadaan täysin vastaamaan hierarkkista mallia.

OffSide lukee ohjelmamallia vastaavat analyysiohjelmat *Application.xml*- ja *Settings.xml*-nimisistä tiedostoista. Näiden lisäksi samaan kokonaisuuteen liittyy *ApplicationGraphicalFile.xml*, joka kuvailee analyysiohjelmien visuaalisen asettelun PCM600:ssa. Tätä ohjelmamallin ulkopuolista tiedostoa ei prosessoida OffSidessa mitenkään, mutta sitä tarvitaan luettaessa konfiguraatiota takaisin PCM600:aan.

Application.xml-tiedosto koostuu ohjelmista, jotka sisältävät ohjelmaan liittyvät funktiolohkot sekä näiden väliset signaalit. Esimerkki Application.xml-tiedoston rakenteesta on kuvattu listauksessa 4.3.

```
<?xml version="1.0" encoding="utf-8"?>
<OffSideApplications xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Applications>
    <Application Name="TRIG_J05">
      <FunctionBlocks>
        <FunctionBlock    TypeName="AddMilliseconds" InstanceId="4"
          ExecutionOrder="15" Name="AddMilliseconds" />
        ...
        <FunctionBlock    TypeName="ReadSampleInt" InstanceId="1"
          ExecutionOrder="21" Name="PREV_AR_STATU" />
         ...
      </FunctionBlocks>
      <Signals>
        <Signal    FromFunctionBlockType-Name="AddMilliseconds"
            FromFunctionBlockId="4" FromParameterName="Output"
            FunctionBlockTypeName="ReadSampleInt"  ToFunctionBlockId="1"
            ToParameterName="TimeStamp" />
        ...
      </Signals>
    </Application>
    ...
  </Applications>
</OffSideApplications>
```

Listaus 4.3 *Esimerkki analyysiohjelman rakenteesta xml-muodossa.*

Listauksessa kuvataan ohjelmasta nimeltä ”TRIG_J05” kaksi funktiolohkoa, jotka ovat tyypeiltään *AddMilliseconds* ja *ReadSampleInt*. Näiden välillä menee signaali AddMilliseconds-lohkon Output-nimisestä ulostulosta ReadSampleInt-lohkon TimeStamp-nimiseen sisääntuloon.

Asettelut sisältävän XML-tiedoston hierarkia on yksinkertaisempi. Jokaisesta asettelusta löytyy asetteluun liittyvän ohjelman nimi, asettelun parametrisoitu oliotyyppinimi, funktiolohkon tunnistenumero, asettelun nimi sekä tietysti asettelun arvo. Esimerkki

edellisen kappaleen ohjelmaa vastaavasta Settings.xml-tiedostosta löytyy listauksesta 4.4.

```
<?xml version="1.0" encoding="utf-8"?>
<OffSideSettings    xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Settings>
    ...
    <Setting ApplicationName="TRIG_J05"
             ParametrizedObjectTypeName="ReadSampleInt"
             FunctionBlockId="1" Name="StreamName" Value="CB_STATUS" />
    <Setting ApplicationName="TRIG_J05"
             ParametrizedObjectTypeName="ReadSampleInt"
             FunctionBlockId="1" Name="StreamSource" Value="J05" />
  </Settings>
</OffSideSettings>
```

Listaus 4.4 *Esimerkki analyysiohjelman asetuksista XML-muodossa.*

Listaus kuvaa ReadSampleInt-lohkoon liittyvät StreamName- ja StreamSource-asettelut. AddMilliseconds-funktiolohkoon ei liity asetteluja.

4.3.7 Liitettävyyspaketti (ConnPack)

Liitettävyyspaketti (ConnPack) on PCM-kehyksen liitännäinen. Sillä voidaan tarjota kuvauksia IED:iden toiminnallisuuksista, kommunikoida IED:iden kanssa ja konfiguroida niitä. OffSide-palvelinta voidaan mallintaa IED-tyyppisenä laitteena, vaikka se ei tiukasti ottaen olekaan IED. Tätä tarkoitusta varten OffSideen on kehitetty oma liitettävyyspaketti.

Liitettävyyspaketin on määriteltävä erilaisia IED-laitteen ominaisuuksiin liittyviä tyyppidatoja, jotta sen kautta voidaan käyttää PCM600:aan liittyviä työkaluja. OffSiden tapauksessa sen oliomalli rakennetaan dynaamisesti liitännäisten metatiedoista .NETin reflektion avulla. Pääsypisteenä liitettävyyspaketin ja PCM:n välillä toimii *OffSideObjectType*, joka toteuttaa PCM-sovelluskehyksen vaatimia rajapintoja.

Tarpeettomien riippuvuuksien vähentämiseksi OffSiden liitännäiset eivät suoraan toteuta PCM600:n rajapintoja, vaan PCM600-yhteensopiva oliomalli rakennetaan liitettävyyspaketin *metatietomanagerin* (metadata manager) avulla. Se on singleton-suunnittelumallin mukainen olio, jonka vastuulla on tyyppidatan muuttaminen OffSiden ja PCM600:n välillä. Muuttamista tapahtuu, kun:

- CDS-tyyppidataa generoidaan OffSiden parametrisoiduista oliokuvauksista
- PDS-tyyppidataa generoidaan OffSiden parametrikuvauksista
- SCL-tyyppidataa generoidaan OffSiden IEC 61850 -datatyyppipohjista
- Ohjelmia ja niihin liittyviä asetuksia kirjoitetaan ja luetaan.

Liitettävyyspaketissa on toteutettu *CRWTransactionDispatcher*-luokka, jota CRW käyttää transaktioiden logiikan hoitamiseen. Se määrittelee, missä järjestyksessä työka-

lujen tulee tehdä pyyntöjä liitettävyyspaketille. CDS-data tulee lukea ennen PST-dataa, koska CDS-dataa tarvitaan PST-datan globaalien parametrien nimien rakentamiseen. Muuten työkalujen operaatioiden järjestyksellä ei ole merkitystä. Aluksi CRWTransactionDispatcher ottaa yhteyden OffSide-palvelimelle käyttäen *OffSideClient*-oliota paljastaen kyseisen olion muille työkaluille tiedonsiirron mahdollistamiseksi. Samalla tehdään versionumeroiden tarkistus ConnPackin ja palvelimen välillä. Lukuoperaatio ei vaadi erityistä transaktiota, sillä palvelimen ei epäonnistuneen operaation yhteydessä tarvitse palata mihinkään tiettyyn tilaan. Kirjoitusoperaatio puolestaan pakottaa OffSide-palvelimen käynnistymään uudestaan, jotta uusi konfiguraatio saadaan ladattua.

4.3.8 Tietovirrat

Sähköasemalla data voi tulla lukuisista eri lähteistä, kuten esimerkiksi IED:istä erilaisten protokollien yli, eri sähköasemilta tai suoraan tietokannoista. Tätä varten OffSidessä datan prosessointi on abstrahoitu tietovirtojen (streams) taakse. Tietovirrat pyrkivät piilottamaan analyysiohjelmien kehittäjiltä kuhunkin datalähteeseen liittyvät ominaispiirteet ja erilaiset käsittelytavat tarjoamalla yhtenäisen rajapinnan datan prosessointiin.

Analyysiohjelman näkökulmasta tietovirrat voidaan jakaa kahteen kategoriaan:

- *Sisääntulotietovirrasta* (input stream) ohjelma lukee dataa prosessointia varten
- *Ulostulotietovirtaan* (output stream) ohjelma kirjoittaa dataa, kuten prosessoinnin tuloksia.

Erilaisista datanlähteistä johtuen OffSiden käyttöön on kehitetty kaksi eri tietovirtaperhettä:

- *Näytetietovirrat* (sample streams), jotka koostuvat aikaleimatuista skalaarinäytteistä. Esimerkkinä näytetietovirrasta on häiriötallenteen yksittäisen kanavan arvo tietyllä ajanhetkellä.
- *Tallennetietovirrat* (record streams), jotka koostuvat avaimella tunnistettavista tallenteista. Yksittäinen tallenne puolestaan on useampia nimettyjä kenttiä sisältävä monikko. Esimerkkinä tallennetietovirrasta on relaatiotietokannan yksittäinen taulu.

Sekä näytetietovirrat että tallennetietovirrat voivat olla niin sisääntulotietovirtoja kuin ulostulovirtojakin. Vaikka sama datanlähde voi toimia sekä sisääntulona että ulostulona, se mallinnetaan OffSiden tapauksessa kuitenkin kahtena erillisenä tietovirtana riippuen datan kulkusuunnasta.

Jokaisen tietovirran tulee tietenkin olla tunnistettavissa. Yhtenäisenä tunnistamismekanismina jokaiselle tietovirralle käytetään yksinkertaista *tietovirrantunnistetietuetta* (streamId), joka sisältää lähteestä (source) ja nimestä (name) koostuvat kentät. Molemmat kentät ovat tekstimuotoisia, ne eivät saa olla tyhjiä ja niiden yhdistelmän tulee olla yksikäsitteinen. Muita varsinaisia rajoituksia tunnistamistietueille ja niiden kenttien syntakseille ei ole. Kuitenkin suositeltu tapa tunnisteen muodostamiselle on käyttää lähdettä osoittamaan datan alkupiste ja nimeä indikoimaan haluttu datapiste. Kun tätä tapaa sovelletaan järjestelmällisesti, eri lähteestä tulevalla samanlaisella datalla on yhte-

näinen nimiosio tunnistetietueessa. Näin vaihtaminen datanlähteestä toiseen onnistuu yksinkertaisesti vaihtamalla vain tunnistetietueen lähdeosio.

Erilaisten datalähteiden mahdollistamiseksi OffSidessa käytetään *tietovirrantarjoajia* (stream providers). Tietovirrantarjoajan vastuulla on tietovirtojen rakentamiseen tarkoitettujen tehdasmetodien toteuttaminen. Tarjoajan tulee myös hoitaa mallintaminen tietovirrantunnistetietueen ja datanlähteessä olevan varsinaisen datapisteen nimen välillä.

OffSidessa on valmiina kaksi eri tietovirrantarjoajaa: muistitietovirrantarjoaja ja RTDB-tietovirrantarjoaja. Muistitietovirrantarjoaja mahdollistaa pääsyn muistissa oleviin näytetietovirtoihin, joiden sisältö häviää, kun järjestelmä suljetaan. RTDB-tietovirrantarjoaja taas tukee sekä näyte- että tallennetietovirtoja nimensä mukaisesti RTDB-tietokannasta. Näytetietovirrat vastaavat RTDB:n muuttujia, ja tallennetietovirrat RTDB:n kustomoitavien taulujen rivejä ja niiden kentät tauluihin liittyviä sarakkeita. Jotta vältytään kovakoodatuilta kuvauksilta tietovirrantunnisteiden ja tietokannan muuttujien sekä taulujen välillä, RTDB-tietovirrantarjoajassa on jätetty mahdollisuus loppukäyttäjälle kuvauksen määrittelemiseen XML-tiedostossa. Esimerkki tästä on listauksessa 4.5.

```
<ProviderProperty Name= ”VariableNameFormat” Value=”{0}.{1}” />
```

Listaus 4.5 *Esimerkki RTDB-tietovirrantarjoajan kuvauksesta.*

Kun esimerkkitapauksessa tältä tarjoajalta kysytään tietovirtaa, jonka lähde on ”JKI_J05” ja nimi ”IL1”, saadaan vastaukseksi tietovirta, joka tarjoaa pääsyn RTDB:n muuttujaan JKI_J05.IL1. Kuvauksessa {0} vastaa siis tunnisteen lähdeosiota ja {1} tunnisteen nimiosiota.

Näytetietovirroista tapahtuvaan arvojenlukemiseen tarvitaan aikaleima tai aikaleimaväli. Sähköasema toimintaympäristönä puolestaan aiheuttaa haasteita tietojen lukemisen ajoituksen suhteen, sillä vain osa asemalla olevista tietoa tuottavista laitteista on reaaliaikaisia. Koska OffSide on suunniteltu toimimaan historiatietokannasta luettavalla datalla, mutta eri lähteistä saapuva data saattaa saapua eri ajoituksilla, näytetietovirtojen yhteyteen on kehitetty erillisiä *ajoituksentarjoajia* (timing providers). Tällä hetkellä tarjolla on reaaliaikaa seuraava ajoituksentarjoaja, joka seuraa asematietokoneen Windowsin kelloa, sekä COMTRADE-ajoituksentarjoaja, joka pyrkii varmistamaan että koko COMTRADE-tallenne on saatu tallennettua tietokantaan. Uudesta häiriötallenteesta tulevan ilmoituksen yhteydessä COMTRADE-ajoituksentarjoaja väliaikaisesti estää keskeneräisen häiriötallenteen prosessoinnin. Kun COMTRADE-tallenne on kokonaisuudessaan tietokannassa, siirtyy COMTRADE-ajoituksentarjoaja takaisin normaaliin toimintaan. Normaalitilassa COMTRADE-ajoituksentarjoaja toimii kuten reaaliaikaa seuraava ajoituksentarjoaja pienellä viiveellä lisättynä.

Koko tietovirtakonsepti on toteutettu *tietovirtapalveluun* (streams service), joka on konfiguroitavissa erillisellä XML-tiedostolla. Esimerkki tietovirtapalvelun konfiguraatiotiedostosta on listauksessa 4.6.

```
<StreamsServiceConfiguration
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <StreamProviderClass
    Class="ABB.OffSide.Plugins.RTDBStreamProvider.RTDBStreamProvider,
    ABB.OffSide.Plugins.RTDBStreamProvider" ID="LocalRTDB">
    <ProviderProperty Name="DSN" Value="" />
    <ProviderProperty Name="Username" Value="" />
    <ProviderProperty Name="Password" Value="" />
  </StreamProviderClass>
  <TimingProviderClass
    Class="ABB.OffSide.Plugins.COMTRADEImportService.COMTRADETimingProvider,
    ABB.OffSide.Plugins.COMTRADEImportService" ID="RTDB_from_COMTRADE">
    <ProviderProperty Name="Offset" Value="5000" />
  </TimingProviderClass>
  <TimingProviderClass
    Class="ABB.OffSide.Common.Streams.FollowRealTimeTimingProvider,
    ABB.OffSide.Common.Streams" ID="RTDB_from_OffSide">
    <ProviderProperty Name="Offset" Value="2000" />
  </TimingProviderClass>
  <StreamMapping StreamProviderID="LocalRTDB"
    TimingProviderID="RTDB_from_OffSide"
    SourcePattern="^OffSide$" NamePattern="*">
    <ProviderProperty Name="TableNameFormat" Value="{0}_{1}" />
  </StreamMapping>
  <StreamMapping StreamProviderID="LocalRTDB"
    TimingProviderID="RTDB_from_COMTRADE"   SourcePattern="JKI.J05"
    NamePattern="(AnalogChannel)[1-9][0-9]*$">
    <ProviderProperty Name="VariableNameFormat" Value="{0}.{1}" />
  </StreamMapping>
</StreamsServiceConfiguration>
```

Listaus 4.6 *Esimerkki tietovirtapalvelun konfiguraatiosta.*

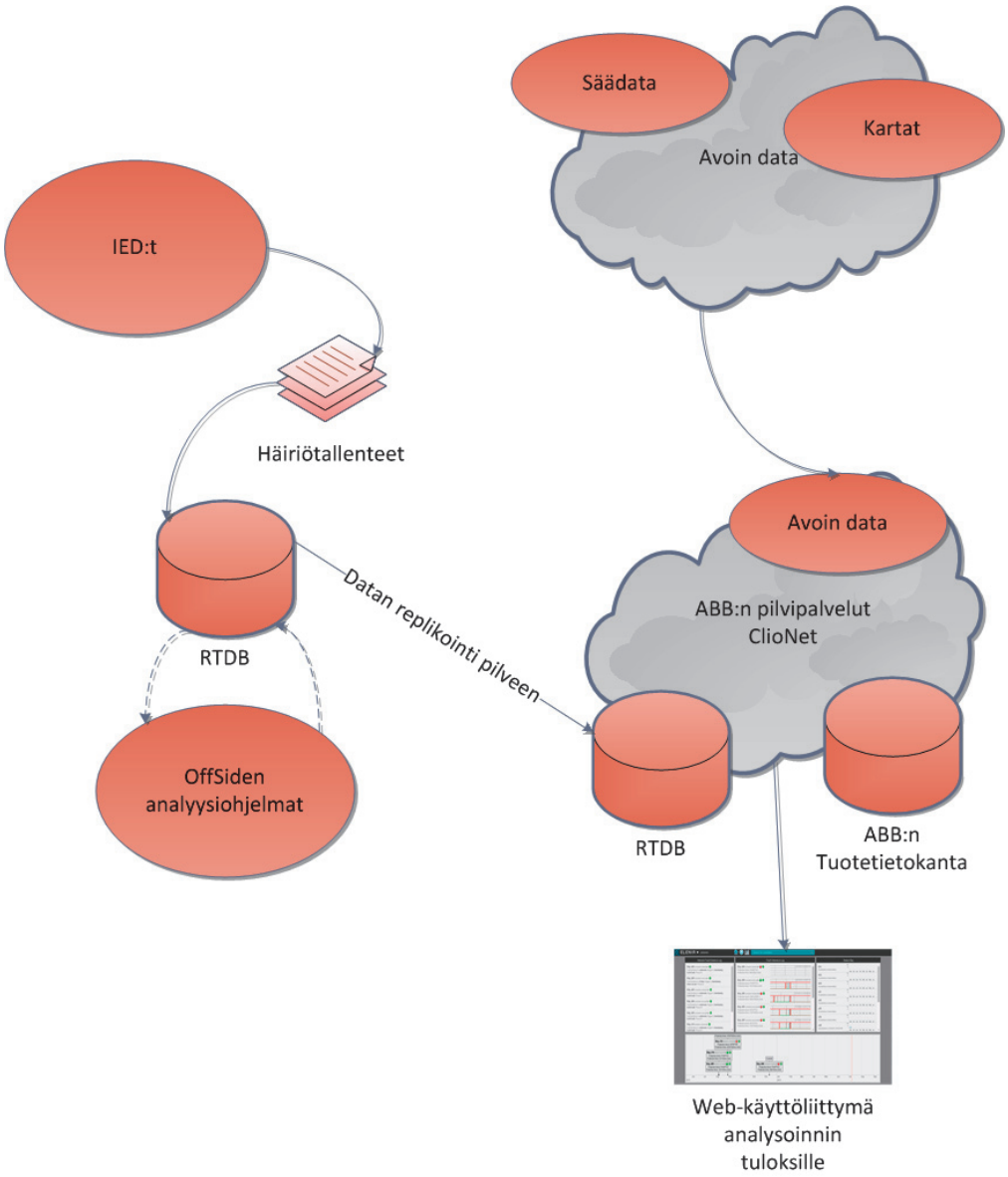
Listauksessa 4.6 on tietovirrantarjoajana RTDB-tietovirrantarjoaja ja ajoituksentarjoajina sekä COMTRADE-ajoituksentarjoaja että reaaliaikaa seuraava ajoituksentarjoaja. Lisäksi listaus sisältää *tietovirtakuvauksia* (stream mapping), jotka vastaavat varsinaisten tietovirtojen mallintamisesta. Tietovirtakuvaus yhdistää tietovirrantarjoajan, ajoituksentarjoajan, tietovirrantunnisteen ja mahdollisen tietolähdekohtaisen nimiformaatin kuvauksen. Tunnisteen lähde- ja nimiosiot voivat olla .NET-sovelluskehyksen säännöllisten lausekkeiden (regexp, regular expression) (Microsoft, 2014b) mukaisia. Säännölliset lausekkeet ovat kuvauksia merkkijonojen rakenteista.

5 PILOTTIHANKE ELENIA OY:N KANSSA JA VAATIMUKSET KONFIGUROINTITYÖKALULLE

Elenia Oy on sähköverkkoyhtiö, joka toimii yli sadan kunnan alueella Kanta- ja Päijät-Hämeessä, Pirkanmaalla, Keski-Suomessa sekä Etelä- ja Pohjois-Pohjanmaalla. Sillä on asiakkaina noin 410 000 kotitalous-, yritys- ja yhteiskunta-asiakasta. (Elenia, 2014) Elenialla on arviolta 140 ensisijaista sähköasemaa, joista jokaisessa on keskimäärin 8 lähtöä. Sähköasemien lisäksi Elenialla on noin 5000 ohjattavaa katkaisijaa. Kun arvioidaan yhdellä lähdöllä olevan muutamia kymmeniä datapisteitä ja katkaisijalla viidestä kymmeneen datapistettä, saadaan datapisteiden arvioiduksi kokonaismääräksi noin 100 000. (Puurtinen, 2014)

Älykkäät sähköverkot ja energiamarkkinat (Smart Grids and Energy Markets, SGEM) on suomalainen tutkimusohjelma, jonka tavoitteena on kehittää kansainvälisiä, käytännön ympäristössä demonstroitavia, älysähköverkkoratkaisuja suomalaista tutkimus- ja kehitysinfrastruktuuria hyödyntäen (CLEEN, 2014). Osana SGEMiä toteutetaan pilottihanke, jossa on tarkoitus testata analysointikonseptia Elenian datalla. Pilottihanke jakautuu kolmeen vaatimustasoltaan kasvavaan vaiheeseen, jotta hanke saadaan liikkeelle mahdollisimman vauhdikkaasti ja jotta suuriin datamääriin liittyvät ongelmat pysyvät mahdollisimman pieninä. Ensimmäisessä vaiheessa, johon tämä työ kuuluu, ajetaan ja testataan järjestelmää Elenian tarjoamilla yhden sähköaseman häiriötallenteilla. Kun perustoiminnallisuus on kunnossa ja kaikki osapuolet ovat tyytyväisiä, siirrytään käyttämään reaaliaikaista dataa yhdeltä sähköasemalta asentamalla järjestelmä Elenian palvelimille. Lopulta viimeisessä vaiheessa on tarkoituksena siirtyä tukemaan useita sähköasemia.

Pilottihankkeeseen liittyy tämän työn lisäksi kaksi muuta ABB Oy:lle tehtävää diplomityötä. Joonas Puurtisen työ (Puurtinen, 2014) keskittyy lähtökohtaisten analyysiohjelmien kehittämiseen sekä nimeämiskäytäntöjen miettimiseen. Toisen diplomityön vastuulla on miettiä analyysien tulosten visualisointia web-käyttöliittymän kautta. Havainnollistava kuva pilottihankkeen suunnitellusta järjestelmästä on esiteltynä kuvassa 5.1.



Kuva 5.1 *Pilottihankkeen järjestelmän suunniteltu rakenne. Mukailtu lähteestä (ABB, 2013d).*

Järjestelmä saa prosessoitavan datansa Elenian Jokioisten sähköasemalta haetuista häiriötallenteista. Häiriötallenteet ovat järjesteltynä sähköaseman lähtöjen mukaisesti, joten niiden tallentaminen oikeassa muodossa tietokantaan on mahdollista. Häiriötallenteiden lisäksi järjestelmässä hyödynnetään konfigurointidataa, joka sisältää esimerkiksi suojauslohkon asetetut toiminta-ajat. Myös avoimen datan, kuten säätietojen ja karttojen, hyödyntämistä tutkitaan.

ABB:n ClioNet on suojareleille tarkoitettu tiedonjako- ja varmuuskopiointipalvelu. Pilottihankkeen puitteissa dataa voitaisiin replikoida myös pilveen. Tähän tarkoitukseen on kaavailtu edellä mainittua ClioNetiä, sillä se on jo olemassa oleva pilvipalvelu. Sinne voitaisiin replikoida niin prosessoimatonta dataa kuin analyysien tuloksiakin.

5.1 Pilottihankkeen analyysiohjelmat

Jotta pilottihanke saadaan mahdollisimman nopeasti käyntiin ja jotta saataisiin aikaiseksi samalla konkreettisia tuloksia, on pilottihankkeen ensimmäiseen vaiheeseen valittu toteutettavaksi vain kaksi analyysiohjelmaa. Nämä ovat *vian paikallistamisen analyysi* sekä *suojausten toiminta-ajan analyysi*.

5.1.1 Vian paikallistamisen analyysi

Vian paikallistamisen analyysiohjelma pyrkii nimensä mukaisesti laskemaan etäisyyden linjalla olevaan vikaan. Ohjelmakokonaisuus koostuu kahdesta erillisestä ohjelmasta: laukaisijasta ja laskennasta. Laukaisijaosa odottaa sille määritellyn signaalin nousemista ylös. Kun näin tapahtuu, se laukaisee laskentaosan. Varsinainen laskenta koostuu puolestaan kahdesta vaiheesta. Ensimmäisessä vaiheessa tunnistetaan vian tyyppi, joka voi olla yksivaiheinen maavika, vaiheesta vaiheeseen oleva oikosulku sekä kolmivaiheinen oikosulku. Vian tyypin perusteella valitaan sille sopiva laskentametodi.

Edellä mainitun laukaisijasignaalin lisäksi ohjelma vaatii sisääntuloikseen häiriötallenteista kolme jännite- ja virtakanavaa laskentaa varten. Ulostulona onnistuneen laskennan jälkeen ohjelma tuottaa rivin OffSide_faultpackage-nimiseen tauluun. Yksittäinen rivi sisältää seuraavat tiedot:

- lähettävän loogisen noodin tunniste
- IED-nimi
- liipaisutyyppi
- vikatyyppi
- katkaisun kokonaiskesto
- katkaisujen lukumäärä
- vian vaihe
- vikavirtojen suuruudet
- toteutetusta metodista
- vikaetäisyys
- arvion laskennan onnistumisesta sekä
- tapahtuman ajankohta.

5.1.2 Suojausten toiminta-ajan analyysi

Suojausten toiminta-ajan analyysiohjelman vastuulla on tutkia, mikä suojaustoiminto oli aktiivisena vikatilanteessa ja kuinka nopeasti se suoriutui verrattuna sille aseteltuun toiminta-aikaan. Vian paikallistamisohjelman tavoin tämäkin ohjelma koostuu laukaisijaosasta ja analyysiosasta. Laukaisijaosa odottaa uuden rivin ilmaantumista comtrade-nimiseen tauluun. Mikäli ilmaantuneella rivillä on ohjelmalle aseteltua IED-nimeä vas-

taava IED-nimi, käynnistää laukaisija analyysiosan rivistä luetulla tallenteen alkuajanhetkellä.

Analyysiosa aloittaa toimintansa tutkimalla, onko katkaisija auennut tallenteessa. Mikäli katkaisija on auennut, siirrytään tutkimaan, minkä suojausfunktion toimintasignaali (operate) on ollut aktiivinen. Oikean suojausfunktion löydyttyä lasketaan suojauksen toiminta-aika katkaisijan aukeamisen ja suojausfunktion aloitusajan välisenä erotuksena. Laskettua tulosta verrataan sille aseteltuun toiminta-aikaan. Mikäli toiminta-aika eroaa liikaa asetetusta, nostetaan joko varoitus tai hälytys riippuen erosta.

Comtrade-taulun lisäksi analyysiohjelman sisääntuloina on katkaisimen tilasta kertovat auki- ja kiinnisignaalit sekä maksimissaan 13 erilaista suojausfunktion käynnistys- ja toimintasignaaliparia. Laskennan onnistuttua ohjelman ulostulona on uusi rivi Off-Side_fpt-nimiseen tauluun. Yksittäinen rivi taulussa sisältää seuraavat tiedot:

- sähköaseman nimi
- lähdön tunniste
- suojalohkon nimi
- suojalohkon toiminta-aika
- suojalohkolle asetettu toiminta-aika
- hälytyksen tila sekä
- katkaisuajankohta.

5.2 Nimeämiskäytännöt

Joonas Puurtisen diplomityössä (Puurtinen, 2014) esiteltiin ehdotelma yksikäsitteiselle nimeämiskäytännölle. Se perustuu IEC 61850- ja IEC 81346 -standardeille ja koostuu useammasta pisteellä erotetusta kentästä. Nimeämiskäytäntöä on havainnollistettu listauksessa 5.1.

```
Maa.Yhtiö.AsemaID.Jännitetaso.Lähtö.Laite.Signaali
```

Listaus 5.1 *Esitetty nimeämiskäytäntö.*

Nimiensä mukaisesti maa identifioi, mistä maasta signaali tulee, ja yhtiö identifioi minkä yhtiön laitteet tuottavat signaalin. AsemaID:llä puolestaan tunnistetaan kyseessä oleva sähköasema. Jännitetasolla ja lähdöllä identifioidaan, mistä sähköaseman lähdöstä on kyse. Yleensä jännitetaso ja lähtö voidaan yhdistää käyttämällä yhtä kirjainta yksilöimään jännitetaso ja juoksevaa numerointia yksilöimään jännitetasojen lähdöt. Laite identifioi mistä fyysisestä laitteesta, kuten suojareleestä, signaali tulee, ja signaalilla tunnistetaan nimensä mukaisesti signaali.

Pilottihankkeen ensimmäisessä osassa keskitytään yhden sähköaseman ratkaisuihin. Tästä syystä nimeämiskäytäntöä sovellettiin käyttämällä AsemaID:n, jännitetason ja lähdön yhdistelmää sekä signaalia. Esimerkiksi Jokioisten sähköaseman 20 kV:n viidennen lähdön DPHLPDOC_OP-signaali kulkee analyysiohjelmissa nimellä

”JKI.J05.DPHLPDOC_OP”. Tämä nimeämiskäytännön taso riittää itse asiassa pilotti-hankkeelle yleisestikin, sillä hankkeessa tarkastellaan vain yhden yhtiön suojareleitä yhden maan sisällä.

5.3 Järjestelmän konfiguroiminen

Tässä kohdassa kuvataan, miten järjestelmä konfiguroitiin diplomityön alkuvaiheessa. Alkutilannetta tutkimalla etsittiin varsinaiselle toteutukseen liittyviä vaatimuksia, jotka kuvataan kohdassa 5.5.

Järjestelmän konfiguroiminen aloitettiin tutkimalla Elenialta saatuja Jokioisten sähköaseman COMTRADE-häiriötallenteita. Analysoitaviksi valittiin niiden lähtöjen tallenteet, joista löytyivät sekä cfg- että dat-tiedostot. Jokainen cfg-tiedosto käytiin läpi, ja niissä olevista kanavanimistä poistettiin skandinaaviset erikoismerkit ja kaikki tyhjät välit korvattiin ’_’-merkillä. Näin saatiin aikaisesti paremmin tietokannan muuttujiksi sopivat kanavanimet.

RTDB-muuttujien generoimisen helpottamiseksi OffSiden yhteydessä on kehitetty muutamia BAT- ja Perl-skriptitiedostoja. BAT-tiedostot ovat komentorivin kautta ajettavia komentojonoja Windowsissa, kun taas Perl on tulkattava skriptimäinen ohjelmointikieli. COMTRADE-tiedostojen lukemiseen käytettiin muokattua Perl-skriptiä, joka käy cfg-tiedoston läpi ja luo muuttujien luomiseen tarvittavat SQL-komennot omaan tiedostoonsa. Kyseinen SQL-tiedosto ajettiin muokatulla BAT-skriptillä, jotta tietokantaan saatiin luotua COMTRADE-kanavien mukaiset muuttujat.

Analyysiohjelmien monistamiseksi tehtiin tämän diplomityön puitteissa palvelu ohjelmien monistamiseen (Application Cloning Service), joka on kuvattu tarkemmin kohdassa 6.1. Tätä palvelua käytettiin analyysiohjelmien monistamiseen jokaiselle lähdölle. Monistamisessa otettiin huomioon Elenialta saadut lähtökohtaiset asettelut, mikäli niissä oli eroavaisuuksia. Tiedot Elenian asetteluista olivat useammassa eri Excel-taulukossa.

Jotta monistetut analyysiohjelmat saataisiin suoritukseen OffSidessa, täytyi konfiguroida vielä tietovirta- ja COMTRADE-tuontipalvelut. Molemmat palvelut konfiguroitiin manuaalisesti muokkaamalla XML-tiedostoja tekstieditorilla. Käytännössä tämä tarkoitti yhdelle lähdölle kehitetyn ohjelman vaatimien konfiguraatioiden kopioimista ja muokkaamista ottaen huomioon lähtökohtaiset erot, kuten lähtöjen tunnisteet. COMTRADE-tuontipalvelu asetettiin tarkkailemaan paikallisen tietokoneen lähtökohtaisia kansiorakenteita sekä kuuntelemaan geneeristä OPC-palvelinta, sillä tarvittavia OPC-signaaleja ei ollut saatavilla.

Kun OffSide analyysiohjelmineen oli käynnissä, käynnistettiin häiriötallenteiden tuonti RTDB-tietokantaan kopioimalla häiriötallenteet tarkkailtaviin lähtökohtaisiin kansioihin. Uuden tallenteen ilmaantuminen tarkkailtavaan kansioon havahdutti COMTRADE-tuontipalvelun, joka aloitti tallenteen tietojen tallentamisen lähtökohtaisiin muuttujiin tietokannassa. Käsiteltyään yksittäisen tallenteen tuontipalvelu kirjoitti tietokannan kustomoituun comtrade-nimiseen tauluun häiriötallenteen tiedostopolun, alkuajan, loppuajan sekä häiriötallenteen tuottaneen IED:n nimen. Uuden rivin ilmaan-

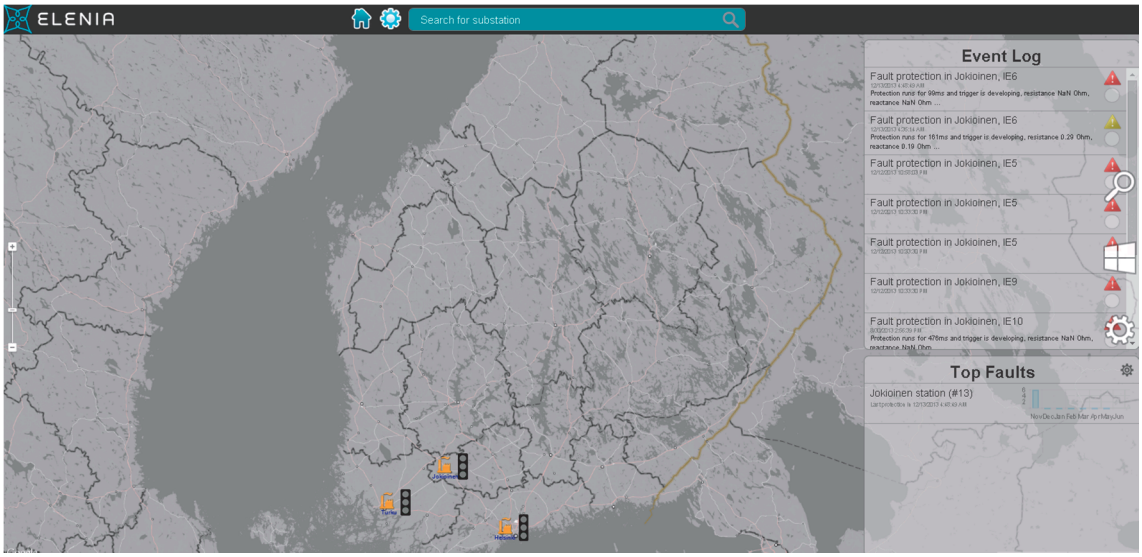
tuminen tauluun käynnisti puolestaan IED-nimeä vastaavan suojausten toiminta-ajan analyysiohjelman. IED-nimeä vastaava vian paikallistamisen analyysiohjelma käynnistettiin manuaalisesti OffSiden monitorointiohjelmalla, vaikka suojausten toiminta-ajan analyysiohjelman kaltaisen automaattisen käynnistymislogiikan toteuttaminen olisikin ollut mahdollista.

Seuravaksi kun kokonaisuus oli konfiguroitu ja testattu toimivaksi paikallisella tietokoneella, oli aika siirtyä ajamaan samat analyysiohjelmat pilottihanketta varten pystytetylle palvelimelle. Pilottihankkeen palvelimelle siirrettiin täysin sama versio OffSidesta kuin mikä oli paikallisella koneella. Tulosten säilömiseksi palvelimelle asennettiin oma, uudempi versio RTDB:stä. Lopulta analyysiohjelmat ajettiin pilottihankkeen palvelimella ja varmistettiin, että tulokset vastasivat paikallisen koneen omia.

5.4 Tulosten visualisointi

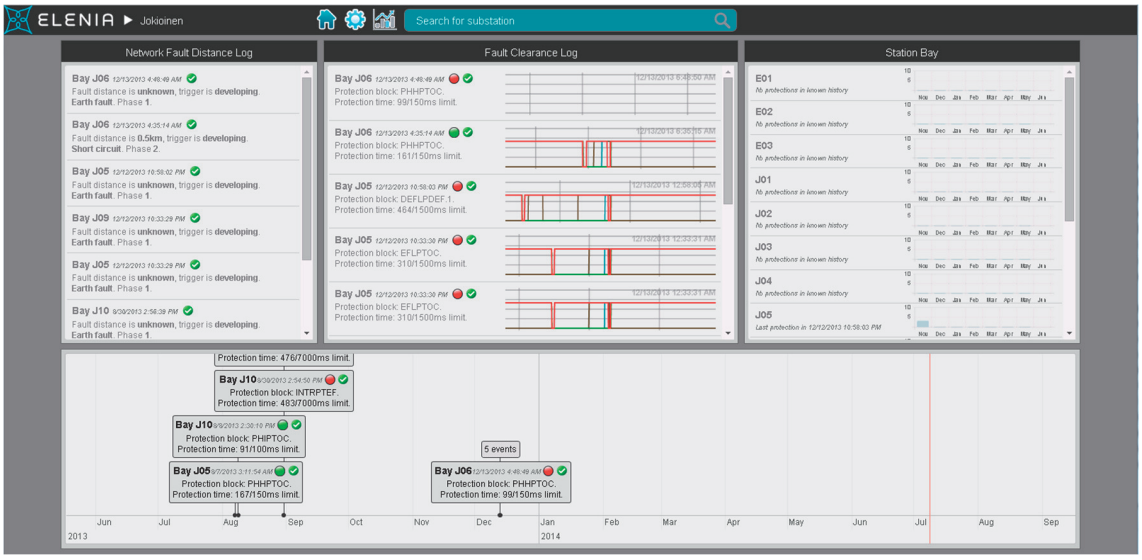
Analyysiohjelmien tulokset täytyy tietenkin saada visualisoitua loppukäyttäjälle. Tätä tarkoitusta varten pilottihankkeessa on kehitetty ehdotelma web-pohjaisesta tulosten visualisoinnista. Visualisointi perustuu cpmPlus-tuoteperheessä olevaan web-käyttöliittymään. Käyttöliittymä saa datansa suoraan pilottihankkeen palvelimella olevasta RTDB-tietokannasta. Käyttöliittymä koostuu päänäkymästä, sähköasematason näkymästä sekä sähköasematason raportointinäkymästä. Lisäksi näkymien välillä tapahtuvien siirtymien helpottamiseksi käyttöliittymän yläosassa on työkalupalkki.

Päänäkymä koostuu kolmesta osasta: sähköasemien sijaintia havainnollistavasta kartasta, viimeaikaisten tapahtumien lokista sekä eniten vikoja sisältävien sähköasemien listasta. Liikuteltavassa ja suurennettavassa kartassa on jokaisen sähköaseman kohdassa liikennevalot, jonka vihreä valo indikoi kaiken olevan kunnossa, keltainen valo ilmaisee sähköasemalla olevan varoituksen ja punainen valo puolestaan kertoo asemalla olevan hälytyksen. Tapahtumalokissa näytetään tiedot viimeisimmistä tapahtumista ja niiden vakavuudesta. Käyttäjä voi halutessaan kuitata tapahtumia luetuiksi. Eniten vikoja sisältävän listan tarkoituksena puolestaan on kiinnittää käyttäjän huomio ongelmallisimpiin sähköasemiin. Listassa on aseman nimen ja vikojen kokonaismäärän lisäksi näkyvillä pieni vikojen kuukausikohtaista kertymää havainnollistava pylväsdiagrammi. Käyttöliittymän päänäkymästä on havainnollistava kuva 5.2.



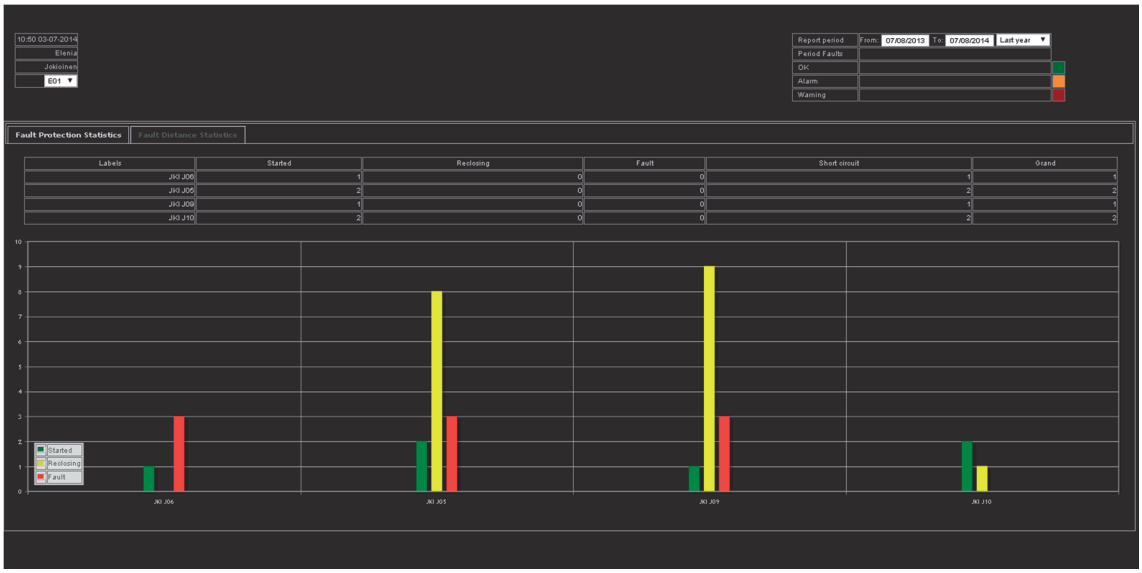
Kuva 5.2 *Pilottihankkeen käyttöliittymän päänäkymä.*

Sähköasematason näkymässä esitetään analyysiohjelmien tulokset vikaetäisyyslistassa (Network Fault Distance Log) ja viansuojauslistassa (Fault Clearance Log). Vikaetäisyyslista näyttää viimeisimmät lasketut vikaetäisyydet tyyppeineen. Viansuojauslistassa taas on suojauksen toiminta-ajan analyysiohjelman laskemat toiminta-ajat sekä toiminta-ajan kestoa suhteessa suojauksen asetettuun arvoon indikoivat varoitusvalot. Näiden lisäksi viansuojauslistassa on kutakin suojausoperaatiota vastaava diagrammi, joka sisältää suojauslohkon käynnistys- ja operointisignaalit sekä katkaisimen toimintasignaalit. Sähköasematason näkymässä on myös lista, jossa tapahtumat on jaoteltu lähtökohtaisesti, sekä aikajana, jossa kaikki aseman tapahtumat ovat esillä. Aikajanalla pystytään paremmin havainnollistamaan samaan vikaan liittyvät tapahtumat. Esimerkki sähköasematason näkymästä on kuvassa 5.3.



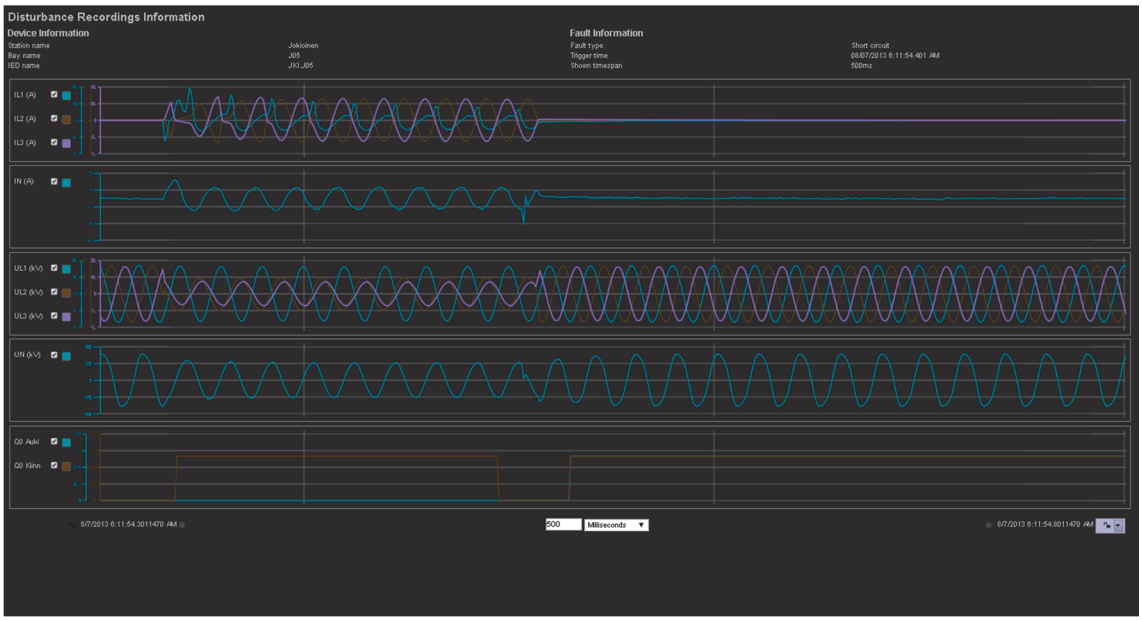
Kuva 5.3 *Sähköasematason näkymä.*

Sähköasematason työkalupalkin raportointinapista avautuu kyseisen sähköaseman raportointinäkymä. Sen tarkoituksena on tarjota sähköasemilla tehtäviä koestusraportte- ja vastaava näkymä. Raportointinäkymässä käyttäjä voi valita raportoinnin aikavälin sekä raportoitavan lähdön. Varsinainen raportti sisältää tilastot vikaetäisyyksistä sekä suojalohkojen toiminta-ajoista. Kuvassa 5.4. on esimerkki raportointinäkymästä.



Kuva 5.4 *Raportointinäkymä.*

Painamalla sähköasematasolla jonkin tapahtuman diagrammia avautuu kyseiseen tapahtumaan liittyvä COMTRADE-katselunäkymä. Se sisältää häiriötallenteen virta- ja jännitekanavat sekä katkaisijan aukeamista ja sulkemista kuvaavat signaalit. Kuva 5.5 havainnollistaa COMTRADE-häiriötallenteen katselunäkymää.



Kuva 5.5 *COMTRADE-häiriötallenteen näkymä.*

5.5 Vaatimukset konfigurointityökalulle

Diplomityötä aloitettaessa ABB:llä oli infrastruktuuri valmiina analyysiohjelmien kehittämiselle ja suorittamiselle. Valmiina ei kuitenkaan ollut mahdollisuutta konfiguroida valmista suoritettavaa pakettia, vaan eri palveluiden konfigurointi tuli tehdä manuaalisesti, kuten Elenian pilottihankkeen alkuvaiheessa. Yksittäiselle lähdölle kehitetylle analyysiohjelmalle tämä ei ole vielä kovinkaan haastavaa, mutta kun ohjelmien, lähtöjen ja sähköasemien määrä kasvaa, kasvaa myös manuaalisen konfiguroinnin työläys. Tätä taustaa vasten syntyi tarve erilliselle konfigurointityökalulle, jolla suoritettavan paketin tekeminen olisi mahdollisimman vaivatonta.

Yleisesti vaivattomuus konfigurointityökalussa tarkoittaa mahdollisimman suurta automatisointia. Käyttäjällä tulisi kuitenkin olla mahdollisuus vaadittujen asioiden muokkaamiseen, eli vaatimuksena on graafinen työkalu. Yhtenäisyyden vuoksi konfigurointityökalun tulisi hyödyntää jo olemassa olevia järjestelmiä ja palveluita. Tämä vaatimus puolestaan käytännössä pakottaa työkalun toteutusympäristöksi .NET-sovelluskehyksen.

OffSidessa suoritettavan paketin on sisällettävä vähintään ohjelmakonfiguraatio (Application.xml), ohjelmien asetukset (Settings.xml) sekä sisään- ja ulostulojen konfiguraatio tietovirtojen muodossa (StreamsService.xml). Mikäli analyysiohjelmille halutaan tuottaa uutta dataa analysoitavaksi, täytyy työkalun tuottaa myös COMTRADE-tuontipalvelun konfiguraatio (COMTRADEImportServiceConfiguration.xml). Koska COMTRADE-standardissa on määritelty kanavien nimeäminen varsin löyhästi, eivät kanavien nimet ole välttämättä kovin yhtenäisessä muodossa. Tästä syystä konfigurointityökalu voisi tarjota mahdollisuuden COMTRADE-kanavien ja ohjelmien vaatimien signaalien väliseen mallintamiseen, eli COMTRADE-kanavakonfiguraation (COMTRADEChannelConfiguration.xml) luomiseen.

Jo kehitettyjä analyysiohjelmia tulisi olla mahdollista ajaa useammassa eri lähdössä ja useammalla eri sähköasemalla. Tämä luo tarpeen analyysiohjelmien valitsemiselle ja monistamiselle. Monistettaessa tulisi ottaa huomioon myös lähtökohtaiset eroavaisuudet. Koska työkalun käytön tulisi olla mahdollisimman vaivatonta, monistamiseen vaadittavien asematunnusten ja lähtöjen nimien lukemisen tulisi olla automaattista ja varsinaisen monistaminen voisi tapahtua myös automaattisesti käyttäjän valitsemien lähtöjen ja ohjelmien mukaan.

6 TOTEUTUS

Elenian pilottihankeen alkuvaiheessa kerättyjen vaatimuksien (kohta 5.5) perusteella tarve konfigurointityökalulle oli ilmeinen. Tämä luku kuvaa sen toteutukseen liittyvät asiat. Kohdassa 6.1 esitellään yleiset komponentit ja muutokset nykyiseen järjestelmään. Kohta 6.2 puolestaan käy läpi työn puitteissa kehitetyn konfigurointityökalun. Lopulta kohdassa 6.3 arvioidaan syntynyttä toteutusta.

6.1 Yleiset komponentit ja muutokset nykyiseen järjestelmään

Konfigurointisovellus hyödyntää useita diplomityön puitteissa kehitettyjä tukikomponentteja. Tässä kohdassa esitellään näistä oleellisimmat. Jokaista tukikomponenttia voidaan käyttää myös OffSide-ajoympäristössä ilman konfigurointisovellusta. Tukikomponenttien lisäksi tässä esitellään myös merkittävimmät muutokset nykyiseen järjestelmään.

6.1.1 Analyysiohjelmien monistaminen

Analyysiohjelmien monistaminen on abstrahoitu kahden eri tason rajapinnan taakse. Matalan tason rajapintana toimi *ApplicationCloningAPI*, jota korkeamman tason palvelu *ApplicationCloningService* käyttää analyysiohjelmien monistamiseen.

ApplicationCloningAPI
ApplicationCloningAPI on staattinen luokka, joka tarjoaa metodit ohjelmien ja niihin liittyvien asetusten yhdistämiseen ja monistamiseen. Ohjelmien yhdistäminen pitää huolen, että jokaisen yhdistettävän analyysiohjelman funktiolohkolla on yksikäsitteinen instanssinumero samaa tyyppiä olevien funktiolohkojen joukossa. Samaten monistaminen hoidetaan niin, ettei päällekkäisyyksiä synny monistettujen funktiolohkojen instanssinumeroissa.

Monistamisen apuna ApplicationCloningAPI käyttää *ApplicationCloneDefinition*- ja *SettingCloneDefinition*-nimisiä luokkia. ApplicationCloneDefinition määrittelee, montako, ja minkä nimistä kloonia yhdestä analyysiohjelmasta tehdään. SettingCloneDefinition puolestaan kertoo, miten jonkin yksittäisen asetuksen arvo muuttuu kloonatessa. Monistaminen voidaan tehdä joko XML-muodossa oleville asetuksille ja ohjelmille tai jo objektimuotoon ladatulle datalle.

ApplicationCloningService

ApplicationCloningService on OffSidessa oleva liitännäinen ja globaali palvelu, joka on konfiguroitavissa XML-tiedostolla. Se käyttää ApplicationCloningAPIa ohjelmien ja asetusten monistamiseen ja tarjoaa täten samankaltaiset operaatiot sen käyttäjille.

Palvelun konfiguroiva XML -tiedosto on tyypiltään *ApplicationCloneModelin* mukainen. Se sisältää listan kloonattavista ohjelmista, niiden nimistä sekä kaikki asetukset, jotka ovat muuttuneet klooneissa verrattuna alkuperäiseen ohjelmaan. Nämä kloonausohjeet muutetaan palvelussa ApplicationCloningAPIn ymmärtämään muotoon. Esimerkki konfiguraatiotiedostosta on listauksessa 6.1. Siinä DPHHPDOC1-ProtectionTime-nimisestä ohjelmasta tehdään kaksi kloonia, joista toisessa muutetaan yhden funktiolohkon näytteenottotaajuutta.

```
<?xml version="1.0" encoding="utf-8"?>
<CloningDefinitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <ApplicationCloneModel OriginalApplicationName="DPHHPDOC1ProtectionTime">
    <Clones>
      <ApplicationClone ClonedName="MAS.J02_DPHHPDOC1ProtectionTime">
        <ChangedSettings>
          <Setting ApplicationName="MAS.J02_DPHHPDOC1ProtectionTime"
            ParametrizedObjectTypeName="1phPreproc" FunctionBlockId="1"
            Name="SamplingFrequency" Value="1600" />
        </ChangedSettings>
      </ApplicationClone>
      <ApplicationClone ClonedName="MAS.J05_DPHHPDOC1ProtectionTime">
        <ChangedSettings>
        </ChangedSettings>
      </ApplicationClone>
    </Clones>
  </ApplicationCloneModel>
</CloningDefinitions>
```

Listaus 6.1 *Esimerkki ApplicationCloningService:n konfiguraatiosta.*

6.1.2 Tietovirtojen monistaminen

Tietovirtojen monistamiseen tehtiin *StreamServiceCloner*-niminen luokka. Luokka ottaa parametrina tietovirtapalvelun peruskonfiguraation, jota käytetään pohjana luotavalle konfiguraatiolle. Peruskonfiguraatioon voidaan yhdistää uusia konfiguraatioita, jotka käytännössä vastaavat ohjelmapohjien vaatimia konfiguraatioita. Konfiguraatioon voidaan lisätä myös uusi IED-kohtainen osio ohjelmapohjan tietovirtapalvelun konfiguraatiosta. Tällä mahdollistetaan luotujen ohjelmapohjakloonien mukaisen konfiguraation rakentaminen.

StreamServiceCloner käyttää apunaan *StreamSettingTester*-luokkaa. StreamSettingTester on singleton-suunnittelumallin mukainen hyötyluokka, joka tarjoaa metodit

tietovirta-asetusten ja COMTRADE-asetusten erotteluun analyysiohjelman vaatimista asetuksista. StreamServiceClonerin lisäksi sitä tarvitaan COMTRADE-signaalien asettelussa konfigurointityökalussa.

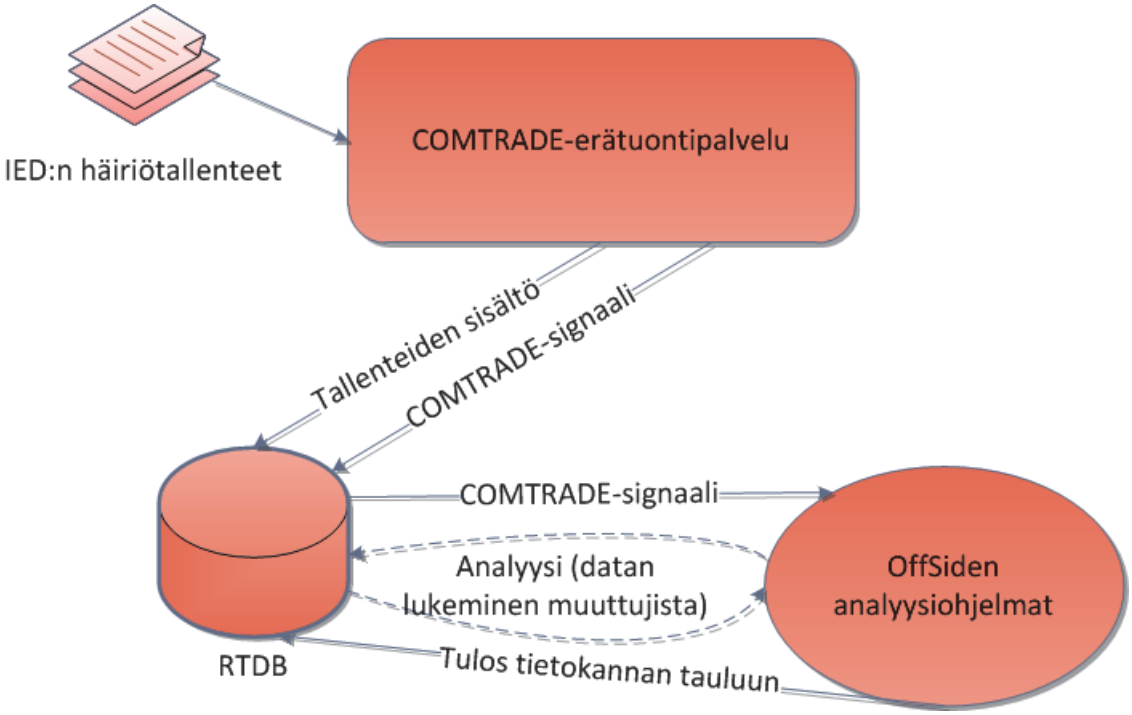
6.1.3 COMTRADE-tiedostojen tuominen erissä

OffSidessa oli valmiina palvelu COMTRADE-tiedostojen tuontiin. Olemassa oleva palvelu ei kuitenkaan ollut kaikista optimaalisin ratkaisu kaikkiin tuontitilanteisiin, sillä lisättäessä uutta sähköasemaa sille on voinut kertyä jo runsaskin määrä häiriötallenteita, joille analyysiohjelmia halutaan ajaa. Useampi tallenne on tärkeää tuoda tietokantaan aikajärjestyksessä vanhimmasta alkaen, sillä RTDB ei tue datan lisäämistä muuttujissa jo olevan uudemman datan taakse. Tätä tarkoitusta varten kehitettiin *BatchCOMTRADEImportService*.

BatchCOMTRADEImportService on XML-tiedostolla konfiguroitava OffSide-liitännäinen ja palvelu, joka tuo COMTRADE-tiedostoja RTDB-tietokantaan erissä. Tavallisen COMTRADE-tuontipalvelun tavoin se kiinnittyy tiedostojärjestelmässä oleviin konfiguraatiotiedoston mukaisiin kansioihin. Erätuontipalvelu ei kuitenkaan tarkkaile tiedostojärjestelmän muutoksista aiheutuvia tapahtumia, vaan se tarkastaa konfiguroidun aikavälin mukaisesti kaikki kansioissa olevat häiriötallenteet, joista ehyiden tallenteiden tiedot tuodaan tietokantaan. Varsinainen tuonti hoidetaan lähtökohtaisesti alkaen vanhimmasta tallenteesta. Kansiorakenteiden pitämiseksi siistinä palvelu siirtää onnistuneesti tuodut tallenteet omaan alikansioonsa ja virheelliset tallenteet omaansa.

Kuten alikohdassa 3.2.1 mainittiin, COMTRADE-häiriötallenteessa olevat kanavanimet voivat olla tyhjiä. Tästä johtuen tuontipalvelussa ei haluttu turvautua kanavanimien perusteella nimettyihin tietokantamuuttujiin, vaan haluttiin muodostaa yhtenäinen ja yksikäsitteinen tapa käsitellä COMTRADE-kanavien tietokantamuuttujia. Koska yksittäisellä kanavalla on aina tyyppi ja numero, valittiin tuontipalvelun käyttämien muuttujien nimeämiskäytännöksi yhdistelmä IED-tunnisteesta, kanavatyypistä ja kanavan numerosta. Tuontipalvelun kysyessä esimerkiksi ”JKI.J05”-lähdön häiriötallenteen toista analogikanavaa, kysyy se tietovirtapalvelulta tietovirtaa, jonka lähde on ”JKI.J05” ja nimi ”AnalogChannel2”. RTDB-tietokannassa vastaava muuttujan nimi on puolestaan ”JKI.J05.AnalogChannel2”.

Useat analyysiohjelmat asettavat itsensä käynnistymään, kun yksittäisen lähdön häiriötallenne on saatu tuotua tietokantaan. Tämä voidaan hoitaa tarkkailemalla tietokantaan tehtyä COMTRADE-taulua ja vertailemalla sinne ilmaantuvien uusien rivien IED-tunnisteita. BatchCOMTRADEImportService tarjoaa kuitenkin vaihtoehtoisen tavan analyysiohjelmien käynnistämiseen kuuntelemalla totuusarvosignaalia. Tämä on mahdollista, koska tuontipalvelu nostaa yhden tallenteen tuomisen jälkeen kyseiseen lähtöön liittyvän COMTRADE-signaalin ylös tallenteen alkuajankohtaan. Signaalin varsinainen nimi on konfiguroitavissa, mutta käytännössä yhtenäisyyden vuoksi sovittiin analyysiohjelmien kuuntelevan aina IED-tunnisteesta ja ”COMTRADE”-nimisestä signaalista pisteellä erotettua tietovirtaa. Analyysiohjelmien käynnistyminen COMTRADE-signaalin perusteella on esitettynä kuvassa 6.1.



Kuva 6.1 *Analyysiohjelmien käynnistyminen COMTRADE-signaalista.*

6.1.4 Tietovirran hakeminen COMTRADE-kanavakonfiguraatiosta

Analyysiohjelmat pyytävät aina kyseiseen ohjelmaan liittyviä signaaleja, jotka ovat nimetty löyhästi perustuen IEC61850 -standardiin. Esimerkiksi kysyttäessä Jokioisten J05-lähdön suunnatun maasulkusuojan pääasetteluja, kysyy ohjelma ”JKI.J05.DELPDEF.1”-signaalia. Koska COMTRADE-kanavamuuttujat päätettiin nimetä geneerisellä tavalla, eivät kysytyt signaalit vastaa suoraan RTDB:ssä olevia muuttujia. Ongelman ratkaisemiseksi tehtiin *COMTRADERTDBStreamProvider*-niminen luokka.

COMTRADERTDBStreamProvider tarjoaa vain yhden julkisen operaation, jolla tietovirtatunnisteen perusteella voidaan etsiä sitä vastaavan RTDB-muuttujan tunniste. Tämän mahdollistamiseksi luetaan COMTRADE-kanavakonfiguraatio, josta etsitään kysyttyä tunnistetta vastaavaa IED-tunnistetta ja signaalia. Mikäli vastaavaa ei löydy, palautetaan alkuperäinen tunniste. Tätä metodia käytetään tietovirtapalvelussa, kun siltä pyydetään uutta tietovirtaa analyysiohjelmia ladattaessa ja alustettaessa. Näin ohjelmat pääsevät käsiksi oikeisiin muuttujiin tietämättä mitään tietokannan varsinaisesta rakenteesta.

6.1.5 Muutosilmoitusten kuunteleminen historiatietokannasta

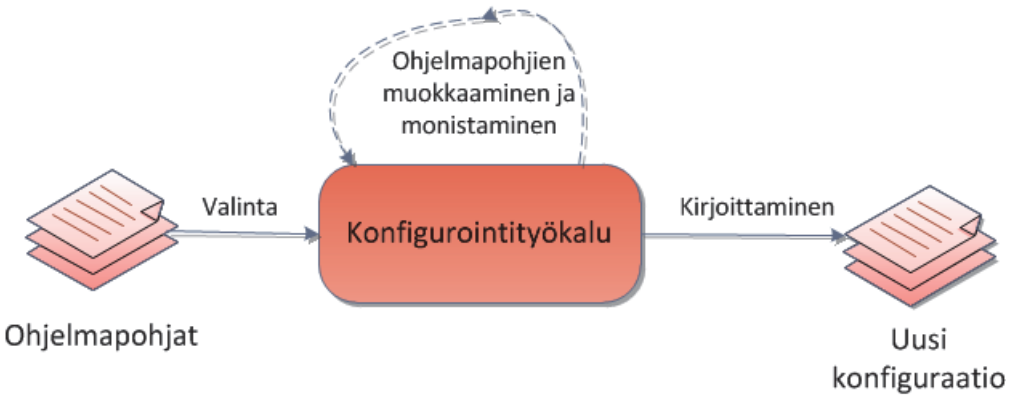
OffSide konfiguroidaan oletuksena nykytilanteeseen, eli odottamaan uutta prosessoitavaa dataa. Tämä aiheuttaa sen, että signaalien, kuten ohjelmia käynnistävien sellaisten, muutoksia kuunnellaan vasta OffSiden käynnistyshetkestä alkaen. COMTRADE-tallenteiden erätuontipalvelu kuitenkin mahdollistaa vanhojen tallenteiden tuomisen prosessoitavaksi. Koska vanhan tallenteen tuomisessa kirjoitetaan COMTRADE-signaali ylös tallenteen alkamisajankohdasta ja analyysiohjelmat kuuntelevat muutoksia alkaen OffSiden käynnistysajankohdasta, eivät signaalien muutokset koskaan välity analyysiohjelmille. Täten COMTRADE-signaalista käynnistymään asetetut ohjelmat eivät koskaan käynnisty vanhoilla tallenteilla.

Ongelman ratkaisemiseksi RTDB-tietovirrantarjoajalle lisättiin ominaisuus *EnableNotificationsFrom*, joka voidaan asettaa tietovirtapalvelun konfigurointitiedostossa. Mikäli se on määritelty konfigurointitiedostossa, huomioidaan kuunneltavien signaalien muutokset alkaen sille asetetusta päivämäärästä. Muuten muutoksia kuunnellaan kuten ennenkin, eli OffSiden käynnistymisajankohdasta alkaen.

6.2 Konfigurointityökalu

Varsinainen konfigurointityökalu on Microsoftin .NET 4.0 –sovelluskehyksellä kehitetty graafinen konfigurointivelho. Konfigurointityökalu sijaitsee PCM600-työkalussa yhtenä asematietokoneen kontekstivalikon toimenpiteenä, mutta se on siirrettävissä pienillä muutoksilla esimerkiksi omaksi sovelluksekseen. Tarkemmin sanottuna työkalu on osa OffSiden ConnPack-laajennuksen OffSideObjectType-projektia. Yksinkertaistettuna työkalun tehtävänä on tuottaa OffSideen ajettavaksi valmis paketti analyysiohjelmia.

Konfigurointisovelluksen perusteena on jo kehitettyjen lähtökohtaisten analyysiohjelmien muokkaaminen ja monistaminen. Tätä tarkoitusta varten kehitettiin *ohjelmapohjien* (ApplicationTemplate) käsite. Yleistason kuvaus konfigurointityökalun toiminnasta on esitetty kuvassa 6.2.



Kuva 6.2 *Konfigurointityökalun toiminta.*

Yksittäinen ohjelmapohja on yhtä lähtöä varten kehitetty analyysiohjelma, joka sisältää kaikki kyseisen ohjelman suorittamiseen tarvittavat XML-tiedostot. Käytännössä nämä ovat suorittamiseen tarvittavat Application.xml ja Settings.xml, sekä sisään- ja ulostulojen mallintamiset määrittelevä StreamsService.xml. Yksittäinen ohjelmapohja määritellään XML-muodossa, jossa viittaukset näihin tiedostoihin tehdään yksinkertaisesti tiedostojen nimillä. Tiedostoviittauksien lisäksi siinä määritellään ohjelmapohjan nimi, IED:n nimi, jolle ohjelmapohja on kehitetty sekä kaikki käyttäjän muutettavissa olevat asetukset. Ohjelmapohjan nimi tarvitaan, koska se voi koostua useammasta suoritettavasta analyysiohjelmasta. IED:n nimeä käytetään lähtökohtaisten muutosten mahdollistamiseksi. Käyttäjän muutettavissa olevien asetusten listaaminen puolestaan abstrahoi loppukäyttäjää kiinnostamattomat analyysiohjelman toimintalogiikkaan liittyvät yksityiskohdat pois.

Kaikki valittavissa olevat ohjelmapohjat määritellään ApplicationTemplates.xml-tiedostossa. Se sisältää jokaisen ohjelmapohjan lisäksi viittauksen tietovirtapalvelun konfigurointitiedostoon, jota käytetään muodostettavan ajopaketin tietovirtapalvelun konfigurointitiedoston pohjana. Esimerkki ohjelmapohjien konfigurointitiedoston rakenteesta löytyy listauksesta 6.2. Siinä on pilottihankkeessakin käytetyt vian paikallistamisen analyysiohjelma, jossa on kaksi käyttäjän muokattavissa olevaa asetusta, sekä suojauksen toiminta-ajan analyysiohjelma, jossa on yksi muokattava asetus.

```
<?xml version="1.0" encoding="utf-8"?>
<ApplicationTemplates xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xmlns:xsd=http://www.w3.org/2001/XMLSchema
  StreamsServicesFile="StreamsServiceBase.xml">
  <ApplicationTemplateInfos>
    <ApplicationTemplateInfo
       Name="Fault distance calculation in compensated networks"
       IEDName="JKI.J05" ApplicationFile="COMP_FLOC_Application.xml"
       SettingFile="COMP_FLOC_Settings.xml"
       StreamServiceFile="COMP_FLOC_StreamsService.xml">
      <RequiredSettings>
        <Setting ApplicationName="DFLOC"
          ParametrizedObjectTypeName="3phPreproc" FunctionBlockId="1"
          Name="GroupDelay" Value="0" />
        <Setting ApplicationName="DFLOC"
          ParametrizedObjectTypeName="3phPreproc" FunctionBlockId="1"
          Name="SamplingFrequency" Value="1000" />
      </RequiredSettings>
    </ApplicationTemplateInfo>
    <ApplicationTemplateInfo Name="Fault clearance time analysis"
       IEDName="JKI.J05" ApplicationFile="PROTECTION_TIME_Application.xml"
       SettingFile="PROTECTION_TIME_Settings.xml"
       StreamServiceFile="PROTECTION_TIME_StreamsService.xml">
      <RequiredSettings>
        <Setting ApplicationName="MainApp"
```

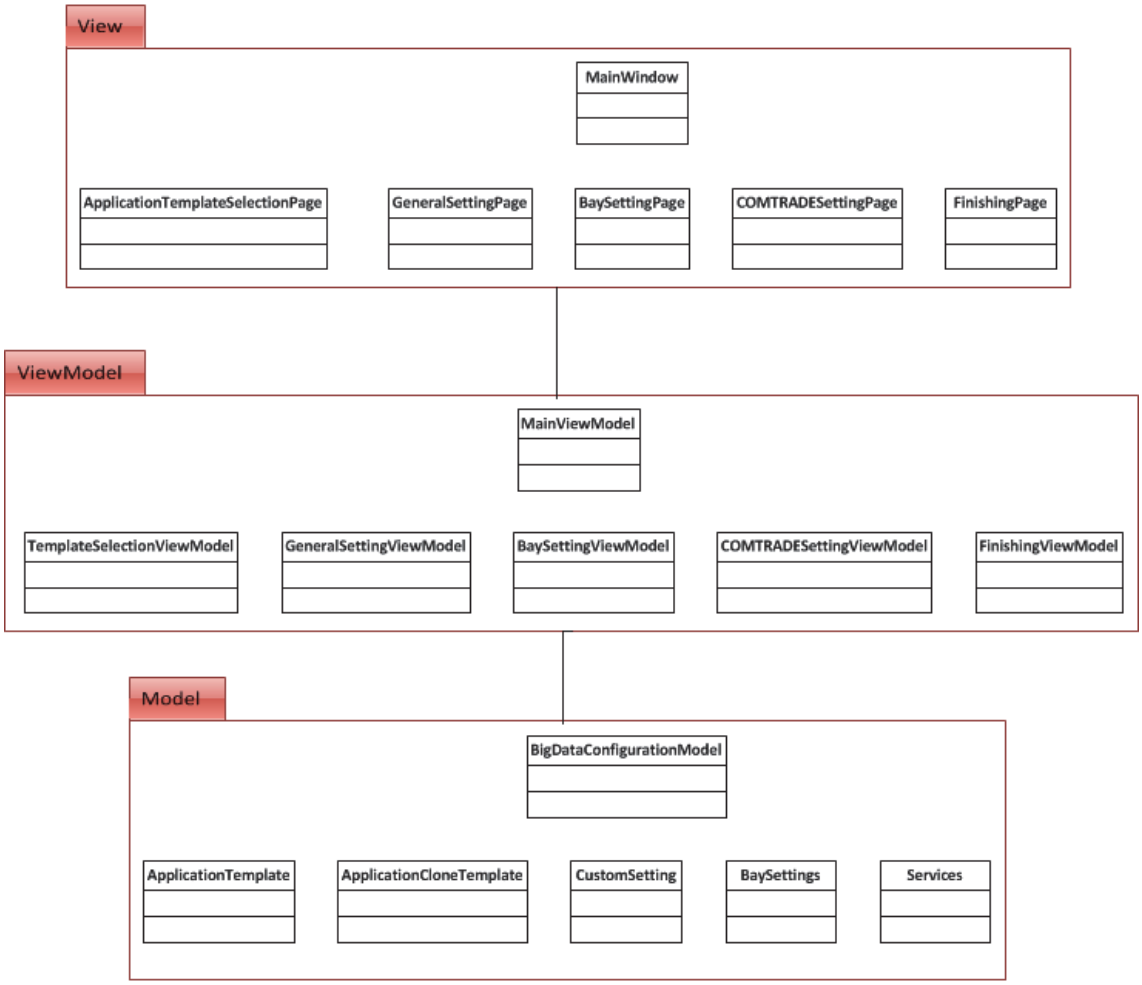
```
            ParametrizedObjectTypeName="ProtectionOperationTimeCalculation"
            FunctionBlockId="1"    Name="warningTolerance" Value="20" />
        </RequiredSettings>
      </ApplicationTemplateInfo>
    </ApplicationTemplateInfos>
</ApplicationTemplates>
```

Listaus 6.2 *Esimerkki ohjelmapohjien konfiguraatiosta.*

6.2.1 Konfigurointityökalun yleisarkkitehtuuri

Konfigurointityökalun graafinen osio on tehty käyttäen WPF-grafiikkakirjastoa (Windows Presentation Foundation), joka on Windows Vistan ja sitä uudempien Windows-versioiden graafinen rajapinta .NET-sovelluskehyksessä. (Microsoft, 2014a). Tästä johtuen työkalun yleisarkkitehtuurissa on päädytty noudattamaan MVVM-mallia, joka on suositus WPF:ää hyödyntävissä sovelluksissa (Smith, 2009). MVVM-malli koostuu näkymästä (View), näkymämallista (View Model) ja mallista (Model). Näkymä sisältää kaikki GUI:ssa, eli graafisessa käyttöliittymässä, näkyvät elementit ja malli sisältää käsiteltävän datan joko objektimallina tai datan käsittelykerroksena. Näkymämalli toimii abstraktiona näkymän ja mallin välillä. Sitä käytetään deklaratiivisen datasidonnan kohteena, eli näkymässä näytettävien tietojen sitomiseen.

Korkean tason arkkitehtuurikuvaus konfigurointityökalun luokista on esitetty kuvassa 6.3.



Kuva 6.3 *Toteutetun konfigurointityökalun korkean tason arkkitehtuuri.*

Työkalun päänäkymänä toimii navigointi-ikkuna, joka sisältää kaikki konfigurointivelhon sivut. Yksittäisiä sivuja ovat ohjelmapohjien valintasivu, yleisten asettelujen sivu, lähtökohtaisten asettelujen sivu, COMTRADE-asetussivu sekä konfiguraation viimeistelysivu. Sekä navigointi-ikkunalla että jokaisella sivulla on oma näkymämallinsa, jotta näkymät saadaan riippumattomiksi malleista. Navigointi-ikkunan näkymämalli toimii samalla päänäkymämallina, joka omistaa kaikki muut näkymämallit näkymien tapaan.

Päämallina konfigurointityökalussa toimii *BigDataConfigurationModel*. Sen vastuulla on ohjelmapohjien lataaminen xml-tiedostoista ja niiden eheyden tarkistaminen, sähköaseman nimen ja lähtöjen tunnisteiden kerääminen PCM600:sta, tietojen säilöminen näkymien välillä sekä kirjoitettavan konfiguraation ylläpitäminen. Päämallin lisäksi työkalussa on muutamia muita malleja, kuten CustomSetting ja ApplicationCloneTemplate, joita käytetään joko BigDataConfigurationModelissa tai näkymämalleissa.

6.2.2 Työnkulku konfigurointisovelluksessa

Työnkulku konfigurointisovelluksessa tapahtuu sivu kerrallaan. Näytettävät sivujen järjestys ja täten myös työnkulku on ohjelmapohjien valitseminen, yleiset asettelut, lähtökohtaiset asettelut, COMTRADE-asetukset sekä konfiguraation viimeistely.

Ohjelmapohjien valitseminen
Käyttäjän käynnistäessä konfigurointisovelluksen ladataan kaikki ohjelmapohjat muistiin ApplicationTemplates.xml-tiedoston perusteella. Mikäli yhdessäkin ohjelmapohjan määrittelemässä tiedostossa on virhe, virheen sisältävää ohjelmapohjaa ei ladata. Samalla ladataan myös sähköasemien ja sähköasemalla olevien lähtöjen tunnisteet, joista muodostetaan pisteellä erotettuina lähtökohtaiset tunnisteet. Tunnisteista ja valittavissa olevista ohjelmapohjista muodostetaan valintamatriisi, jossa oletuksena on valittuna jokainen ladattu ohjelmapohja jokaiselle lähdölle. Käyttäjän tulee valita vähintään yksi ohjelmapohja jollekin lähdölle, jotta työkalussa voidaan liikkua eteenpäin.

Yleiset asettelut
Siirryttäessä yleisten asetteluiden sivulle yhdistetään jokainen valittu ohjelmapohja yhdeksi paketiksi. Tämä tehdään, jotta päällekkäisiä instanssinumeroita ja samalla suorituskelvotonta konfiguraatiota ei kirjoitettaisi ajoympäristöön.

Yleisten asetteluiden sivulla jokaisesta valitun ohjelmapohjan käyttäjän määriteltävissä olevista asetuksista etsitään samankaltaiset, jotka esitetään yhteisinä asetuksina. Jäljelle jäävät asetukset esitetään ohjelmakohtaisesti. Ideana on, että tässä vaiheessa on helppo muokata ohjelmapohjan niitä asetuksia, jotka ovat samoja suurelle osalle lähdöistä.

Lähtökohtaiset asettelut
Ennen siirtymistä lähtökohtaisiin asetteluihin työkalu luo lähtötunnisteiden mukaisesti kloonit valituista ohjelmapohjista. Lähtökohtaisten asettelujen sivuilla näytetään näistä klooneista käyttäjän määriteltävissä olevat asetukset lähtöjen mukaan. Tässä vaiheessa voi siis esimerkiksi muuttaa pelkästään yhden lähdön näytteenottotaajuutta, mikäli kyseisellä lähdöllä on käytössä eri näytteenottotaajuudelle aseteltu suojarele.

COMTRADE-asetukset
Kun liikutaan COMTRADE-asetuksiin, käy konfigurointityökalu lukemassa ohjelmapohjien vaatimat sisään- ja ulostulot. Nämä kaikki oletetaan COMTRADE-signaaleiksi, jotka jaotellaan analogisiin ja binäärisiin muuttujien tyypin perusteella. Valittujen ohjelmapohjien perusteella käyttäjälle tuodaan määriteltäväksi jokaisen lähdön COMTRADE-signaalit, eli käyttäjän tulee valita mistä COMTRADE-tallenteen analogi- tai binäärikanavasta kukin signaali löytyy. Valintojen helpottamiseksi tarjolla on mahdollisuus ladata esimerkki kyseisen lähdön COMTRADE-tallenteesta, jonka avulla valinnat pyritään tekemään automaattisesti tallenteesta luettujen kanavanimien perusteella.

Lisäksi tässä näkymässä on tarjolla testikäyttöön mahdollisuus luoda ja alustaa lähtökohtaiset COMTRADE-muuttujat RTDB-tietokantaan.

Konfiguraation viimeistely

Konfiguraation viimeistelysivulla on vielä mahdollisuus asetella COMTRADE-tiedostojen tuontiin liittyvät asetukset, eli tiedostopolku, johon tallenteet tulevat sekä tarkastusväli, jolla uusien tallenteiden ilmaantuminen tarkastetaan. Näiden asettelujen jälkeen tarjolla on mahdollisuus konfiguraation viimeistelyyn ja samalla analyysiohjelmien uudelleenkäynnistämiseen.

Kun uusi konfiguraatio viimeistellään, konfigurointityökalu ottaa yhteyden OffSiden etärajapintaan. Yhteyden muodostettua tuotetaan OffSiden datakansioon uusi instanssinumerolla varustettu kansio, jonka instanssinumero on yhdellä suurempi kuin siellä tuotoshetkellä oleva suurin instanssinumero. Uusi kansio sisältää monistetut analyysiohjelmat ja niihin liittyvät asetukset, monistetun tietovirtapalvelun konfiguraation, käyttäjän asetteleman COMTRADE-kanavakonfiguraation sekä COMTRADE-tuontipalvelun konfiguraation, eli seuraavat viisi XML-tiedostoa:

- Application.xml
- Settings.xml
- StreamsService.xml
- COMTRADEChannelConfiguration.xml
- BatchCOMTRADEImportServiceConfiguration.xml.

Uuden konfiguraation ilmestyminen OffSideen aiheuttaa kyseisen konfiguraation lataamisen ja täten uusien analyysiohjelmien käynnistymisen, mikäli OffSide on suorituksessa kirjoitushetkellä. Muuten uusi konfiguraatio ladataan käynnistettäessä OffSide seuraavan kerran.

6.3 Toteutuksen arviointi

Syntynyttä toteutusta arvioitiin testaamalla sitä käytännössä sekä arvioimalla sen rakennetta.

6.3.1 Toteutuksen testaaminen

Konfigurointityökalu

Konfigurointityökalua testattiin konfiguroimalla analyysiohjelmat ajoon uudelle sähköasemalle. Sähköasemaksi valittiin Masalan sähköasema, koska ABB:llä oli valmiina sieltä useamman vuoden aikana kerätyt häiriötallenteet. Masalan asemalta analysoitavaksi valittiin seitsemän eri lähtöä, joista häiriötallenteita kertyi hivenen yli 2300.

Ennen varsinaista konfigurointia Masalan tallenteiden kanavarakennetta verrattiin Jokioisten tallenteiden kanavarakenteisiin. Vertailussa havaittiin, että Masalan tallenteissa on enemmän käynnistys- ja operointisignaalipareja binäärikanavissa. Lisäksi Masalan binäärikanavissa useammalla käynnistyssignaalilla on yhteinen operointisignaali.

Näistä syistä Elenialle kehitetty suojausten toiminta-ajan analyysiohjelma ei suoraan soveltunut suoritettavaksi Masalan häiriötallenteille, joten niille valittiin suoritettavaksi aluksi vain vian paikallistamisen analyysi.

Konfigurointi aloitettiin luomalla PCM600-työkaluun uusi projekti. Uuteen projektiin luotiin Masalan analysoitavia lähtöjä vastaava laitehierarkia, eli asematietokone sekä suojareleet jokaiselle lähdölle. Tämän jälkeen asematietokoneelle käynnistettiin konfigurointityökalu, joka kirjoitti paikallisen koneelle konfiguraation tuloksen. Valmis konfiguraatio siirrettiin Elenian pilottihankkeen palvelimelle yhdessä Masalan häiriötallenteiden kanssa. Lopulta palvelimella oleva OffSide käynnistettiin uudella konfiguraatiolla, joka toi jokaisen lähdön häiriötallenteet aikajärjestyksessä tietokantaan. Uuden tallenteen ilmestyminen tietokantaan käynnisti aina lähtöä vastaavan vian paikallistamisen analysointiohjelman. Itse konfigurointi osoittautui toimivaksi, sillä kaikki konfiguroidut analyysiohjelmat saatiin onnistuneesti ladattua ajoympäristöön ja suoritukseen.

Ajoympäristön suorituskyky

Kuten jo mainittua, Jokioisten sähköaseman tallenteille kehitetty suojausten toiminta-ajan analyysi ei suoraan soveltunut Masalan sähköaseman tallenteille. Jotta siitä saataisiin paremmin muokkautuva ohjelma, päätettiin se pilkkoa pienempiin osiin. Yksittäisen kolmetoista käynnistys- ja operointisignaaliparia sisäänsä ottavan ohjelman sijasta kokeiltiin lähestymistapaa, jossa yksittäinen analyysiohjelma vastaa vain yhden käynnistys- ja operointiparin tarkkailemisesta. Kun tällä tavalla kehitetään ohjelmapohjia konfigurointityökalulle, voidaan jokaisen lähdön perusteella valita sille sopivat analyysiohjelmat. Uuteen versioon suojauksen toiminta-ajan analyysista tehtiin lisäksi virta- ja jännitekanavien huippuarvojen tarkkailu.

Masalan häiriötallenteille sopivia toiminta-ajan analyysiohjelmapohjia tuli lopulta 22 kappaletta. Kun nämä kaikki valittiin konfiguroinnissa seitsemälle lähdölle, saatiin suoritukseen yhteensä 154 erillistä analysointiohjelmaa. Koska kaikki analysoitavat tallenteet oli jo tuotu tietokantaan, haluttiin uudet analyysiohjelmat ajaa suoraan tietokannassa oleville tallenteille. Tämän mahdollistamiseksi käytettiin kohdassa 6.1.5 kuvattua RTDB-tietovirtaan lisättyä ominaisuutta. Sen avulla uudet versiot toiminta-ajan analysointiohjelmista käynnistyivät automaattisesti oikeista ajanhetkistä lähtökohtaisten COMTRADE-signaalien perusteella.

Uusien analyysiohjelmien ollessa suorituksessa havaittiin, että suuren ohjelmamäärän suorittaminen samanaikaisesti oli yllättävän hidasta. Analyysiohjelmien annettiin olla ajossa yhtäjaksoisesti lähes neljä vuorokautta, jonka aikana jokainen 154 analysointiohjelmasta sai käytyä läpi karkeasti arvioiden noin 20 COMTRADE-tallennetta. Pullonkaulaksi epäiltiin OffSiden RTDB-rajapintaa, jota kaikki analyysiohjelmat käyttävät samanaikaisesti. Jatkokehitysideana RTDB-rajapinta voisi hyödyntää VtrinLib:in tarjoamia nopeita iteraattoreita, joista on tosin dokumentaatiota varsin rajallisesti saatavilla. Toinen vaihtoehto on, että ohjelmat lukevat tietokannasta dataa muistiin isompina paloina, mikä vähentää tietokantaan kohdistuvien operaatioiden lukumäärää.

6.3.2 Rakenne

Konfigurointityökalu

Opinnäytetyössä toteutettu konfigurointityökalu ei lopulta hyödyntänyt juurikaan PCM600:n tarjoamia rajapintoja. Ainoa PCM:stä suoraan riippuva asia on PCM600-työkalun hyödyntäminen OffSiden konfiguraation kirjoittamisessa. Koska PCM600-konfigurointityökalu on tarkoitettu yksittäisen sähköaseman konfigurointiin, voidaan toteutetulla konfigurointityökalullakin tällä hetkellä konfiguroida analyysiohjelmat vain yhden sähköaseman lähdöille. Työkalu tukee kuitenkin usean sähköaseman samanaikaista konfigurointia, joten työkalu voidaan halutessaan eriyttää PCM:stä laajemman konfiguroinnin mahdollistamiseksi. Jatkossa useiden sähköasemien konfiguroiminen kerralla ja täten myös työkalun eriyttäminen onkin järkevää.

Konfigurointityökalun käynnistymisessä havaittiin muutaman sekunnin viive. Viive kasvoi, kun ohjelmapohjien valinnassa siirryttiin käyttämään matriisimaista toteutusta. Microsoftin WPF-kirjaston .NET 4.0-versiossa ei ole suoraan matriiseja tukevaa grafiikkaelementtiä, joten työkalussa päädyttiin käyttämään *BindableDictionary*-luokkaa. Sen käyttämien dynaamisten jäsenmuuttujien ajonaikainen sitominen grafiikkaelementteihin on todennäköisesti syynä kasvaneeseen viiveeseen. Konfigurointityökalun katsottiin kuitenkin olevan riittävän nopea tämän työn puitteissa tehtävään konfigurointiin.

Konfigurointityökalun käyttäminen mahdollistaa valmiiden ohjelmien hyödyntämisen eri lähdöissä. Vaikka työkalu vähentääkin merkittävästi manuaalista konfigurointia, edellyttää ohjelmapohjien luominen edelleen manuaalista XML-tiedoston muokkaamista. Ohjelmapohjien manuaalinen muokkaaminen tehdään kuitenkin huomattavasti kootummin johtuen niiden kokoamisesta yhtenäiseen konfigurointitiedostoon. Tulevaisuudessa voitaisiin kuitenkin kehittää kenties oma työkalu ohjelmapohjien luomiseen ja hallitsemiseen. Vaihtoehtoisesti konfigurointityökalu voitaisiin laajentaa tukemaan suoraan PCM600:ssa olevia ACT-ohjelmakonfiguraatioita.

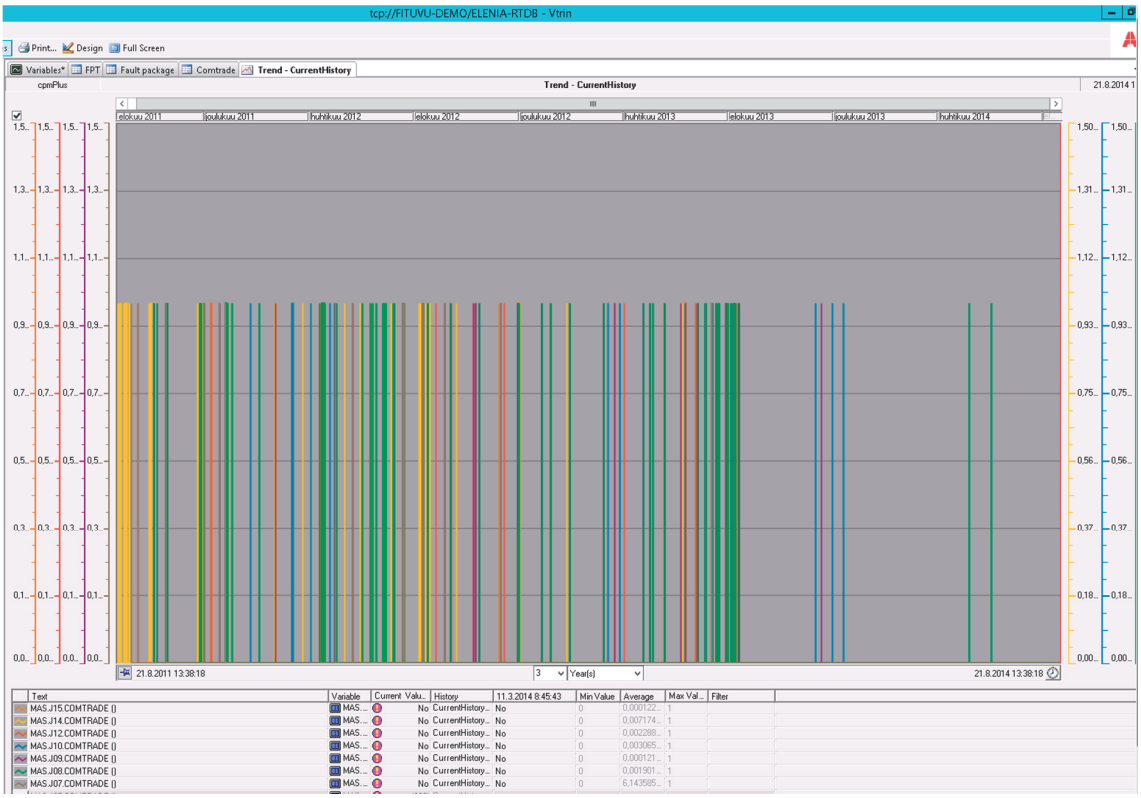
Yleiset komponentit

Häiriötallenteiden erätuontipalvelu mahdollistaa hyvin vanhojen häiriötallenteiden tuomisen prosessoitavaksi. Erätuontipalvelu käyttää hyödykseen osaa valmiina olleen tuontipalvelun metodeista, mutta sen toteutus sisältää myös osittain päällekkäisiä metoditoteutuksia. Tuontipalvelut voitaisiinkin yhdistää esimerkiksi yhdeksi palveluksi, jolla on kaksi eri toimintamoodia. Toisaalta kaksi erillistä palvelua voidaan myös säilyttää, mutta ylläpidettävyyden helpottamiseksi yhteinen toiminnallisuus kannattaisi eriyttää omaksi kokonaisuudekseen.

COMTRADE-kanavien indekseihin perustuva tietokantamuuttujien nimeäminen harmonisoi tietokannan rakennetta. Samalla päästään eroon monimuotoisten kanavanimien tarkkailusta tallenteiden tuontivaiheessa. Vaikka nimeämiskäytäntö lisää yhden pakollisen konfiguraatiokerroksen tuottamisen ja lukemisen, on se myös panostus tulevaisuuteen. Mikäli uudet analyysiohjelmat vaativat COMTRADE-kanavia, jotka eivät ole mallinnettuina kanavakonfiguraatiossa, tarvitsee ohjelmien ajamiseksi päivittää

vain kanavakonfiguraatio. Näin tietokannassa on aina tuotuna häiriötallenteiden kaikki tiedot, eikä sen rakennetta tarvitse muuttaa muuttujien luomisen jälkeen.

Analyysiohjelmien käynnistäminen lähtökohtaisten COMTRADE-signaalien perusteella osoittautui toimivaksi ratkaisuksi. Erityisesti mahdollisuus lukea COMTRADE-signaalin muutoksia historiasta antaa helpon keinon ohjelmien suorittamiselle jokaiselle tietokannassa olevalle häiriötallenteelle. Tästä on etua varsinkin, kun häiriötallenteita on kerätty usean vuoden ajalta, kuten Masalan sähköasemalla on tehty. Esimerkki Masalan tallenteiden COMTRADE-signaaleista RTDB-tietokannassa löytyy kuvasta 6.4.



Kuva 6.4 *COMTRADE-signaalit tietokannassa.*

Analyysiohjelmien käynnistäminen lähtökohtaisten COMTRADE-signaalien perusteella tukee myös uusien ohjelmien kehittämistä, sillä uudet analyysiohjelmat voidaan ajaa kaikille tuoduille tallenteille vain yhdellä muutoksella tietovirtapalvelun konfiguraatiossa.

7 JOHTOPÄÄTÖKSET

Tämä opinnäytetyö keskittyi ABB:n kehittämään OffSide-ohjelmistoalustaan, jolla voidaan suorittaa sähköasemalta saatavalle datalle kehitettyjä analyysiohjelmia. Analyysiohjelmien konfigurointia varten kerättiin vaatimukset erilliselle konfigurointityökalulle Elenia Oy:n kanssa aloitetussa pilottihankkeessa, jossa aluksi keskityttiin ajamaan analyysiohjelmia Jokioisten sähköasemalta saaduille häiriötallenteille. Vaatimuksien perusteella työssä kehitettiin konfigurointityökalu, joka liitettiin osaksi PCM600-työkalun ConnPack-laajennusta. OffSiden ohjelmistoa muokattiin myös tukemaan analyysiohjelmien konfigurointia paremmin. Toteutettua työkalua testattiin konfiguroimalla pilottihankkeen analyysiohjelmat suoritukseen toisen sähköaseman häiriötallenteille.

Koska PCM600-työkalu on tarkoitettu yhden sähköaseman konfiguroimiseen kerralla, soveltuu toteutettu konfigurointityökalukin nykyisellä sijainnillaan analyysiohjelmien konfiguroimiseen kerralla vain yhden sähköaseman lähdöille. Uuden konfiguraation kirjoittaminen ajoympäristöön aiheuttaa sen lataamisen ajoon, sekä mahdollisen vanhan konfiguraation keskeyttämisen. Näistä syistä yhdessä ajoympäristössä voidaan kerrallaan suorittaa vain yhden sähköaseman analyysiohjelmia. Käytännössä jokainen sähköasema, jolle halutaan ajaa analyysiohjelmia nykyhetkessä, vaatiikin oman OffSide-ohjelmistoalustan sisältävän sähköasematietokoneen. Tällöin sähköasemakohtaiset tulokset voidaan replikoida yhteiseen tietokantaan, jota käytetään varsinaisen pilvipalvelun pohjana.

Sähköasemakohtaiset OffSide-sovellukset vaikuttavat olevan paras ratkaisu myös suorituskyvyn skaalautuvuuden kannalta. Samaan aikaan suoritettavien analyysiohjelmien määrän kasvaessa yksittäisen ohjelman suoritusaika hidastuu merkittävästi. Pullonkaulana vaikuttaisi olevan OffSiden abstraktiokerros RTDB-tietokantaan, johon kohdistuu suuri määrä rinnakkaisia lukuoperaatioita. Jatkokehitysideana voitaisiin tutkia, nopeutuuko ohjelmien suorittaminen käyttämällä tietojen lukemiseen VtrinLibin tarjoamia nopeita iteraattoreita. Toinen vaihtoehto on, että ohjelmat lukevat tietokannasta dataa muistiin isompina paloina, mikä vähentää tietokantaan kohdistuvien operaatioiden lukumäärää.

Analyysiohjelmien kehittäminen edellyttää yhtenäisiä suunnittelukäytäntöjä. Yksittäinen analyysiohjelma kannattaa suunnitella yhtä sähköaseman lähtöä varten, jotta analyysiohjelman monistaminen toisille lähdöille onnistuisi. Monistamista tukee myös ohjelmien pitäminen mahdollisimman yksinkertaisina. Tästä saatiin hyvä esimerkki, kun 13 häiriötallenteen signaaliparia ottava ohjelma kannatti pilkkoa 13 erilliseksi ohjelmaksi. Lisäksi erityisesti ohjelmien vaatimien signaalien nimien tulee olla yhtenäisiä, jotta järjestelmän toiminta saadaan mahdollisimman automatisoiduksi ja täten tehok-

kaaksi. IEC 61850 -standardiin perustuva nimeämiskäytäntö onkin hyvä ottaa yleiseksi tavaksi ohjelmia suunniteltaessa.

Toteutetun konfigurointityökalun käyttämien analyysiohjelmapohjien avulla saadaan hyödynnettyä jo toteutettuja analyysiohjelmia. Ohjelmapohjien luominen ja ylläpitäminen edellyttää kuitenkin jonkin verran manuaalista XML-tiedoston editointia. Jatkokehitysideana onkin pohjien luominen suoraan PCM600-työkalun ACT-moduulista.

Häiriötallenteita datalähteinä käyttävien analyysiohjelmien käynnistäminen tietokannan lähtökohtaisesta COMTRADE-muuttujasta osoittautui hyväksi ratkaisuksi. Sen avulla uudet analyysiohjelmat voidaan suorittaa automaattisesti jokaiselle tietokantaan tuodulle tallenteelle. Nykyinen rakenne ei kuitenkaan tue uusien ohjelmien yhdistämistä suorituksessa olevien analyysiohjelmien kanssa. Tätä tarkoitusta varten OffSideen voitaisiin lisätä mahdollisuus tuoda uusia analyysiohjelmia suoritukseen ilman muutoksia ajossa oleviin ohjelmiin.

Toteutuksessa päädyttiin harmonisoimaan tietokanta nimeämällä häiriötallenteiden kanavien muuttujat kanavan tyypin ja indeksin mukaan. Tämä loi tarpeen COMTRADE-kanavakonfiguraation käyttämiselle. Uutta asemaa lisätessä kanavakonfiguraatiota ei nykytilanteessa oteta automaattisesti huomioon RTDB:n laitemallissa, jonka avulla tietojen visualisointi web-puolella toteutetaan. Jatkokehitysideana laitemalli kannattaa päivittää automaattisesti, kun uusi kanavakonfiguraatio luodaan.

Perusinfrastruktuuri analyysiohjelmien suorittamiselle sähköasemadatalle on täysin toimiva. Tämän osoitti pilottihankkeessa saadut relevantit tulokset häiriötallenteilla. Järjestelmä vaatii kuitenkin jatkokehitystä ja lisää testausta prosessien parantamiseksi. Lisäksi IEC 61850:n mukaisia datapisteitä voitaisiin lisätä tietokantaan, jotta uudet, monipuolisemmat, analyysiohjelmat voisivat hyödyntää niitä.

LÄHTEET

ABB Ltd. 2010. Architecture Specification for PCM600. Sisäinen dokumentti.

ABB Ltd. 2012. Technical Report – OffSide Architecture and Design. Sisäinen dokumentti.

ABB Ltd. 2013a. cpmPlus Platform Architecture Description. Sisäinen dokumentti.

ABB Ltd. 2013b. cpmPlus Data Abstraction Layer. Sisäinen dokumentti.

ABB Ltd. 2013c. PCM600 Product Guide.

ABB Ltd. 2013d. Big Data Mining. Sisäinen dokumentti.

ABB Ltd. 2013e. Technical Note – OffSide Developer manual. Sisäinen dokumentti.

Cleen. 2014. Smart Grids and Energy Markets (SGEM). [WWW]. [Viitattu 15.07.2014]. Saatavissa: http://www.cleen.fi/fi/sgem.

Cigre Working Group B5.20. 2010. New Trends for Automated Fault and Disturbance Analysis. Cigre report.

Elenia. 2014. Tietoa Elenia-konsernista. [WWW]. [Viitattu 15.07.2014]. Saatavissa: http://www.elenia.fi/yritys/elenia_info.

Energiavirasto, 2012. Sähköverkkotoiminnan tunnusluvut 2012. [WWW]. [Viitattu 28.10.2014]. Saatavissa: http://www.energiavirasto.fi/sahkoverkkotoiminnan-tunnusluvut-2012.

IEC 61850-1. 2013. Communication networks and systems for power utility automation – Part 1: Introduction and overview. Edition 2.0. International Electrotechnical Comission. 33 s.

IEC 61850-5. 2013. Communication networks and systems for power utility automation – Part 5: Communication requirements for functions and device models. Edition 2.0. International Electrotechnical Comission. 143 s.

IEC 61850-6. 2009. Communication networks and systems for power utility automation – Part 6: Configuration description language for communication in electrical substations related to IEDs. Edition 2.0. International Electrotechnical Comission. 215 s.

IEC 61850-7-1. 2011. Communication networks and systems for power utility automation – Part 7-1: Basic communication structure – Principles and models. Edition 2.0. International Electrotechnical Comission. 132 s.

IEC 61850-7-2. 2010. Communication networks and systems for power utility automation – Part 7-2: Basic information and communication structure – Abstract communication service interface (ACSI). Edition 2.0. International Electrotechnical Comission. 213 s.

IEC 61850-7-4. 2010. Communication networks and systems for power utility automation – Part 7-4: Basic communication structure – Compatible logical node classes and data object classes. Edition 2.0. International Electrotechnical Comission. 179 s.

IEC 61850-8-1. 2011. Communication networks and systems for power utility automation – Part 8-1: Specific communication service mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3. Edition 2.0. International Electrotechnical Comission. 189 s.

IEEE Std C37.111-1999. 1999. IEEE Standard Common Format for Transient Data Exchange (COMTRADE) for Power Systems. Institute of Electrical and Electronics Engineers. 55 s.

Ilmatieteenlaitos. 2014. Ilmatieteen laitoksen avoin data. [WWW]. [Viitattu 15.09.2014]. Saatavissa: https://ilmatieteenlaitos.fi/avoin-data.

Kezunovic, Mladen; Matic Cuka, Biljana; Popovic, Tomo. 2009. Substation Data Integration and Utilization. Power Systems Conference and Exposition, 2009.

Martin, Robert C. 2002. Agile Software Development: Principles, Patterns, and Practices. Prentice Hall, 2002.

Microsoft. 2014a. Introduction to WPF. [WWW]. [Viitattu 16.10.2014]. Saatavissa: http://msdn.microsoft.com/en-us/library/aa970268%28v=vs.110%29.aspx

Microsoft. 2014b. Regular Expression Language - Quick Reference. [WWW]. [Viitattu 16.10.2014]. Saatavissa: http://msdn.microsoft.com/en-us/library/az24scfc%28v=vs.110%29.aspx.

OPC Foundation. 2014. What is OPC?. [WWW]. [Viitattu 16.10.2014]. Saatavissa https://opcfoundation.org/about/what-is-opc/.

Puurtinen, Joonas. 2014. Big Data mining as Part of Substation Automation and Network Management. Tampere: Tampere University of Technology, 2014.

Ryan, Bob and Shank, Charles. 1999. IEEE Standard Common Format for Transient Data Exchange (COMTRADE) for Power Systems. New York: The Institute of Electrical and Electronics Engineers, Inc., 1999.

Smith, Josh. 2009. WPF Apps With The Model-View-ViewModel Design Pattern. [WWW]. [Viitattu 09.07.2014]. Saatavissa: http://msdn.microsoft.com/en-us/magazine/dd419663.aspx.

Steele, Julie. 2012. Why data visualization matters. [WWW]. [Viitattu 15.09.2014]. Saatavissa http://radar.oreilly.com/2012/02/why-data-visualization-matters.html.

Strauss, Cobus. 2003. Practical Electrical Network Automation and Communication Systems. Newnes, 2003.

Tweed, Katherine. 2012. Smart Grid Has Its Head in the Cloud. [WWW]. [Viitattu 10.09.2014]. Saatavissa http://www.greentechmedia.com/articles/read/smart-grid-has-its-head-in-the-cloud.

Valtari, Jani ja Verho, Pekka. 2011. Requirements and Proposed Solutions for Future Smart Distribution Substations. Journal of Energy and Power Engineering 5 (2011).

Yle. 2013. Elenia. Elenia maksaa miljoonakorvaukset sähkökatkoista. [WWW]. [Viitattu 28.10.2014]. Saatavissa http://yle.fi/uutiset/elenia_maksaa_miljoonakorvaukset_sahkokatkoista/6944301.