TAMPEREEN TEKNILLINEN YLIOPISTO

ERKKA KETTUNEN

SÄHKÖASEMATIETOKONEEN KONFIGUROINTI

Diplomityö

Tarkastaja: Professori Hannu-Matti Järvinen
Tarkastaja ja aihe hyväksytty Tieto- ja sähkötekniikan tiedekunnan kokouksessa 6. lokakuuta 2010.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Sähkötekniikan koulutusohjelma
KETTUNEN, ERKKA: Sähköasematietokoneen konfigurointi
Diplomityö, 47 sivua, 1 liitesivu
Toukokuu 2011
Pääaine: Sulautetut järjestelmät
Tarkastaja: Professori Hannu-Matti Järvinen
Avainsanat: keskitetty suojaus- ja ohjaustekniikka, sähköverkon suojalaite, asematietokone

Sähkönkulutuksen ja hajautetun sähköntuotannon lisääntyminen vaatii jakeluverkon ja verkostoautomaation päivittämistä. Automaatiojärjestelmien päivittäminen on aiheuttanut pitkiä käyttökatkoja ja lisäkuluja jakeluyhtiöille. Tämä opinnäytetyö liittyy ABB:n kehittämään sähköaseman keskitettyyn suojaus- ja ohjaustekniikkaan, joka yksinkertaistaa ja helpottaa automaatiojärjestelmien ylläpitoa ja päivittämistä. Tekniikka lisää sähköaseman toiminnallisuutta hyödyntämällä IEC 61850 -standardia ja sähköasematietokonetta.

Opinnäytetyön tavoitteena oli kehittää asematietokoneelle konfigurointityökalu ja laajentaa asematietokoneen ohjelmisto konfiguroitavaksi. Kirjallisuusosassa esitellään keskitetty suojaus- ja ohjaustekniikka, määritellään vaatimukset asematietokoneen konfiguroinnille ja esitellään IEC 61850 -standardi. Tutkimusosassa esitellään asematietokoneen ohjelmisto ja konfigurointiin käytettävä PCM600-työkalu.

Tutkimustuloksena kuvataan PCM600-työkaluun kehitetyn Connectivity Package -laajennuksen toteutus ja tiedonsiirto konfigurointityökalun ja asematietokoneen välillä. Tuloksissa esitellään myös asematietokoneen ohjelmistoon tehdyt muutokset, jotka mahdollistivat sen konfiguroinnin. Konfigurointi noudattaa IEC 61850 -standardin määrittelemää prosessia ja asematietokoneen konfiguraatio kuvataan kokonaisuudessaan standardoidulla SCL-kielellä.

Työn lopputuloksena saatiin toimiva konfigurointiympäristö asematietokoneelle. Toteutuksen yhteydessä asematietokoneen ohjelmistossa havaittiin rajoituksia, jotka tulee ottaa huomioon järjestelmän jatkokehityksessä. Konfigurointityökalun toteutus on yhteensopiva myös tulevaisuuden asematietokoneiden kanssa, sillä konfigurointiprosessi noudattaa IEC 61850 -standardia. Keskitetty suojaus- ja ohjaustekniikka todettiin käyttökelpoiseksi, ja sitä voidaan suositella käytettäväksi tulevaisuuden sähkönjakeluautomaatiojärjestelmissä.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Electrical Engineering
KETTUNEN, ERKKA: Configuration of Electrical Substation Computer
Master of Science Thesis, 47 pages, 1 appendix page
May 2011
Major: Embedded systems
Examiner: Professor Hannu-Matti Järvinen
Keywords: centralized protection and control, IED, electrical substation computer

An increase in power consumption and distributed generation requires updates to both electrical grid and distribution automation systems. Updating the automation systems has required long maintenance breaks in the distribution causing additional costs to operators. This thesis is related to centralized protection and control system developed by ABB. The system will simplify the maintenance of the distribution automation systems by utilizing station computer and IEC 61850 standard.

The purpose of this thesis was to develop a configuration environment for the station computer and extend the configurability of the station computer's software. In the literature part of the thesis the concept of the centralized protection and control is introduced. The requirements for configuration environment are also defined. Additionally, the relevant parts of the IEC 61850 standard are discussed. The software of the station computer is introduced as well as the PCM600 tool used for configuration.

As the results of the thesis, the Connectivity Package extension developed for the PCM600 is described in detail. The modifications done for the station computer's software are introduced as well. The configuration process is based on the IEC 61850 standard and the configuration is described in full extend using the standardized SCL language.

A working configuration environment was developed as the result of this thesis. During the development of the station computer's software, minor issues related to the flexibility of the system were found. The issues should be addressed when developing the next generation of the station computer. As conclusion, the concept of the centralized protection and control is considered usable and the further development of the system is highly encouraged.

ALKUSANAT

Tämän opinnäytetyön toimeksiantaja oli ABB Oy Sähkönjakeluautomaatio-yksikkö. Haluan kiittää Jani Valtaria opinnäytetyön ohjaamisesta ja lukuisista neuvoista projektin eri vaiheissa. Kiitokset kuuluvat myös Hannu-Matti Järviselle opinnäytetyön tarkastamisesta.

Erityiskiitokset kuuluvat perheelleni ja läheisilleni, jotka ovat kannustaneet ja auttaneet minua opintojeni eri vaiheissa. Ilman teitä tämän opinnäytetyön tekeminen ei olisi koskaan ollut mahdollista.

Tampereella 10. huhtikuuta 2011

Erkka Kettunen

SISÄLLYS

LYHENTEET JA MERKINNÄT

ACT	Application Configuration Tool, PCM600-työkalun tarjoama työkalumoduuli sovelluksen konfiguroimiseen.
CCT	Communication Configuration Tool, ABB:n kehittämä työkalu kommunikaation konfiguroimiseen eri laitteiden välillä.
CID	Configured IED Description, IEC 61850 -standardin määrittelemä konfiguroidun suojalaitteen kuvaustiedosto.
ConnPack	Laajennus PCM600-konfigurointityökaluun.
CRW	Common Read and Write, PCM600-työkalun tarjoama työkalu konfiguraation lukemiseen ja kirjoittamiseen.
FC	Functional Constraint, funktionaalinen rajoite.
FTP	File Transfer Protocol.
GOOSE	Generic Object Oriented System Event, IEC 61850 -standardin määrittelemä kommunikaatioprotokolla.
HDF	HiDraw Definition Format, HiDraw-työkalun tiedostomuoto.
HiDraw	ABB:n kehittämä graafinen ohjelmointityökalu.
ICD	IED Capability Description, IEC 61850 -standardin määrittelemä suojalaitteen ominaisuuksien kuvaustiedosto.
IEC	International Electrotechnical Commission, kansainvälinen sähköalan standardointiorganisaatio.
IED	Intelligent Electronic Device, sähköverkon suojalaite.
IID	Instantiated IED Description, IEC 61850 -standardin määrittelemä kuvaustiedosto käyttöönotetulle suojalaitteelle.
LD	Logical Device, looginen laite.
LLN0	Logical node zero, IEC 61850 -standardin määrittelemä looginen noodi.
LN	Logical Node, looginen noodi.
LPHD	Physical device information, IEC 61850 -standardin määrittelemä looginen noodi, joka täytyy löytyä jokaisesta loogisesta laitteesta.
MMS	Manufacturing Message Specification, reaaliaikaiseen tiedonsiirtoon käytetty protokolla.
MU	Merging Unit, liittymisyksikkö.
PCM600	ABB:n kehittämä työkalu suojareleiden konfiguroimiseen.

PST	Parameter Setting Tool, PCM600-työkalun tarjoama työkalumoduuli suojalaitteen asetteluiden konfigurointiin.
RTX	Real Time eXtensions for Windows, IntervalZero-yrityksen valmistama reaaliaikalaajennus Windows-käyttöjärjestelmälle.
SCD	Substation Configuration Description, IEC 61850 -standardin määrittelemä sähköaseman kuvaustiedosto.
SCL	Substation Configuration description Language, sähköaseman konfiguraation kuvauskieli.
SED	System Exchange Description, IEC 61850 -standardin määrittelemä tiedostomuoto järjestelmätietojen vaihtamiseen kahden eri järjestelmän välillä.
SPA	ABB:n kehittämä protokolla sähköasemalla tarvittavaan kommunikaatioon.
SSD	System Specification Description, IEC 61850 -standardin määrittelemä kuvaustiedosto järjestelmän ominaisuuksista.
XEX	ABB:n kehittämä reaaliaikavuorontaja.
XML	Extensible Markup Language, rakenteellinen kuvauskieli.

1 JOHDANTO

Sähkönjakeluautomaatiojärjestelmien käyttöaika on perinteisesti ollut 30–50 vuotta. Toimialan kilpailun vapautuminen ja lainsäädännölliset muutokset ovat muuttaneet järjestelmien käyttöaikaa ja vaatimuksia merkittävästi. Uusien vaatimusten vuoksi toimintojen lisääminen on tullut välttämättömäksi ja perinteisesti ongelma on ratkaistu lisäämällä sähkönverkon suojalaitteiden (Intelligent Electronic Device, IED) toiminnallisuutta. Suojalaitteiden uusimisen lisäksi sähköaseman infrastruktuuriin on jouduttu tekemään useita päivityksiä, jotka ovat vaatineet pitkiä huoltotaukoja. Lisäksi päivitysten vaatimat kustannukset ovat nousseet korkeiksi. [27, 28]

Yhdeksi ratkaisuksi ongelmaan on ehdotettu täysin keskitettyä suojausta. Siirtämällä kaiken toiminnan keskitettyyn asematietokoneeseen voidaan primääristen suojalaitteiden käyttöaikaa pidentää merkittävästi. Uusien vaatimusten ja toiminnallisuuden lisääminen on yksinkertaista, sillä vain keskitetty asematietokone tarvitsee päivittämistä. Asematietokoneesta muodostuu kuitenkin kriittinen piste koko järjestelmän toimivuudelle; mikäli asematietokoneeseen tulee toimintahäiriö, vaarantuu koko järjestelmän toiminta. Käytännön ratkaisuissa vaadittaisiin joko kahdennettu asematietokone tai primäärisiä suojalaitteita järjestelmän toiminnan turvaamiseksi. [27]

Ratkaisuksi on ehdotettu myös järjestelmää, joka hyödyntää sekä keskitettyä asematietokonetta että erillisiä suojalaitteita. Tässä ratkaisussa järjestelmä koostuu keskitetystä asematietokoneesta ja suojareleistä. Kaikkein kriittisimmät toiminnot sijaitsevat edelleen suojareleissä, mutta osa toiminnallisuudesta siirretään asematietokoneeseen. Asematietokoneeseen voidaan näin sijoittaa toiminnallisuutta, jonka odotetaan tarvitsevan päivittämistä uusien vaatimusten vuoksi. Lisäksi keskitettyyn asematietokoneeseen voidaan lisätä toimintoja, joilla ei ole kovia reaaliaikavaatimuksia. [28]

Tämä opinnäytetyö liittyy ABB Oy:n kehittämään keskitettyyn suojaus- ja ohjaustekniikkaan, joka perustuu asematietokoneen ja suojareleiden yhdistelmään. Opinnäytetyön tavoitteisiin kuuluu asematietokoneen ohjelmiston laajentaminen siten, että siinä suoritettavien toimitoimintojen ominaisuuksia voidaan konfiguroida järjestelmän ulkopuolelta. Opinäytetyössä kuvataan myös asematietokoneen konfigurointityökalun toteutus ja tiedonsiirrossa käytettävän tiedostoformaatin määrittely ja vaatimukset.

Luvussa 2 kuvataan asematietokoneen toimintaympäristö ja keskitetyn suojaus- ja ohjaustekniikan toimintaperiaate. Luvussa esitellään myös ABB:n kehittämä järjestelmä ja pohditaan sen eroavaisuuksia suojareleisiin verrattuna. Luku 3 esittelee työssä sovelletta-

van IEC (International Electrotechnical Commission) 61850 -standardin [10] sisällön siltä osin, kun sitä työssä on sovellettu. Luvussa 4 on kuvattu asematietokoneen ohjelmistoympäristö ja ajoympäristön kehittämiseen käytetty työkalu. Luku 5 kuvaa ABB:n kehittämän PCM600-konfigurointityökalun, jonka päälle asematietokoneen konfigurointityökalu kehitettiin. Luku 6 käsittelee ajoympäristön konfiguroinnin ja konfigurointityökalun toteutusyksityiskohtia, sekä arvioi toteutuksen onnistumista. Luku 7 sisältää johtopäätökset ja opinnäytetyön aikana huomatut jatkokehitysajatukset.

2 KESKITETTY SUOJAUS- JA OHJAUSTEKNIIKKA

Tietoliikenneverkkojen kehittyminen on mahdollistanut yhä suurempien sähköverkkojen hallitsemisen keskitetysti yhdestä valvomosta. Samalla yritysfuusioiden myötä valvomon käsittelemän datan määrä on kasvanut voimakkaasti. Myös hajautetut tuotantolaitokset tuovat omat haasteensa keskijänniteverkon ylläpitämiselle ja kehittämiselle. Hajautetun tuotannon myötä jakeluverkkoa tullaan käyttämään eri tavalla, kuin mihin se alun perin suunniteltiin. Uudet vaatimukset on perinteisesti ratkaistu lisäämällä suojalaitteiden toiminnallisuutta, mutta ratkaisu on vaatinut pitkiä huoltotaukoja ja järjestelmän testaamista uudelleen. Ratkaisuksi ongelmaan on esitetty keskitettyä suojaus- ja ohjaustekniikkaa. [28, 22]

2.1 Yleiskuvaus tekniikasta

Sähkönjakeluautomaatiojärjestelmien päivittäminen on ollut kallista ja vaatinut pitkiä huoltokatkoja. Keskitettyä suojaus- ja ohjaustekniikkaa on ehdotettu ratkaisuksi vaatimusten täyttämiseksi ilman pitkiä huoltokatkoja. Keskitetty suojaus- ja ohjaustekniikka mahdollistaa suojalaitteiden toiminnallisuuden siirtämisen keskitettyyn asematietokoneeseen, jonka ohjelmiston päivittäminen mahdollistaa vaatimusten täyttämisen ilman primääristen suojalaitteiden uusimista. [29, 28]

Keskitetty suojaus- ja ohjaustekniikka tuo uusia mahdollisuuksia kehittää tarkempia ja monimutkaisempia suojausalgoritmeja, sillä asematietokone voi kerätä mittaustietoja useista suojareleistä. Suojareleissä vaaditaan kovia reaaliaikavaatimuksia, eikä niihin ole voitu tämän takia kehittää ylimääräisiä laskentakapasiteettia vaativia suojausalgoritmeja ilman valmistuskustannusten merkittävää kasvua. Asematietokonetta voidaan käyttää hyödyksi tällaisissa tilanteissa. Asematietokone voi myös kerätä vikatietoja ja raportteja automaattisesti suojareleiltä ja koota niistä yhtenäisen koko asemaa koskevan raportin. Yhtenäisen raportin tuottaminen helpottaa myös sähkönjakeluyhtiöiden ylläpitotehtäviä. [28]

Toimialalla on kehitetty kaksi erilaista keskitettyyn suojaus- ja ohjaustekniikkaan liittyvää ratkaisua. Ensimmäinen lähestymistapa luottaa kahdennettuun asematietokoneeseen ja lähes kaikki suojaustoiminnot toteutetaan ohjelmistolla asematietokoneessa. Toinen lähestymistapa on ollut säilyttää reaaliaikaiset suojaustoiminnot suojareleissä ja siirtää muu tominnallisuus keskitettyyn asematietokoneeseen. [23, 28]

2.1.1 Kahdennettu asematietokone

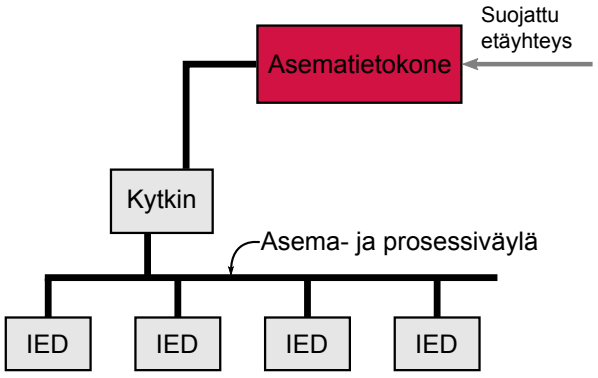
Sähköaseman kaikki toiminnot on mahdollista keskittää yhteen asematietokoneeseen, jonka kautta kaikki tieto ja ohjauskomennot kulkevat kennotason laitteille, kuten esimerkiksi katkaisijoille. Tällaisessa järjestelmässä saavutetaan yksinkertainen kokoonpano, mutta yksittäinen asematietokone muodostuu järjestelmän toiminnan kannalta kriittiseksi pisteeksi. Ongelma voidaan ratkaista kahdentamalla asematietokone, jolloin toisen koneen vioittuessa järjestelmä on vielä toimintakykyinen. [23]

Järjestelmä koostuu virta-, jännite- ja katkaisijamoduuleista, jotka yhdistetään asematietokoneisiin. Suojaustoimintoja voidaan lisätä järjestelmään liittämällä IEC 61850 -standardin mukainen suojalaite keskitettyyn asematietokoneeseen. Suurin osa järjestelmän toiminnoista toteutetaan ohjelmistolla, joka käyttää muun laitteiston lähettämää mittaustietoa. Mahdolliset vikatilanteet voidaan havaita ja paikantaa mittaustietoa analysoimalla. [6, 29]

2.1.2 Suojareleitä ja asematietokonetta yhdistelevä ratkaisu

Vaihtoehtoinen ratkaisu yhdistelee suojareleiden toimintoja asematietokoneen kanssa. Kaikkein tärkeimmät suojaustoiminnot sijaitsevat edelleen suojareleissä ja asematietokoneelle siirretään vain osa suojaustoiminnoista. Asematietokone toimii näin toissijaisena suojana ja tuo lisätoiminnallisuutta sähköasemalle. Järjestelmän päivittäminen onnistuu myös ilman mittavia käyttökatkoja, sillä primäärisen suojauksen toiminta ei riipu asematietokoneesta lainkaan. [28]

Järjestelmän toiminta perustuu IEC 61850 -standardin mukaisiin kommunikaatioprotokolliin ja sähköaseman mallintamiseen. Suojareleet lähettävät mittadataa asematietokoneelle standardin määrittelemää prosessiväylää käyttäen. Asemaväylää käytetään suojareleiden väliseen kommunikaatioon GOOSE (Generic Object Oriented Substation Event) -viestien välityksellä. Kuvassa 2.1 on esitetty järjestelmän kokoonpano. [28]



Kuva 2.1. *Suojareleitä ja asematietokonetta yhdistelevän ratkaisun kokoonpano. [28]*

2.2 Keskitetyn asematietokoneen ja suojareleen eroavaisuudet

2.2.1 Toimintaperiaatteet ja käyttötarkoitus

Suojareleen ensisijainen tehtävä on suojata sähköverkkoa epänormaaleilta tilanteilta. Epänormaalissa tilanteessa rele lähettää ohjauskäskyn katkaisijalle ja viallinen osa irroitetaan terveestä verkosta. Suojarele ei yksin pysty suoriutumaan suojaustehtävistä, vaan se tarvitsee avukseen myös muita komponentteja. [25]

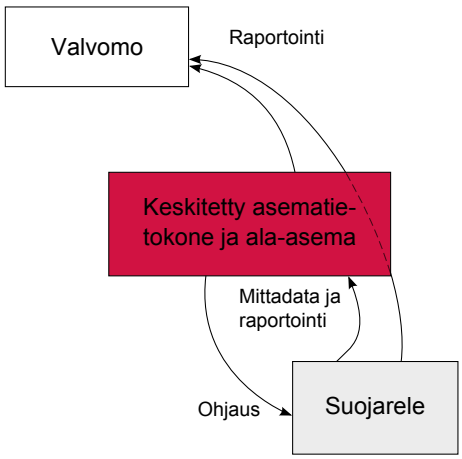
Mitta-arvot rele saa mittalaitteelta analogisina arvoina. Mittalaitteita ovat esimerkiksi virta- ja jännitemuuntajat. Suojarele muuntaa mitta-arvot digitaaliseksi dataksi jatkokäsittelyä ja suojaustoimintoja varten. Mikäli suojareleen kommunikaatio on suunniteltu IEC 61850 -standardin mukaisesti, voi suojarele edelleen lähettää digitaaliset mitta-arvot kommunikaatioverkkoon. [25, 19]

Keskitetty asematietokone toimii ainostaan ”virtuaalisilla” kommunikaatioväylältä saatavilla mitta-arvoilla. Keskitetyn asematietokoneen toiminta on rakennettu IEC 61850 -standardin määrittelemän kommunikaation ympärille, eikä se tämän vuoksi sisällä suojareleen kaltaista fyysistä yhteyttä mittalaitteisiin. [28]

Keskitetty asematietokone voi vastaanottaa mitta-arvoja usealta erilliseltä mittalaitteelta, kun taas suojarele on tavallisesti kytkettynä vain yhteen mittalaitteeseen kerrallaan. Keskitetty asematietokone voi näin hyötyä suuremmasta datamäärästä ja sitä varten voidaan kehittää täysin uudenlaisia suojausalgoritmeja. [28]

2.2.2 Sijainti järjestelmässä

Suojarele sijoittuu koko asematasolla kaikkein alhaisimmalle tasolle, eikä se kommunikoi hierarkiassa alaspäin. Keskitetty asematietokone sijoittuu taas yhtä tasoa korkeammalle ja se kommunikoi sekä alaspäin suojareleille että ylöspäin valvomoon. Kuvassa 2.2 on havainnollistettu komponenttien sijaintia järjestelmässä. [28]

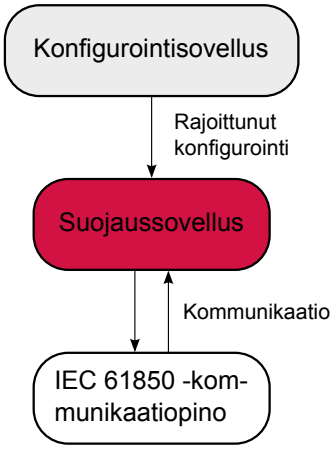


Kuva 2.2. *Keskitetyn asematietokoneen ja suojareleiden sijainti suhteessa valmomoon.*

Kuvasta nähdään, että molemmat komponentit raportoivat valvomoon toiminnastaan. Keskitetyllä asematietokoneella on mahdollisuus kerätä raportteja alatasolta ja koota niistä koko asemaa koskeva raportti suojauksen toiminnasta. Suojareleet raportoivat valvomoon ala-aseman kautta, joka voi toimia samalla myös keskitettynä asematietokoneena. Tällöin asematietokone toimii vain välittävänä komponenttina, eikä se käsittele raportteja lainkaan.

2.3 Nykyinen järjestelmä

ABB:n kehittämä keskitetty suojaus- ja ohjaustekniikka koostuu asematietokoneessa toimivasta ohjelmistosta, jonka konfiguroitavuus on rajoitettua. Järjestelmä toimii normaalilla PC-tietokoneella Windows-käyttöjärjestelmän päällä. Reaaliaikaiset ominaisuudet on toteutettu käyttämällä reaaliaikalaajennusta. Järjestelmän osakomponentit on esitetty kuvassa 2.3. [2]



Kuva 2.3. *Nykyisen järjestelmän osakomponentit.*

Konfigurointi on toteutettu hyvin rajallisesti, eikä käytönaikaista konfigurointia voida tehdä lainkaan. Konfigurointi vaatii myös manuaalisen järjestelmän pysäyttämisen ja uudelleenkäynnistämisen. Lisäksi konfiguraatio on tallennettu standardoimattomaan tekstitiedostoon, jonka muokkaaminen on käyttäjän vastuulla. [2]

Järjestelmä kommunikoi IEC 61850 -standardin mukaisesti muiden suojareleiden kanssa. Lisäksi järjestelmä voi vastaanottaa mittadataa eri mittamuuntajilta hyödyntämällä IEC 61850 -standardin mukaista kommunikaatioväylää. [2]

2.4 Konfiguroinnin vaatimukset

Konfiguroinnille asetettiin seuraavanlaiset vaatimukset:

- Suojausohjelman sisäistä logiikkaa tulee pystyä muokkaamaan.
- IEC 61850-9-2 -standardin mukaista dataa lähettävä osa pitää pystyä mallintamaan ja siihen liittyvä mittasignaali tulee olla liitettävissä muihin toimilohkoihin.

– IEC 61850-8-1 -standardin mukaista kommunikaatiota pitää pystyä konfiguroimaan vapaasti.

– Konfigurointi tulee onnistua samalla työkalulla, kuin nykyisten ABB:n valmistamien releiden konfigurointi.

– Konfigurointityökalu ei ole sidottu fyysisesti asematietokoneeseen, vaan sitä tulee pystyä käyttämään eri työpisteeltä.

– Konfigurointiprosessin tulee noudattaa IEC 61850 -standardin kuvaamaa prosessia.

– Ajonaikainen konfigurointi tulee olla mahdollista. Jos uudelleenkäynnistys on tarpeellinen, tulee sen olla mahdollisimman nopea käyttökatkon minimoimiseksi.

Standardin asettamat vaatimukset on kuvattu tarkemmin luvussa 3. Käytettävä konfigurointityökalu ja siihen liittyvät laajennukset on esitetty luvussa 5.

2.4.1 Järjestelmätason konfigurointi

IEC 61850 -standardin osa 8-1 määrittelee releiden välisen kommunikaation asemaväylää käyttäen. Kommunikaatiossa käytetään GOOSE-viestejä. GOOSE-viestien konfigurointi vaatii järjestelmätason työkalun, jolla kommunikaatio eri laitteiden välillä voidaan määrittää. Konfiguraatio kirjoitetaan standardoituun tiedostoon, jossa määritellään tarkemmin mitä viestejä väylälle lähetetään.

GOOSE-kommunikaatio sisältää ohjauskomentoja ja tilatietoja. Asemaväylän toimintaa on esitelty tarkemmin kohdassa 3.2.1.

2.4.2 Järjestelmän monitorointi

Suojareleen tilaa tulee pystyä monitoroimaan myös ajonaikaisesti. Ajonaikainen monitorointi sisältää eri signaalien arvojen tarkkailua ja esimerkiksi häiriötallenteiden lukemista laitteesta. IEC 61850 -standardi määrittelee yhtenäisen kommunikaation raportoinnille.

Keskitetystä asematietokoneesta tulee olla mahdollista tarkkailla koko verkon tilaa. ABB:llä on kehitetty COM600-asematietokone, jossa verkkoa voidaan valvoa asemakuvan välityksellä.

3 IEC 61850 -STANDARDI

Sähköverkon suojalaitteiden kehittyminen elektromekaanisista laitteista mikroprosessoriohjatuiksi järjestelmiksi on luonut mahdollisuuden lisätä suojalaitteiden toiminnallisuutta. Suojalaitteet sisältävät monia erilaisia toimintoja kuten suojausta, paikallista ja etänä tapahtuvaa ohjausta sekä raportointia. Toimintojen määrän kasvaessa myös laitteiden välinen kommunikaatio on muodostunut tärkeäksi osaksi järjestelmän toimintaa. Ennen IEC 61850 -standardia laitevalmistajat ovat käyttäneet omia suljettuja protokollia ja eri valmistajien laitteiden väliseen kommunikaatioon on vaadittu monimutkaisia protokollamuuntimia. [10]

Toimialalla oli selvä tarve luoda yhtenäiset kommunikointimenetelmät eri valmistajien laitteiden välille. Tämä tarve mahdollisti IEC 61850 -standardin kehittämisen. Standardin tavoitteena on kehittää kommunikointistandardi, joka mahdollistaa laitteiden yhteentoimivuuden säilyttäen kuitenkin mahdollisuuden uusille teknologisille ratkaisuille. Standardi mallintaa suojalaitteiden suojausfunktiot ja niiden välittämän datan, mutta ei rajoita suojausfunktioiden määrää tai ominaisuuksia millään tavalla. [10]

3.1 Mallintaminen

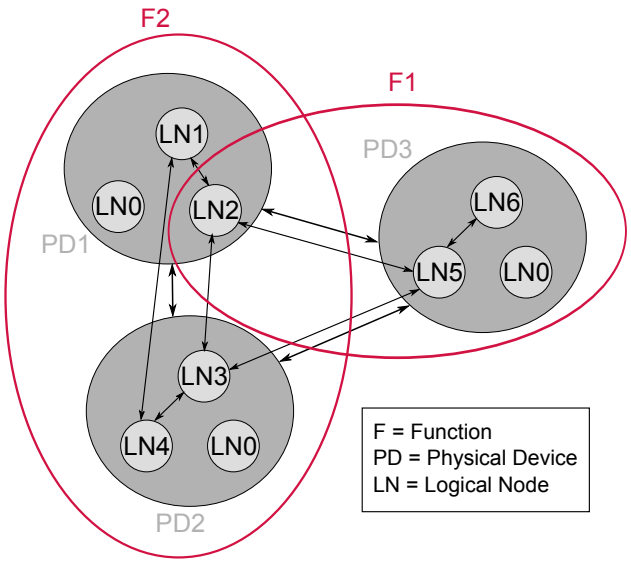
Suojalaitteen eri osien yksityiskohtainen mallintaminen on tärkeä osa IEC 61850 -standardia. Standardi mallintaa sähköaseman eri komponentit pienempien osakokonaisuuksien avulla ja määrittelee kommunikaation niiden välille. Esimerkiksi suojalaitteesta mallinnetaan sen sisältämä toiminnallisuus ja fyysiseen laitteeseen liittyvät ominaisuudet. Mallintamisen tarkoituksena on luoda yhtenäiset termit ja määritelmät sähköaseman eri osille. Tämä mahdollistaa eri valmistajien tuotteiden käsittelemisen yhdenvertaisina kokonaisuuksina. [10]

3.1.1 Sovellusnäkymä – toimilohkojen mallintaminen

IEC 61850 -standardi mallintaa suojalaitteiden toiminnallisuuden funktioiden eli toimilohkojen avulla. Yksi suojalaite voi sisältää eri määrän toimilohkoja riippuen laitteen suorituskyvystä, kustannusrajoituksista ja teknologisista ratkaisuista. Toimilohko voidaan hajottaa pienempiin osakokonaisuuksiin ja suorittaa niitä fyysisesti erillisissä suojalaitteissa. Toimilohkon toiminnallisuus voidaan toteuttaa myös hajautetusti hyödyntämällä laitteiden välistä kommunikaatiota. [10]

Funktiot voidaan jakaa kolmelle eri tasolle: sähköasema-, kenno- ja prosessitasolle. Sähköaseman laitteita ovat esimerkiksi asematietokone, operaattorin työpiste ja rajapinnat etähallintaan. Kennotason laitteisiin kuuluu esimerkiksi suojaus-, ohjaus- ja valvontayksiköitä jokaista kennoa kohti. Prosessitason komponentteja ovat puolestaan esimerkiksi sensorit ja katkaisijat. Funktioita käytetään kuvaamaan näiden eri laitteiden toimintoja ja ominaisuuksia. [12]

Funktiot koostuvat pienemmistä osakomponenteista, joita nimitetään standardissa *loogisiksi noodeiksi* (logical node, LN). Käytännössä yksittäinen looginen noodi voi mallintaa esimerkiksi kolmivaiheisen ylivirtasuojan toiminnan. Yhden funktion toteuttavat loogiset noodit voivat sijaita eri laitteissa kuvan 3.1 esittämällä tavalla. Esimerkiksi funktio F1 koostuu neljä erillisestä loogisesta noodista, joista yksi sijaitsee eri laitteessa kuin kolme muuta. Loogisten noodien väliset nuolet tarkoittavat tiedonsiirtoa niiden välillä. Eri laitteissa sijaitsevien loogisten noodien välinen yhteys on mahdollinen vain silloin, kun myös laitteiden välillä on yhteys. Tätä yhteyttä kuvaa nuolet fyysisten laitteiden välillä. [12]



Kuva 3.1. *Funktion koostuminen loogisista laitteista ja noodeista. [12]*

Loogiset noodit kommunikoivat lähettämällä ja vastaanottamalla dataa. Vastaanottava LN tietää, mitä dataa se tarvitsee toiminnon suorittamiseen, ja sen on pystyttävä myös tarkastamaan vastaanotetun datan laatu ja oikeellisuuus. Lähettävä looginen noodi voi täyttää suurimman osan tästä tiedosta, mutta vastaanottava noodi tekee lopullisen päätöksen datan kelvollisuudesta. Standardin osassa 7 määritellään laatuattribuutit lähetetylle datalle. Laatuattribuutit ovat tiedonsiirron kannalta tärkeitä, sillä loogiset noodit voivat sijaita eri laitteissa ja tiedonsiirtovirheet laitteiden välillä ovat mahdollisia. [12]

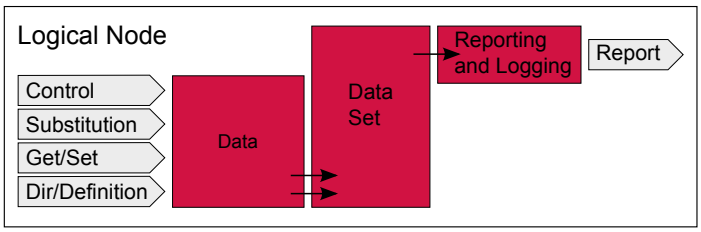
Standardi määrittelee 92 eri loogista noodia yleisimmin käytetyille toiminnoille. Loogiset noodit määritellään tarkemmin standardin osassa 7-4 ja siinä esitettyjen sääntöjen perusteella on mahdollista luoda uusia loogisia noodeja. Standardi jakaa loogiset noodit

eri ryhmiin toimintojen perusteella, ja ryhmätunnus lisätään myös loogisten noodien nimiin. Esimerkiksi suojaustoimintoja mallintavien loogisten noodien nimet alkavat aina ryhmätunnuksella *P* (Protection), joten niiden erottaminen muista loogisista noodeista on tunnuksen perusteella helppoa. [15, 17]

Loogiset noodit ja data

Loogiset noodit sisältävät eri määrän data-attribuutteja. Standardi määrittelee jokaiselle loogiselle noodille tietyn määrän pakollisia data-attribuutteja riippuen sen toiminnallisuudesta. Nämä data-attribuutit muodostavat tiedonsiirron perustan eri laitteiden välillä. Data-attribuuttien määritteleminen mahdollistaa eri valmistajien laitteiden välisen kommunikaation, sillä eri loogisten noodien sisältämä data on tarkasti määritelty. [15]

Data-attribuutit jaetaan eri *dataluokkiin*, jotka määrittelevät ne palvelut ja operaatiot, joita datan perusteella voidaan toteuttaa. Dataluokkaan kuuluvan datan käyttöä rajoitetaan funktionaalisilla rajoitteilla (functional constraint, FC). Näin osa datasta on pelkästään lukemista varten ja osan päälle voidaan esimerkiksi kirjoittaa. Loogisen noodin rakenne on esitetty kuvassa 3.2. Harmailla nuolilla on kuvattu loogiseen noodiin kohdistuvia operaatioita ja palveluita. [15]



Kuva 3.2. *Loogisen noodin rakenne. [15]*

Kuvasta nähdään myös, että dataa voidaan kerätä nipuiksi (data set) ja lähettää eteenpäin tai tallentaa esimerkiksi häiriötallenteeksi myöhempää tarkastelua varten. *Control*-palvelu mallinnetaan myös datana ja sen välityksellä on mahdollista kontrolloida laitetta. Muiden palveluiden välityksellä voidaan lukea ja muokata loogisen noodin sisältämää dataa. [15]

Loogiset noodit ja niiden sisältämä data muodostavat konseptin, joilla voidaan mallintaa reaalimaailman laitteita ja niiden toiminnallisuutta. Loogiset noodit toimivat säiliönä datalle ja ne voidaan sijoittaa mihin tahansa suojalaitteessa. Esimerkiksi laite, joka lukee tietoa loogisesta noodista voidaan mallintaa toisena loogisena noodina. Tiedonsiirto loogisten noodien välillä voidaan mallintaa niiden tarjoamien palveluiden avulla. [15]

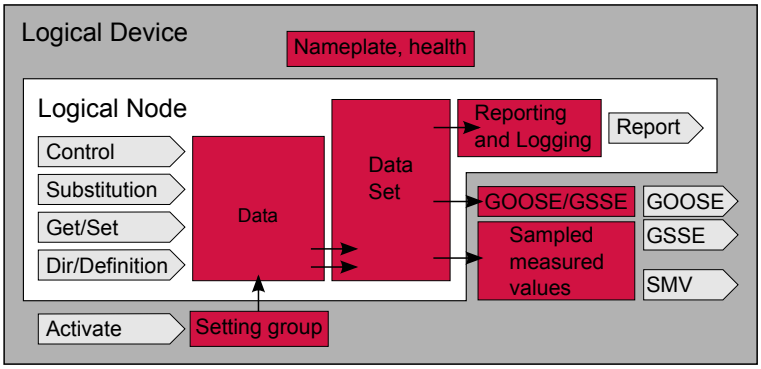
3.1.2 Laitenäkymä – suojalaitteen mallintaminen

Suojalaitteet koostuvat ohjelmistopohjaisista toimilohkoista, joiden tuottamaa dataa suojalaitteet välittävät kommunikaatioverkkoon. Näitä ominaisuuksia mallinnetaan IEC 61850

-standardissa loogisina noodeina ja niihin liittyvänä datana. Suojalaitteet sisältävät myös tietoa niistä itsestään, kuten resurssitietoa ja tietoa siihen kytketyistä laitteista. Suojalaitteet tuntevat myös kommunikaatioyhteydet muihin laitteisiin ja tiettyyn kommunikaatiojärjestelmään. [15]

Kommunikaatio- ja resurssitietoja ei voida esittää loogisten noodien avulla ja tätä varten luotiin *loogisen laitteen* (logical device, LD) konsepti. Looginen laite koostuu pääasiassa loogisista noodeista ja laitteeseen liittyvistä palveluista. Tällaisia palveluita ovat esimerkiksi GOOSE-viestien ja mittausnäytearvojen lähettäminen sekä asetteluryhmät. Looginen laite kuuluu aina yhteen fyysiseen laitteeseen, mutta fyysiseen laitteeseen voi kuulua useita loogisia laitteita. [15]

Yhteen loogiseen laitteeseen voi kuulua useita loogia noodeja. Standardi määrää, että jokaiseen loogiseen laitteeseen täytyy kuulua aina looginen noodi nimeltä LLN0 (Logical node zero). LLN0 sisältää yleisiä tietoja loogisesta laitteesta. Lisäksi standardi vaatii, että jokaiseen loogiseen laitteeseen kuuluu LPHD (Physical device information) -niminen noodi. LPHD sisältää tietoja siitä fyysisestä laitteesta, jossa se sijaitsee. Kuvassa 3.3 on esitetty loogisen laitteen rakenne. [15]



Kuva 3.3. *Loogisen laitteen rakenne. [15]*

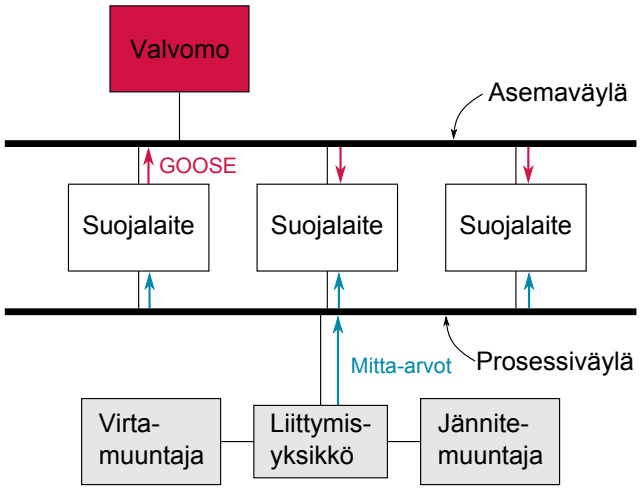
Kuvasta nähdään, että looginen laite koostuu loogisista noodeista ja laitteeseen liittyvistä palveluista, kuten asetteluryhmistä ja GOOSE-kommunikaatiosta. Jokaiseen loogiseen laitteeseen kuuluu lisäksi tunniste- ja tilatiedot. [15]

Loogisen laitteen avulla standardi määrittelee kommunikaation eri laitteiden välille. Loogisen laitteen tietojen perusteella voidaan selvittää millaista kommunikaatioyhteyksiä laite tukee ja millaista dataa välitetään esimerkiksi GOOSE-viestien välityksellä.

3.2 Kommunikaatio

IEC 61850 -standardi määrittelee kaksi erillistä kommunikaatioväylää: asema- ja prosessiväylän. Asemaväylä määrittelee suojalaitteiden välisen kommunikaation, jota käyttämällä suojalaitteet voivat välittää tilatietoa toisilleen. Prosessiväylä on puolestaan tarkoitettu

mitta- ja suojalaitteiden väliseen tiedonsiirtoon. Mittalaitteet lähettävät virta- ja jännitearvoja prosessiväylälle liittymisyksikön välityksellä. Suojalaitteet poimivat tarvitsemansa mitta-arvot suoraan prosessiväylältä. Kuvassa 3.4 on havainnollistettu standardin määrittelemiä kommunikaatioväyliä.



Kuva 3.4. *IEC 61850 -standardissa kuvatut kommunikaatioväylät.*

Kuvaan on merkitty myös kommunikaatioon käytettävät kommunikointimenetelmät eri väreillä. Kommunikaation eri osapuolia ja käyttötarkoituksia käsitellään tarkemmin kohdissa 3.2.1 ja 3.2.2.

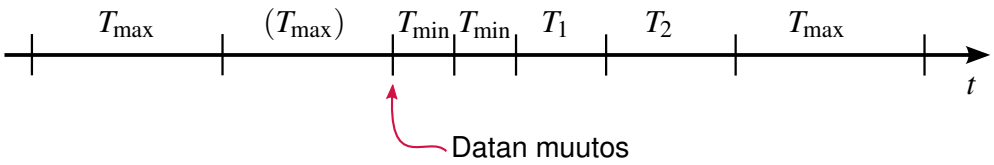
3.2.1 Asemaväylä

Asemaväylä on tarkoitettu suojalaitteiden väliseen kommunikaatioon. Suojalaitteet lähettävät GOOSE-viestejä asemaväylälle, josta muut suojalaitteet poimivat niitä itseään kiinnostavat viestit. GOOSE-viestit lähetetään suoraan Ethernet-verkkoon broadcast-paketteina käyttäen linkkikerrosta. Linkkikerrosta käyttämällä saavutetaan GOOSE-viesteille asetetut reaaliaikavaatimukset. [18]

Asemaväylä helpottaa sähköaseman infrastruktuurin rakentamista, sillä laitteiden väliset johdinvedot voidaan korvata Ethernet-väylällä. Johdinvetojen määrä putoaa Ethernet-väylän myötä merkittävästi, jolloin myös asennustöiden ja mahdollisten asennusvirheiden määrä pienenee. Ethernet-väylä mahdollistaa myös pienemmät vasteajat laitteiden välisessä kommunikaatiossa. Pienemmät vasteajat mahdollistavat laitteiden nopeamman ja tarkemman toiminnan. [9]

Kommunikaatio laitteiden välillä perustuu datanippujen lähettämiseen asemaväylälle. Jokainen suojalaite määrittelee datanipun, joka sisältää tietoja suojalaitteen toiminnasta ja tilasta. Kun jokin datanipun sisältämä tieto muuttuu, lähetetään kaikki datanipun sisältämät arvot asemaväylälle. Datanipun sisältö lähetetään väylälle broadcast-viestinä, eikä suojalaite tiedä ovatko vastaanottajat saaneet viestin vai ei. Tämän vuoksi viestejä lähetetään sovitulla sekvenssillä, jolla pyritään varmistamaan, että viesti tavoittaa vastaanottajat. [18]

Kuvassa 3.5 on esimerkki GOOSE-viestien lähetyssekvenssistä. Suojalaite lähettää muuttuneen datanipun heti sen sisällön muuttuessa ja uudelleen ajan T_{min} kuluttua. Tämän jälkeen viestiä lähetetään suojalaiteriippuvaisella sekvenssillä (T_1 ja T_2). Standardi ei ota kantaa sekvenssiin, jolla dataa lähetetään kahden ensimmäisen lähetyskerran jälkeen, vaan jokainen valmistaja voi toteuttaa sen parhaaksi katsomallaan tavalla. Suojalaitekohtaisen sekvenssin päätyttyä viestien lähettämistä jatketaan sen maksimiarvolla T_{max} seuraavaan datan muutokseen saakka. Ennen datan muutosta oleva lähetysväli voi lyhentyä datan muutoksen vuoksi. [9, 18]



Kuva 3.5. *Esimerkki GOOSE-viestien lähettämissekvenssistä. [9, 18]*

GOOSE-viesteihin liittyy myös laatuattribuutteja, joiden avulla tieto viestien sisällön luotettavuudesta voidaan välittää muille suojalaitteille. Näin muut suojalaitteet voivat käyttää esimerkiksi oletusarvoja epäluotettavan tiedon sijaan. Testausta varten ovat omat laatuattribuuttinsa, joita käyttämällä järjestelmän toimintaa voidaan testata kytkemättä suojalaitteita oikeisiin mittalaitteisiin. [9]

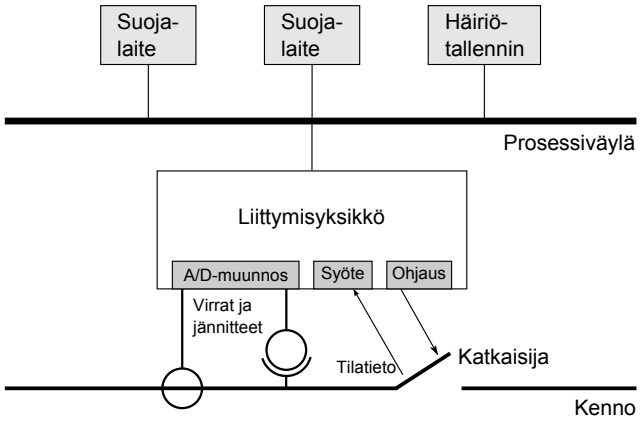
3.2.2 Prosessiväylä

Perinteisesti suojalaitteet on kytketty mittalaitteisiin useilla johtimilla ja uuden mittalaitteen lisääminen järjestelmään on vaatinut uusien johdinvetojen tekemisen. IEC 61850-9-2 määrittelee prosessiväylän käsitteen, jolla tarkoitetaan Ethernet-pohjaista väylää, johon sekä suoja- että mittalaitteet on kytketty. Prosessiväylän käyttäminen vähentää johdinvetojen määrää ja uusien suojalaitteiden lisääminen sähköasemalle helpottuu huomattavasti. [7, 10]

Mittalaitteet kytketään prosessiväylään *liittymisyksikköä* (merging unit, MU) käyttämällä. Liittymisyksikkö lähettää yhdistelmän mittalaitteilta saatavista virta- ja jännitearvoista prosessiväylälle. MU muuntaa mittalaitteilta saadut analogiset arvot digitaalisiksi ennen mitta-arvojen lähettämistä. [7, 11]

Liittymisyksikkö voi myös sisältää toimintoja esimerkiksi katkaisijoiden ohjaamiseen. Liittymisyksikkö vastaanottaa katkaisijalta tilatietoja lähettää niitä edelleen prosessiväylälle. Katkaisijan tilatiedot välitetään prosessiväylältä valvomoon esimerkiksi asematietokoneen välityksellä. Kuvassa 3.6 on esimerkki prosessiväylän käyttämisestä. [24]

Ensimmäisiä käytännön toteutuksia varten toimialalla määriteltiin karsittu versio 9-2-standardin osasta: IEC 61850-9-2 LE (Lite Edition). Karsitun version tarkoituksena on nopeuttaa standardin käyttöönottoa määrittelemällä totetusyksityiskohtia, joihin IEC 61850



Kuva 3.6. *Esimerkki prosessiväylän käytöstä. Prosessiväylään on kytketty liittymisyksikkö, joka välittää mitta-arvoja suojalaitteille.*

ei ota kantaa. IEC 61850-9-2 LE määrittelee esimerkiksi loogisen laitteen liittymisyksikön mallintamiseen ja kaksi näytteistystaajuutta mitta-arvojen näytteistämistä varten. [8]

3.3 SCL-kieli

IEC-61850 -standardin osa 6 määrittelee sähköaseman konfiguraation kuvauskielen (Substation Configuration description Language, SCL). Sitä käytetään suojalaitteen konfiguraation ja suojalaitteiden välisen kommunikaation kuvaamiseen. SCL-kielellä voidaan myös kuvata sähköasema-automaatiojärjestelmän ja kytkinkentän väliset yhteydet. [13]

SCL-kieli mahdollistaa yksittäisen suojalaitekonfiguraation liittämisen järjestelmätason konfiguraatioon yhteensopivasti siten, että myös eri valmistajien työkaluja voitaisiin käyttää yhdessä. SCL-kieli perustuu XML-kielen (Extensible Markup Language) versioon 1.0. [13]

3.3.1 Konfigurointiprosessi SCL:n näkökulmasta

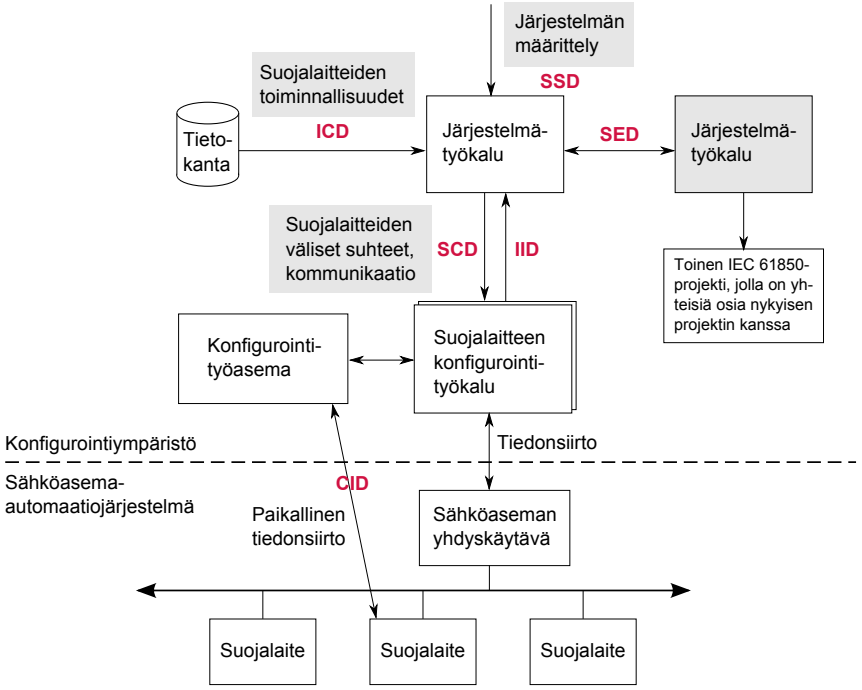
Yleensä sähköasema-automaatiojärjestelmän konfigurointi aloitetaan asentamalla valmiiksi konfiguroituja suojalaitteita sähköasemalle. Vaihtoehtoisesti suojalaitteiden eri ominaisuuksia otetaan käyttöön sitä mukaa, kun automaatiojärjestelmässä on niille tarvetta. Käytännössä sovelletaan molempia vaihtoehtoja sekaisin ja tämä asettaa myös SCL-kielelle vaatimuksia, sillä molemmat konfigurointivaihtoehdot tulee olla mallinnettavissa SCL-kielellä. [13]

Standardin mukaan SCL-kieli on rajattu kolmeen eri tarkoitukseen:

1. sähköasema-automaatiojärjestelmän toiminnalliseen määrittelyyn,
2. suojalaiteen toiminnallisuuden mallintamiseen ja
3. järjestelmätason konfiguraation mallintamiseen. [13]

Kuvassa 3.7 on esitetty konfigurointiprosessin eteneminen ja tiedon välittyminen SCL-kielen välityksellä on merkitty harmaalla taustavärillä. Standardi määrittelee erikseen

käsitteet *järjestelmätyökalu* ja *suojalaitteen konfigurointityökalu*, mutta käytännön työkalutoteutus voi sisältää molemmat työkalut yhdessä sovelluksessa. Järjestelmätyökalu on suojalaitteista riippumaton korkeimman tason työkalu, jolla voidaan käsitellä useiden eri suojalaitteiden konfiguraatioita. Järjestelmätyökalua käyttämällä konfiguraatioon lisätään suojalaitteita koskevia sähköasematason tietoja. Tämä konfiguraatio käsitellään vielä erikseen suojalaitteen konfigurointityökalulla, jolla jokainen suojalaite konfiguroidaan toimimaan osana järjestelmää. [13, 14]



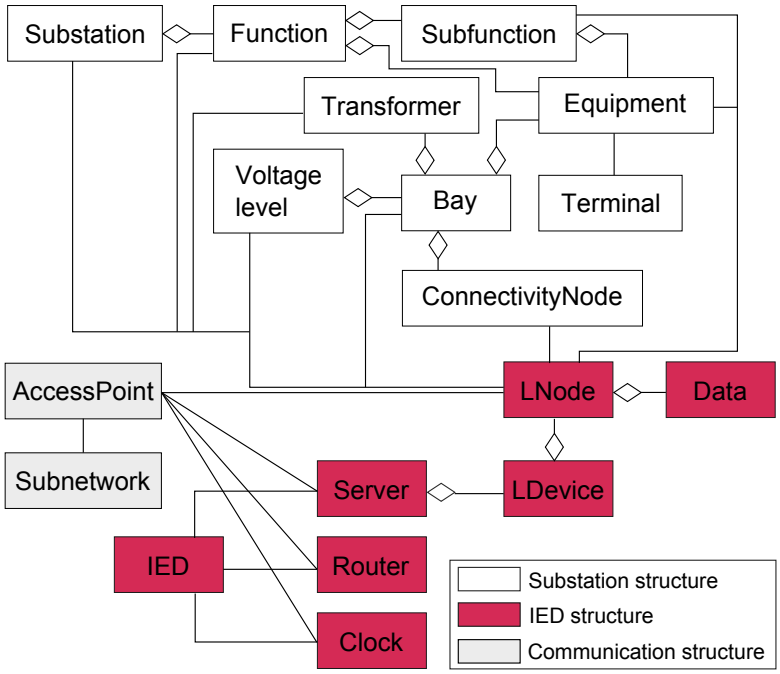
Kuva 3.7. *Konfigurointiprosessi ja tiedonsiirto SCL-tiedostoa käyttäen. Punaisella värillä on merkitty kohdassa 3.3.3 esiteltäviä SCL-kielen tiedostomuotoja. Mukailtu lähteistä [13, 14].*

Konfiguraatio voidaan siirtää suojalaitteeseen käyttäen paikallista kommunikaatioyhteyttä. Standardi ei ota kantaa käytettävään protokollaan tai tiedonsiirtotapaan. Tiedonsiirto voi tapahtua myös käyttäen IEC 61850-7-2 mukaista protokollaa, jolloin SCL:n käyttäminen tiedostomuotona olisi yksi mahdollinen vaihtoehto. Yksittäisen suojalaitteen parametrisointi on myös mahdollista käyttäen osassa 7-2 määriteltyjä palveluita ja MMS (Manufacturing Message Specification) -protokollaa. [13, 16]

3.3.2 Objektimalli

SCL-kielen objektimalli kuvaa sähköasema-automaatiojärjestelmän käyttäen kohdassa 3.1 esiteltyjä loogisia noodeja ja laitteita. SCL-kielellä kuvataan miten eri suojausfunktiot koostuvat loogisista noodeista ja miten ne kommunikoivat muiden loogisten noodien kanssa.

Objektimalli koostuu kolmesta eri perusosasta: sähköasemasta, tuotteista ja kommunikaatiosta. Sähköaseman mallissa kuvataan kytkentäkentän laitteisto ja niiden väliset kytkennät. Tuotteeseen kuuluu kaikki sähköasema-automaatiotuotteisiin liittyvät komponentit, kuten suojalaitteet ja katkaisijat. Kommunikaatio sisältää aliverkot ja yhteyspisteet sekä liitynnät eri komponenttien välillä. Kuvassa 3.8 on esitetty UML-kaavio objektimallista. [13]



Kuva 3.8. *SCL-kielen objektimalli. Mukailtu lähteestä [13].*

Kuvassa esitetyllä IED-komponentilla kuvataan suojalaitteen eri ominaisuudet, kuten toimilohkot ja laitteen tarjoamat kommunikaatioyhteydet. Toimilohkot kuvataan loogisia noodeja käyttäen ja loogisten noodien datasisältö kuvataan erillisen datamallin avulla. Suojalaite liitetään osaksi sähköaseman kommunikaatiota liityntäpisteitä (access point) hyödyntämällä.

Kaavioon on merkitty järjestelmän eri osat eri väreillä. Kaaviosta nähdään myös SCL-kielen hierarkinen rakenne: jokainen ylätason komponentti koostuu hiearkisesta alemman tason komponenteista. Looginen noodi (kuvassa LNode) toimii yhdistävänä tekijänä järjestelmän eri osien välillä. [13]

3.3.3 Tiedostomuodot

SCL-kielen objektimalli tallennetaan XML-tiedostoon ja järjestelmän eri osia kuvataan erityyppisillä SCL-tiedostoilla. Kuvaan 3.7 on merkitty punaisella värillä eri tiedostomuotojen sijainti konfiguroinnin näkökulmasta. Eri tiedostomuotojen tarkoituksena on eriyttää eri konfigurointivaiheet toisistaan ja mahdollistaa parempi yhteensopivuus eri valmistajien työkalujen välillä. [13]

Standardin ensimmäinen versio [13] määrittelee seuraavanlaiset SCL-tiedostot:

– ICD (IED Capability Description) -tiedostolla kuvataan suojalaitteen suorituskyky ja ominaisuudet. ICD-tiedosto sisältää vain yhden suojalaitteen kuvaavan XML-elementin. Tiedostoon kuuluu lisäksi suojalaitteen datasisällön kuvaamiseen tarkoitetut datamallit.

– SSD (System Specification Description) -tiedostolla kuvataan sähköaseman rakenne SCL-kielen objektimallia käyttäen. Järjestelmätyökalu käyttää SSD-tiedostoa sähköaseman yleiskuvauksena, johon liitetään suojalaitteiden kuvauksia esimerkiksi ICD-tiedoston sisällön perusteella.

– SCD (Substation Configuration Description) -tiedosto liittyy tiedon siirtämiseen järjestelmätyökalun ja suojalaitteen konfigurointityökalun välillä. Tiedostoon on mallinnettu suojalaitteiden sijainti sähköasemalla ja niiden välinen kommunikaatio.

– CID (Configured IED Description) -tiedosto sisältää täysin konfiguroidun suojalaitteen sisältäen kommunikaatioon liittyvät tiedot. Tiedostoa käytetään tiedonsiirrossa suojalaitteen konfigurointityökalun ja itse laitteen välillä.

Kuvasta 3.7 nähdään myös, että sähköasema-automaatiojärjestelmän kuvaava SCD-tiedosto koostuu suojalaitteiden tietoja kuvaavista ICD-tiedostoista ja sähköaseman määrittelyn kuvaavasta SCD-tiedostosta. CID-tiedostolla kuvataan puolestaan vain yksittäisen suojalaitteen konfiguraatio, eikä se sisällä tietoa lainkaan muista suojalaitteista.

Standardin toinen versio [14] ottaa enemmän kantaa konfigurointiprosessiin ja lisää kaksi uutta SCL-tiedostotyyppiä konfigurointiprosessin selkeyttämiseksi:

– IID (Instantiated IED Description) -tiedoston tarkoituksena on välittää tietoa suojalaitteen konfigurointityökalusta järjestelmätyökaluun. IID-tiedosto sisältää yhden suojalaitteen tiedot. Järjestelmätyökalu voi hyödyntää tietoja ylemmällä tasolla ja esimerkiksi päivittää suojalaitteen konfiguroituja arvoja.

– SED (System Exchange Description) -tiedosto on tarkoitettu tiedon välittämiseen kahden eri projektin välillä. Tiedosto on karsittu versio SCD-tiedostosta ja se määrittelee rajapinnan kahden erillisen projektin välillä.

Standardi määrää, että IEC 61850 -yhteensopivan suojalaitteen mukana tulee toimittaa suojalaiteen kuvaava CID-tiedosto. Vaihtoehtoisesti valmistaja voi toimittaa työkalun, jolla CID-tiedoston tuottaminen on mahdollista SCD-tiedoston perusteella. [13]

3.3.4 Asema- ja prosessiväylän kommunikaation mallintaminen

Asemaväylän GOOSE-kommunikaatio kuvataan kokonaisuudessaan SCL-tiedostossa. GOOSE-viestin sisältö kuvataan datanippuna ja kontrollilohkona (GOOSE Control Block).

SCL:n kommunikaatio-osuudessa määritellään jokaiselle kontrollilohkolle kommunikaatioon tarvittavat yksityiskohdat. Kommunikaation yhteydessä määritellään myös kohdassa 3.2.1 kuvatut lähetyssekvenssin minimi- ja maksimiarvot. [13]

Prosessiväylän mittadataan liittyvän kommunikaation mallintamiseen SCL-tiedostossa ei ole syntynyt yhtenäistä käytäntöä. IEC 61850-9-2 LE:n mukaan [8] mittadataan liittyy neljä virta- ja jännitearvoa. Kyseinen standardi määrittelee oman loogisen laitteen liittymisyksikölle, eli liikennettä lähettävälle osalle. Vastaanottavan osapuolen mallintamiseen ei ole puolestaan määritelty IEC 61850 -standardin mukaista objektimallia, eikä sen mallintamiseen ole vielä syntynyt yhtenäistä käytäntöä.

4 HIDRAW-AJOYMPÄRISTÖ

HiDraw on ABB:n kehittämä graafinen Windows-pohjainen ohjelmakoodin generointityökalu. Ohjelma rakennetaan graafisista symboleista, joita yhdistelemällä saavutetaan ohjelman looginen toiminta. HiDraw generoi ohjelmakoodia symbolien määrittelemällä tavalla, eikä generoidun koodin ohjelmointikielellä ole näin merkitystä. HiDraw käyttää ohjelman kääntämiseen Make-ohjelmaa, joka ei myöskään aseta rajoituksia käytettävälle kääntäjälle tai työkaluketjulle. [26]

Ohjelman graafinen esitysmuoto muistuttaa elektroniikassa käytettyjä kytkentäkaaviopiirrustuksia ja se toimii myös yksityiskohtaisena dokumentaationa ohjelman rakenteesta ja toiminnasta. Näin HiDraw-ohjelmia voidaan lukea ilman yksityiskohtaista tuntemusta käytetystä ohjelmointikielestä. [26]

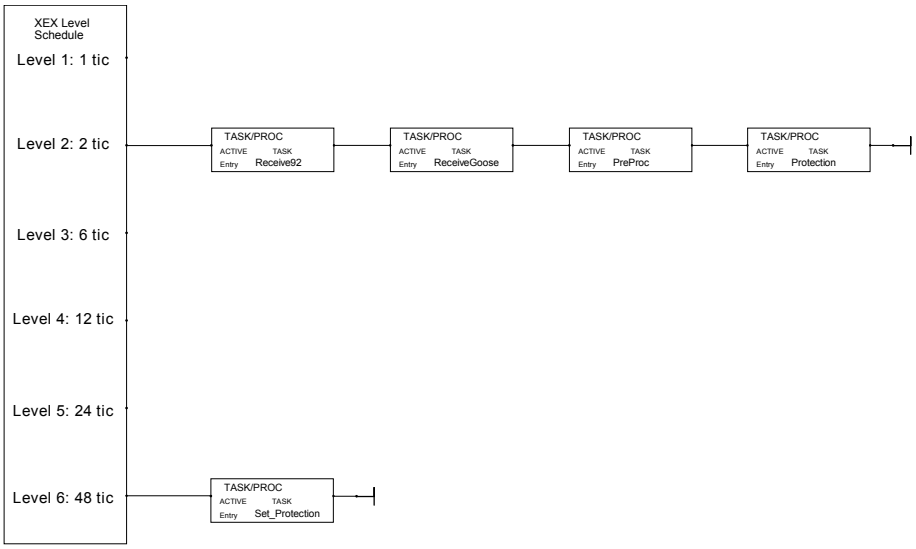
4.1 HiDraw-ohjelmat

HiDraw-ohjelma koostuu monesta sisäkkäisestä piirroksesta, jotka sisältävät ohjelmalogiikan toteuttavat symbolit. Erillinen pääpiirros on välttämätön osa ohjelmaa, sillä se määrittää ohjelman aloituskohdan. Pääpiirros sisältää myös muiden piirrosten suoritusjärjestetyksen määräämän vuorontajataulun. Esimerkki vuorontajataulusta on esitetty kuvassa 4.1. Vuorontajataulu koostuu eri tasoista, jotka määräävät piirrosten suoritusajan. Vuorontajan toiminta muistuttaa RMS (Rate Monotonic Scheduling) -vuoronnusalgoritmin toimintaa. Vuorontaja suorittaa yhteen tasoon littettyjä piirroksia määrätyn aikaviipaleen, jonka jälkeen se suorittaa seuraavaan tasoon liitettyjä piirroksia määrätyn ajan. [26]

HiDraw'n ensisijainen ulostulo on ohjelmakoodi, joka generoidaan piirrosten sisältämien symbolien avulla. Jokaiseen symboliin liittyy kaksi erillistä komponenttia: piirrosmerkki ja koodimalli. Näiden kahden komponentin avulla HiDraw generoi ohjelmakoodin. [26]

4.1.1 Piirrosmerkit

Piirrosmerkit muodostavat HiDraw-piirroksen graafisen osan. Piirrosmerkki sisältää tiedon graafisen symbolin syötteiden, parametrien ja ulostulojen tietotyypeistä. Tyypit sidotaan käytettävän ohjelmointikielen tyyppeihin ja piirrosmerkkiin on mahdollista lisätä tyyppitarkastuksia oikean toiminnan takaamiseksi. HiDraw sisältää yksinkertaiset piirrostyökalut merkkien luomiseksi. [26]



Kuva 4.1. *Pääpiirroksen sisältämä vuorontajataulu.*

Piirrosmerkit kytketään toisiinsa käyttämällä *signaaleja*. Signaalit muodostavat kytkennän piirrosmerkin ulostulosta toisen merkin syötteeseen. Ohjelmakoodin tasolla jokainen syöte ja ulostulo vastaa yhtä ohjelman muuttujaa, jolla on tietty tietotyyppi. Hyödyntämällä syötteen ja ulostulon tyypitystä, voidaan välttyä laittomilta kytkennöiltä. Signaali edustaa näin ollen sijoitusoperaatiota ulostulon edustamasta muuttujasta syötteen edustamaan muuttujaan. [26]

Piirrosmerkin ei ole välttämätöntä sisältää toiminnallisuutta, eikä sitä ole välttämätöntä kytkeä mihinkään. Tällainen piirrosmerkki voi dokumentoida esimerkiksi käytettävän kääntäjän version. Tällöin dokumentaatioon käytetty merkki ei tuota toiminnallista ohjelmakoodia lainkaan. Jos piirrosmerkin halutaan tuottavan ohjelmakoodia, tulee siihen liittää kohdassa 4.1.2 kuvattu koodimalli. [26]

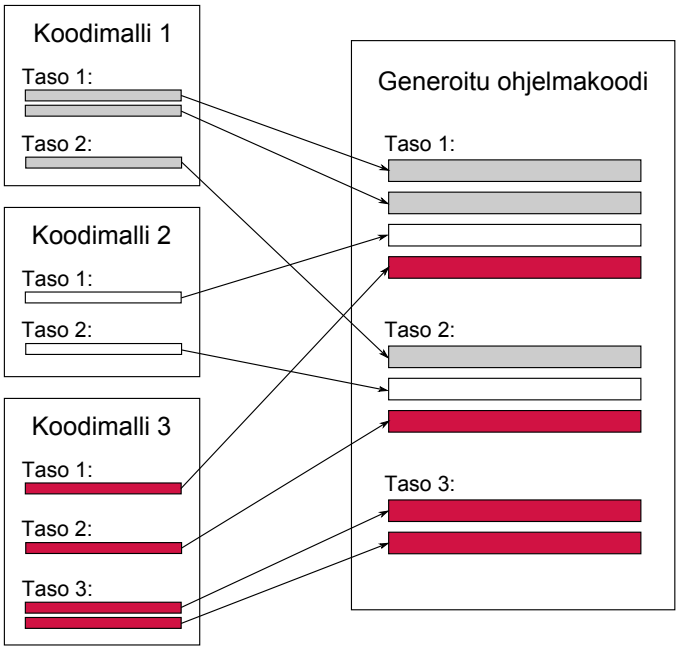
4.1.2 Koodimallit

Koodimalli on tekstitiedosto, joka tiedostopääte on HDF (HiDraw Definition File). HDF-tiedosto koostuu pakollisista identifiointitiedoista ja koodin generointiin liittyvistä ohjauskomennoista. [26]

Koodimallit käyttävät HiDraw'n kontrollirakenteita koodin generoimiseen. HiDraw tukee toisto- ja ehtorakenteita, joita käyttämällä voidaan käydä läpi kaikki symbolin syötteet, parametrit ja ulostulot. HiDraw tukee myös tiedostovirtoja, joiden avulla voidaan kirjoittaa tietoa symbolin tietosisällöstä erilliseen tekstitiedostoon. Myös varsinainen ohjelmakoodin generointi toteutetaan *code*-nimisen tiedostovirran avulla. HiDraw'n tiedostovirrat toimivat samalla periaatteella kuin C++-ohjelmointikielen vastaava ominaisuus. [26]

Kaikki *code*-virtaan syötetty teksti näkyy käännösyksikön sisällössä. Virtaan syötetty teksti tulee olla määrämuotoista, syntaksiltaan valitun ohjelmointikielen mukaista ohjelmakoodia. Ohjelmakoodia syötetään *code*-virtaan generointitasojen määrämässä jär-

jestyksessä. Generointitasot jakavat ohjelmakoodin erillisiin tasoihin, joiden sisällöstä ohjelmakoodi muodostuu kerroksittain. Kuvassa 4.2 on havainnollistettu generointitasojen tarkoitusta.



Kuva 4.2. *Generointitasojen toiminta koodin generoinnissa. Käännösyksikön sisältö kootaan eri koodimalleista generointitasojen perusteella.*

Kuvassa on merkitty eri koodimallien tuottamaa ohjelmakoodia eri väreillä. Kuvasta nähdään myös, ettei kaikkien koodimallien tarvitse tuottaa ohjelmakoodia jokaisella generointitasolla.

4.1.3 Piirrostyypit ja XEX-vuorontaja

HiDraw sisältää erityyppisiä piirroksia, joilla kaikilla on oma tarkoituksensa. Vuorontajataulun sisältämän piirroksen tyyppi on *XEX* ja kaikki muut piirrokset liitetään suoraan vuorontajatauluun. [26]

Taulukoon 4.1 on koottu HiDraw'n eri piirrostyypit. *Task*-tyyppinen piirros liitetään suoraan vuorontajatauluun halutulle tasolle. Vuorontajataulun taso määrää sen, kuinka usein tiettyä *Task*-piirrosta suoritetaan. Usein halutaan, että suojausohjelmaa suoritetaan mahdollisimman usein ja vähemmän tärkeitä osia, kuten parametrien asettelua suoritetaan harvemmin. [26]

Task-piirros voi sisältää alirutiineja eli *Subtask*-tyyppisiä piirroksia. Alirutiineille annetaan eksplisiittinen suoritusjärjestys, jolloin alirutiinit suoritetaan annetussa järjestyksessä. Suoritusjärjestestys määrätään käännösvaiheessa staattisesti, joten sen vaihtaminen ei ole mahdollista ilman ohjelman uudelleenkääntämistä. Liitteessä 1 on esimerkki yksinkertaisesta HiDraw-piirroksesta, jota suoritetaan alirutiinina. [26]

Taulukko 4.1. *HiDraw'n eri piirrostyypit ja niiden tarkoitukset.*

Piirrostyyppi	Tarkoitus
Task	Pienin yksikkö, jonka suoritusta vuorontaja valvoo.
Subtask	Yhteen Task-piirrokseen liittyvä alirutiini, joista Task-piirros koostuu.
XEX	Sisältää sovelluskohtaisen vuorontajataulun, joka määrittää eri rutiinien ajojärjestyksen.
Main	Sisältää ylätason vuorontajataulun eri sovelluksille.

Jokainen piirros jaetaan lisäksi kahteen eri funktioon: alustus- ja suoritusfunktioon. Alustusfunktio suoritetaan ainoastaan kerran ohjelman käynnistyksen yhteydessä. Suoritusfunktiota suoritetaan puolestaan vuorontajataulun määrämän syklin mukaisesti. [26]

4.1.4 Ohjelmabinaarin tuottaminen

Koodin generointi suoritetaan piirros kerrallaan ja jokainen piirros edustaa yksittäistä käännösyksikköä. Kun kaikki piirrokset on käsitelty, on ohjelmakoodi valmis käännöstä varten. [3]

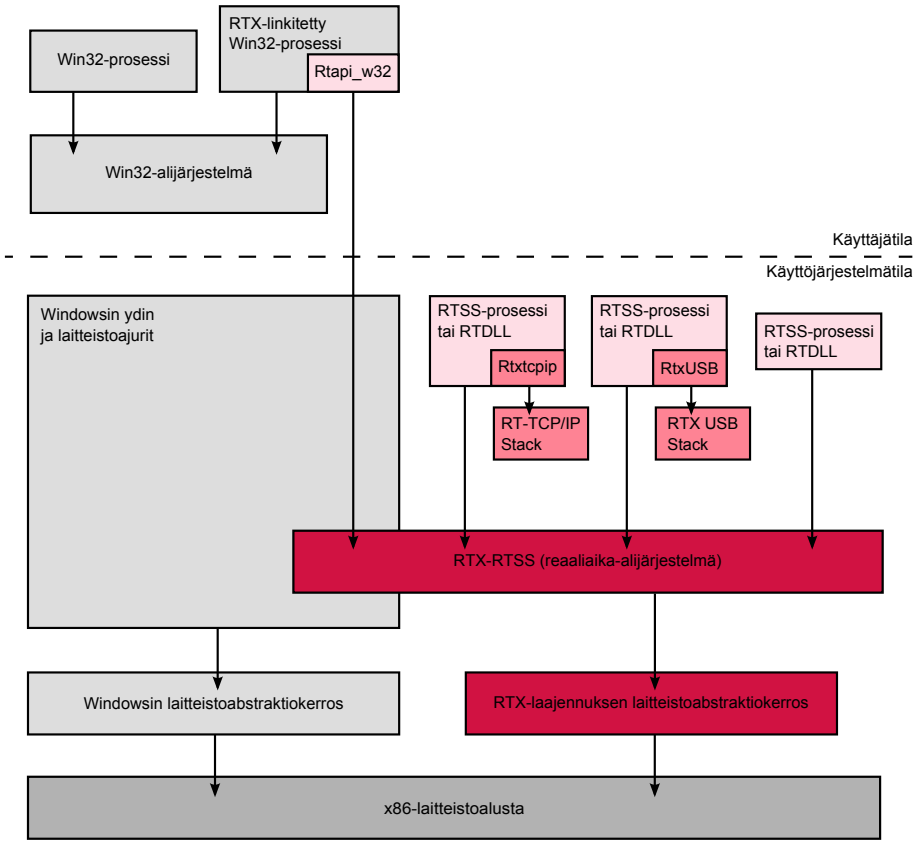
HiDraw ajaa Make-ohjelman, joka kääntää varsinaisen ohjelmakoodin kirjasto-tiedostoksi. Make-ohjelmaa ohjaavan Makefile-tiedoston sisältö on myös HiDraw-ohjelmoijan generoitavissa, joten ohjelmoija voi vapaasti valita käytettävän kääntäjäversion. Lopuksi käännetty kirjasto linkitetään vuorontajasovelluksen kanssa valmiiksi binaariksi. [3]

4.2 RTX-reaaliaikalaajennus

HiDraw-ohjelmaa suoritetaan Windows-käyttöjärjestelmän rinnalla toimivassa reaaliaikalaajennuksessa. Reaaliaikalaajennuksena käytetään IntervalZeron valmistamaa RTX (Real Time eXtensions for Windows) -laajennusta. Laajennus toimii itsenäisenä osajärjestelmänä, eikä se ole riippuvainen Windows-käyttöjärjestelmän palveluista. Moniprosessorijärjestelmissä RTX-laajennuksen on mahdollista ottaa käyttöön yksi tai useampia prosessoriytimiä, jolloin Windows ei tiedä muista olemassa olevista ytimistä mitään. Järjestelmän tulee kuitenkin jättää yksi prosessoriydin Windowsin käyttöön. Tämä mahdollistaa sen, että RTX-laajennus voi käyttää omistamiaan prosessoriytimiä omaan käyttöönsä ja jäljelle jäänyt ydin suorittaa Windowsin palveluita. RTX mahdollistaa myös reaaliaikasovelluksen suorittamisen samalla suorittimella Windowsin kanssa. Tällöin suoritettavan reaaliaikasovelluksen tulee olla vähän suoritinaikaa vievä, etteivät Windowsin palvelut kärsi reaaliaikasovelluksen suorittamisesta. [20]

4.2.1 Reaaliaikalaajennuksen arkkitehtuuri

Reaaliaikalaajennus on toteutettu käyttöjärjestelmätilassa suoritettavana ajuriprosessina. RTX toteuttaa oman laitteistoabstraktiokerroksen, jolla saavutetaan keskeytysten eristäminen Windowsin ja reaaliaikalaajennuksen välillä. Näin reaaliaikalaajennus voi siepata laitteistokeskeytykset ennen Windowsia ja käyttää haluamiaan laitteita omiin tarkoituksiinsa. Laitteistoabstraktiokerroksen toteuttaminen mahdollistaa myös tarkkojen kellojen toteuttamisen reaaliaikasovellusten käyttöön. Kuvassa 4.3 on esitetty RTX-ympäristön arkkitehtuuri. [20]



Kuva 4.3. *RTX-laajennuksen arkkitehtuuri. [20]*

Reaaliaikalaajennuksen on mahdollista siepata Windowsin sammutuskäsky ja reagoida siihen lopettamalla reaaliaikasovellukset hallitusti. Mikäli Windows-käyttöjärjestelmä kaatuu ohjelmistovian takia, on reaaliaikasovelluksilla mahdollisuus reagoida tilanteeseen esimerkiksi sammuttamalla prosessit ennen järjestelmän uudelleenkäynnistymistä. Reaaliaikasovellukset voivat myös jatkaa toimintaa vaikka Windows olisi kaatunut. [20]

4.2.2 Kommunikointi Windows-ympäristön kanssa

RTX-laajennus tarjoaa monipuolisen ohjelmointirajapinnan Windows-käyttöjärjestelmän kanssa kommunikointiin. Laajennus tarjoaa jaetun muistin ympäristöjen välille, sekä erilaisia synkronointiin liittyviä mekanismeja, kuten poissulkemismekanismit ja tapahtumailmoi-

tukset. Poisulkemismekanismit mahdollistavat Windows-prosessin ja reaaliaikaprosessin täydellisen poissulkemisen. [5]

Reaaliaikalaajennuksen tarjoama ohjelmointirajapinta muistuttaa suurelta osin Windowsin tarjoamaa ohjelmointirajapintaa. Joiltain osin laajennos käyttää suoraan Windowsin tarjoamaa palvelua, joten ohjelmointimalli on Windows-ohjelmoijalle tuttu jo entuudestaan. Niiltä osin kun laajennus ei voi käyttää suoraan Windowsin ohjelmointirajapintaa, toteuttaa se oman palvelun, joka on yleensä parametreiltaan identtinen vastaavan Windows-palvelun kanssa. [5]

Jaettu muisti allokoidaan Windowsin muistialueelta käyttämällä Windowsin tarjoamaa palvelua, joka ei ole deterministinen. RTX-laajennus varaa muistialueen (memory pool) Windowsin muistista, jonka jälkeen siihen viittaaminen reaaliaikasovelluksesta on determinististä. IntervalZero suosittelee riittävän muistialueen varaamista etukäteen, ennen reaaliaikasovelluksen käynnistämistä. Tämä menettelytapa vähentää tarvetta täydentäville muistivarauksille Windowsin muistialueelta ja takaa reaaliaikasovelluksen deterministisen suorittamisen. [5]

4.3 Ajoympäristön konfigurointi

Asematietokoneen ohjelmisto on suurelta toteutettu käyttäen HiDraw-työkalua ja uusien vaatimusten mukaisesti sitä pitäisi pystyä konfiguroimaan eri tavoin.

4.3.1 Kytkentöjen konfigurointi

Uusien vaatimusten mukaisesti piirrosmerkkien välisien signaalien kytkentöjä tulee pystyä muokkaamaan. Kytkentöjen muuttaminen on tärkeä osa konfigurointiprosessia ja niitä muokkaamalla voidaan samoilla ohjelmistokomponenteilla toteuttaa erilaisia kokonaisuuksia eri tarpeisiin.

HiDraw-työkalu tekee kytkennöistä staattisia jo käännösaikaisesti, jolloin kytkentöjen muuttaminen vaatii aina ohjelman uudelleenkääntämisen. Ohjelman uudelleenkääntäminen on liian raskas ja aikaavievä operaatio, eikä sellaista voisi jättää loppuasiakkaan tehtäväksi. Vaihtoehtoisesti asemalle voitaisiin asentaa monta eri versiota suojaussovelluksesta ja jokainen suojaussovellus sisältäisi erilaisen konfiguraation. Tämä lähestymistapa lisäisi ylläpitokustannuksia merkittävästi, eikä sillä saavutettaisi täyttä konfiguroitavuutta. Tarve kytkentöjen konfigurointiin on siis ilmeinen.

4.3.2 Asettelujen konfigurointi

Jokaiseen suojausfunktioo liittyy tietty määrä parametreja, joita muuttamalla lohkon toimintaa voidaan hienosäätää. Tähän saakka parametrit on määrätty staattisesti käännösvaiheessa, eikä niitä ole ollut mahdollista muuttaa jälkikäteen.

Lisäksi asetteluja tulee olla mahdollista muuttaa myös ajon aikana, jolloin hienosäätö on mahdollista ilman järjestelmän uudelleenkäynnistämistä. Perinteisessä HiDraw-ohjelmassa kaikki parametrit on kiinnitetty käännösaikana, ja niiden muuttaminen vaatii kytkentöjen tapaan koko sovelluksen uudelleenkääntämisen.

5 PCM600 KONFIGUROINTITYÖKALU

PCM600 on ABB:n kehittämä työkalu, jota käytetään suojareleiden konfiguroimiseen. Perinteisesti jokaista suojalaitetta varten on kehitetty oma konfigurointityökalu, mutta PCM600 mahdollistaa useiden suojalaitteiden konfiguroimisen yhtenäisellä tavalla samalla työkalulla. Työkalua käyttämällä on myös mahdollista seurata releen toimintaa ja kerätä esimerkiksi häiriötallenteita suojalaitteesta. Suojalaite voidaan konfiguroida etänä käyttämällä laitteen Ethernet-yhteyttä tai paikallisesti kytkemällä tietokone suoraan konfiguroitavaan laitteeseen. [4]

PCM600 rakentuu erilaisista työkalumooduleista, joilla kaikilla on oma tarkoituksensa suojalaitteen konfigurointiprosessissa. Työkalumoduulien lisäksi PCM600 tarvitsee vähintään yhden Connectivity Package (ConnPack) -laajennuksen, jota käytetään suojalaitteen ja työkalun välisessä kommunikaatiossa. ConnPack-laajennus abstrahoi suojalaitteiden eroavaisuudet ja välittää työkalumoduuleille niiden tarvitseman suojalaitekohtaisen datan. [4]

Luvun sisältö perustuu yleiskuvauksen osalta lähteeseen [4]. Arkkitehtuurin kuvaus perustuu lähteen [1] sisältöön.

5.1 Työkalut

Työkalut ovat kiinteä osa PCM600-työkalun toiminnallisuutta. Eri työkaluja käyttämällä suojalaitteen eri osia voidaan konfiguroida eri näkökulmista. Työkalut esittävät suojalaitteiden ominaisuudet, joita työkalun käyttäjä voi muokata haluamallaan tavalla. Työkalumoduulit saavat tiedon suojalaitteiden ominaisuuksista suoraan ConnPack-laajennukselta. ConnPack-laajennusten rooliin PCM600-työkalun arkkitehtuurissa palataan tarkemmin kohdassa 5.2.3.

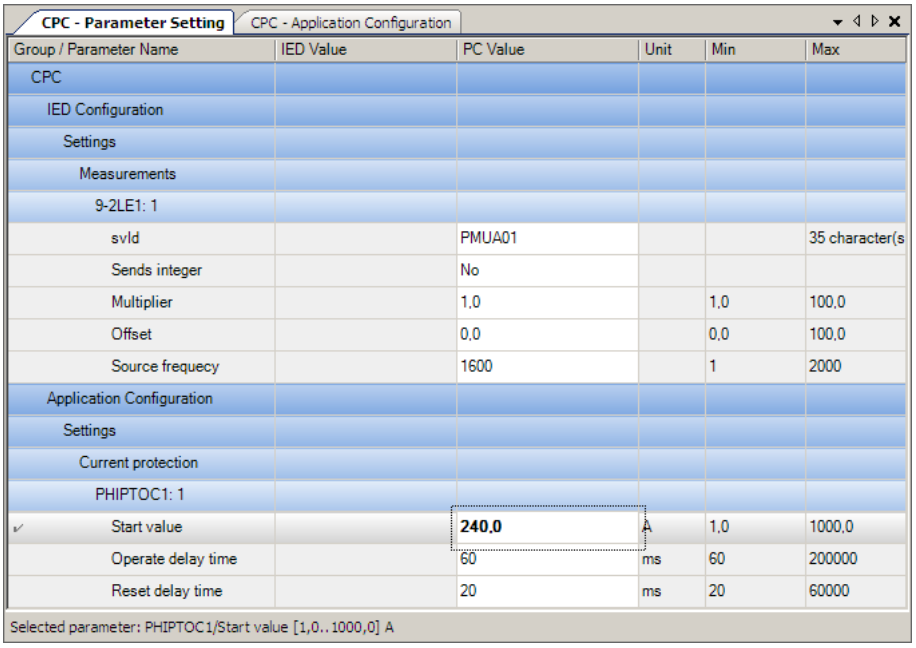
5.1.1 Konfigurointivelho

Konfigurointivelho on ConnPack-laajennuksen toteuttama komponentti, jonka tarkoitus on auttaa käyttäjää suojalaitteen konfiguroinnin aloittamisessa. Velhon avulla käyttäjä valitsee konfigurointiin käytettävän protokollan ja konfigurointimenetelmän. Suojalaitetta voidaan konfiguroida online-tilassa, jolloin konfigurointityökalu tunnistaa suojalaitteen tyypin ja mallin automaattisesti. Offline-konfiguroinnissa käyttäjän täytyy syöttää suojalaitteen tiedot manuaalisesti konfigurintivelhoon.

Kaupallisessa suojalaitteessa konfigurointivelho sisältää myös tilauskoodin syöttämisen, joka määrittää suojalaitteen tuotekohtaisen toiminnallisuuden. Velhon sisältö on erilainen riippuen valitusta konfigurointimenetelmästä.

5.1.2 Parametrien asettelutyökalu

Parametrien asettelutyökalussa (Parameter Setting Tool, PST) on mahdollista konfiguroida suojalaitteen asetteluja. Asettelut vaikuttavat suojalaitteen toimintaa ja osaa asetteluista voidaan muokata myös silloin, kun suojalaite on toiminnassa. Ylivirtasuojan tapauksessa tällaisia asetteluja ovat esimeriksi toiminta-aika ja -arvo. Kuvassa 5.1 on esitetty kuvakaappaus parametrien asettelutyökalusta.



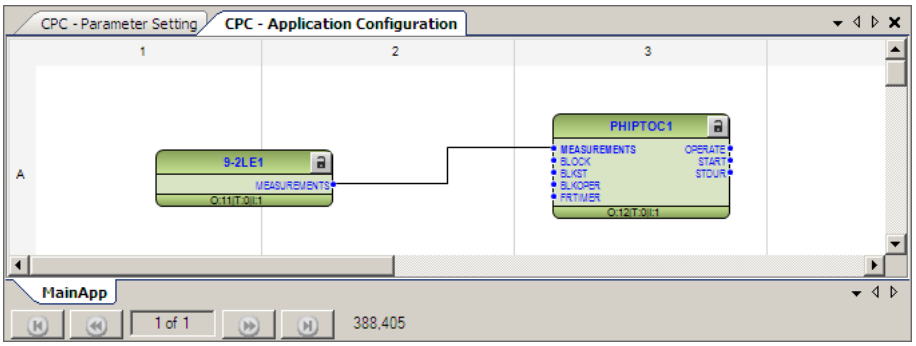
Kuva 5.1. *Parametrien asettelutyökalu.*

Parametrien asettelutyökalu ohjaa käyttäjää parametrien asettelussa tarkastamalla parametrien ääriarvot automaattisesti. Näin käyttäjää voidaan estää tekemästä järjettömiä tai vaarallisia asetteluita.

5.1.3 Suojausohjelman konfigurointityökalu

Sovelluksen konfigurointityökalu (Application Configuration Tool, ACT) mahdollistaa suojausohjelman eri toimilohkojen välisten kytkentöjen konfiguroimisen. Kytkentöjä konfiguroimalla on mahdollista rakentaa eri tavalla toimivia suojausohjelmia. Kuvassa 5.2 on esitetty yksinkertainen esimerkki ACT:n käytöstä.

Sovelluksen konfigurointityökalu mahdollistaa myös signaalien monitoroinnin reaaliaikaisesti. Reaaliaikainen monitorointi on mahdollista vain silloin, kun suojalaiteeseen on muodostettu yhteys.



Kuva 5.2. *Suojausohjelman konfigurointityökalu ja yksinkertainen kytkentä kahden eri toimilohkon välillä.*

5.1.4 Konfiguraation luku- ja kirjoitustyökalu

Kun suojareleen suojausohjelma on suunniteltu ja parameterit aseteltu, tulee konfiguraatio siirtää suojalaitteeseen. Tästä toiminnallisuudesta vastaa konfiguraation luku- ja kirjoitustyökalu (Common Read and Write, CRW).

Konfiguraation lisäksi suojalaitteeseen kirjoitetaan työkalukohtaista dataa, kuten sovelluksen konfigurointityökalun tuottama graafinen piirros ja siihen liittyvät kytkennät. Työkalutietojen tallentaminen suojalaitteeseen mahdollistaa nykyisen konfiguraation palauttamisen suojalaitteesta työkaluun.

5.1.5 Kommunikaatioprotokollat ja -formaatit

ConnPack-laajennuksen on mahdollista käyttää PCM600-työkalun tarjoamia kommunikaatiprotokollia. PCM600 tarjoaa mahdollisuuden käyttää IEC 61850 -standardin mukaista kommunikaatiota tai ConnPack voi halutessaan käyttää vanhempia ABB:n kehittämiä protokollia, kuten esimerkiksi SPA-protokollaa. Kommunikaatioprotokollaa käytetään online-konfigurointiin ja sitä käyttämällä on mahdollista lukea suojalaitteen tietoja reaaliajassa.

Kommunikaatioformaatilla tarkoitetaan puolestaa sitä formaattia, jota käytetään kun laitetta ei ole mahdollista konfiguroida online-tilassa. PCM600 jättää tämän valinnan ConnPack-laajennuksen vastuulle, joten käytettyjen formaattien kirjo on melko laaja.

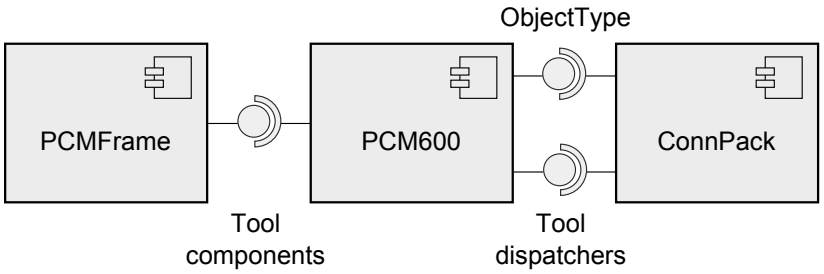
5.2 PCM600-työkalun arkkitehtuuri

PCM600 on komponenttikehyksiin pohjautuva ohjelmistokehys, joka koostuu kolmesta eri pääkomponentista: *PCM-kehyksestä*, *työkalumoduuleista* ja *ConnPack-laajennuksista*. PCM600 on kehitetty käyttämällä Microsoftin .NET-sovelluskehystä.

PCM600 on kehyslajiltaan abstraktin ja plug-in-kehyksen yhdistelmä. Kehyksen työkalumoduulit tarjoavat abstraktin kehyksen tapaan toteutettavan rajapinnan ConnPack-

laajennuksille ilman valmista kantaluokkaa tai yleiskäyttöistä toteutusta. Lisäksi työkalumoduulit tarjoavat ConnPack-laajennuksille oman erikoistamisrajapinnan. [21]

Plug-in-arkkitehtuurin piirteitä on nähtävissä tavassa, jolla PCM600 käsittelee ConnPack-laajennuksia: laajennus ladataan dynaamisesti käyttöön sovitusta kansiosta ja kaikki työkalumoduulien tarvitsemat instanssit ladataan sovitun rajapinnan välityksellä. Korkean tason arkkitehtuuri on esitetty kuvasssa 5.3. [21]



Kuva 5.3. *PCM600-työkalun arkkitehtuuri.*

5.2.1 PCM-kehys

PCM-kehys (PCM Frame) on ohjelmistokehys, jonka päälle PCM600 on rakennettu. PCM-kehys tarjoaa PCM600:n ydintoiminnallisuuden ja antaa siihen pohjautuville sovelluksille yhtenäisen ulkoasun. Ydin tarjoaa myös palvelut eri tietokantojen käyttämiseen ja tarjoaa rajapinnat uusien toiminnallisuuksien toteuttamiseen.

PCM600 hyödyntää tätä laajennusmahdollisuutta ja tarjoaa sovelluskehittäjille mahdollisuuden luoda uusia työkalumoduuleja ja ConnPack-laajennuksia käytettäväksi. Molemmat komponentit käyttävät joko suorasti tai epäsuorasti PCM-kehyksen ytimen rajapintoja eri tarkoituksiin.

5.2.2 Työkalumoduulit

Työkalumoduulit toteuttavat PCM600-työkalun eri toiminnallisuudet ja erottavat eri PCM-kehyksen päälle rakennetut ohjelmat toisistaan. PCM600 jaottelee työkalumoduulit kolmeen eri katerogiaan: järjestelmä-, palvelu- ja objektityökaluihin.

Järjestelmätyökalumoduulit manipuloivat järjestelmätason tietoja, kuten konfigurointiprojektiin liittyviä tietoja. Työkalumoduulien tarvitsemat tietosisällöt on esimerkiksi toteutettu järjestelmätason työkaluina. Palvelutyökalumoduuleja suoritetaan taustalla ilman näkyvää käyttöliittymää ja niiden tarkoitus on erottaa käyttöliittymä varsinaisesta ohjelmalogiikasta.

Objektityökalut käsittelevät aina yhteen tiettyyn objektiin liittyvää dataa. Objekti on pienin yksittäinen komponentti PCM600:n hierarkiassa ja se voi olla esimerkiksi yksittäistä suojalaitetta edustavat osa. Esimerkiksi kohdassa 5.1 esitellyt työkalut on toteutettu objektityökaluina.

5.2.3 ConnPack-laajennukset

ConnPack-laajennus on korkeimman abstraktiotason komponentti ja se käyttää PCM-kehyksen ja työkalumoduulien palveluja hyödykseen. ConnPack-laajennus sisältää tietyn suojalaitteen tai suojalaiteperheen toiminnallisuuden ja mahdollistaa eri suojalaitteiden konfiguroimisen PCM600-työkalulla.

PCM600:n arkkitehtuuri mahdollistaa myös ConnPack-laajennuksien käyttämisen järjestelmätasolla eri kommunikaatioprotokollien abstrahoimiseen. Tällaisia järjestelmätason laajennuksia ovat esimerikiksi IEC 61850- ja SPA-protokollat toteuttavat ConnPack-laajennukset. Kommunikaatioyhteydet toteuttavat laajennukset eivät näy käyttäjälle lainkaan, vaan eri työkalumoduulit käyttävät niitä sisäisesti kommunikaatioyhteyksien toteuttamiseen.

Suojarelekohtaiset ConnPack-laajennukset toteuttavat työkalumoduulien tarjoaman rajapinnan. Laajennus voi myös vapaasti päättää, mitkä työkalumoduulit se toteuttaa. Eri työkalumoduulit ovat myös täysin riippumattomia toisistaan, joten niitä voidaan kehittää erillään ja käyttää uudelleen tarvittaessa.

5.3 ConnPack-laajennuksen arkkitehtuuri

ConnPack-laajennuksen arkkitehtuuri määräytyy sen toteuttamien työkalumoduulien perusteella. ConnPack-laajennus täydentää työkalumoduulissa auki jätetyt asiat suojalaiteriippuvaisella tavalla.

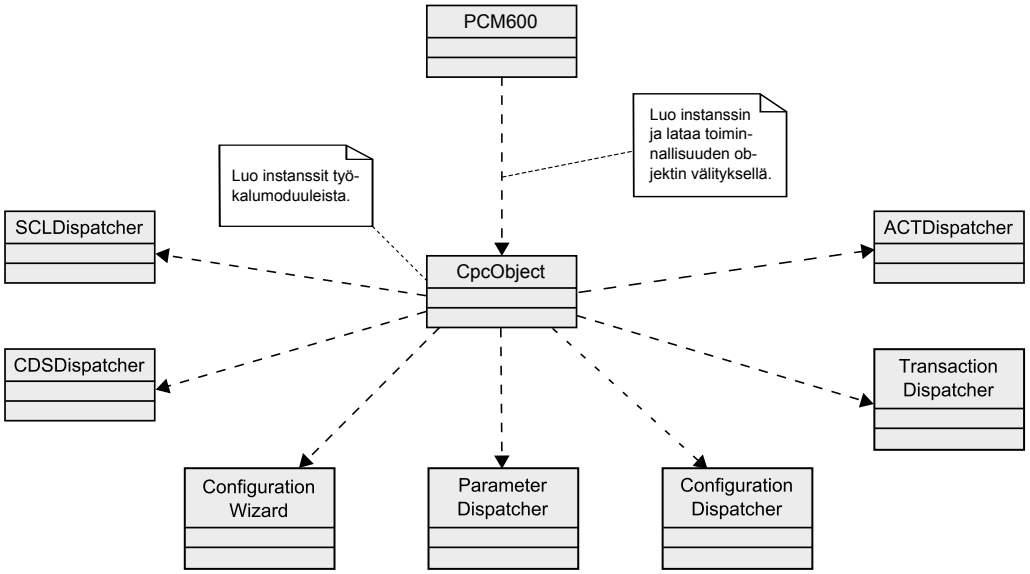
PCM600 lataa ConnPack-laajennuksen käyttöön *objektityypin* välityksellä ja sitä kautta ladataan käyttöön myös eri työkalumoduulit. Laajennus toteuttaa PCM600:n vaatiman datamallin, *tyyppidatan*, jota PCM-kehys käsittelee instanssidatan välityksellä. Tyyppidatassa kuvataan suojalaitteen sisäinen rakenne, kuten eri suojaustoiminnot ja suojalaitteen laitteistoon liittyvät yksityiskohdat.

5.3.1 Objektityyppi

ConnPack-laajennuksen ytimen muodostaa objektityyppi, jonka kautta kaikki toiminnallisuus ladataan käyttöön. Objektityypin tulee toteuttaa *ObjectType*-rajapinta, jonka lisäksi se voi toteuttaa haluamansa määrän työkalumoduuleita koskevia rajapintoja. Rajapintojen avulla PCM600 tietää, mitä ominaisuuksia ConnPack toteuttaa ja osaa näin näyttää oikeat työkalumoduulit PCM600-työkalussa. [1]

PCM600:n arkkitehtuurin näkökulmasta objektityyppi edustaa *välittäjä*-komponenttia. Komponentti välittää PCM600:lle instanssit eri työkalumoduulit toteuttavista luokista. PCM600-ohjelmistokehys ei näin ole riippuvainen muista komponenteista kuin objektityypistä. Komponentti sisältää myös käyttöliittymään välitettäviä tietoja suojalaitteesta, kuten tunnistetiedot ja käytettävissä olevat työkalumoduulit. Kuvassa 5.4 on esitetty esimerk-

ki tilanteesta, jossa *CpcObject* toimii objektityyppinä ja kaikki toiminnallisuus ladataan käyttöön sen kautta. [21]



Kuva 5.4. *PCM600 lataa kaiken toiminnallisuuden käyttöön objektityypin välityksellä.*

Arkkitehtuuri mahdollistaa myös työkalumoduuleiden toiminnallisuuden päivittämisen tarjoamalla toteutettavista rajapinnoista useita eri versioita. Näin useiden työkalumoduulien toimintaa on päivitetty esimerkiksi vastaamaan IEC 61850 -standardin tuomia vaatimuksia. Objektityyppi-komponentti välittää myös tiedon toteutetun rajapinnan versiosta, jotta PCM600 osaa ladata käyttöön oikean version työkalumoduulista.

5.3.2 Tyyppidata ja instanssidata

ConnPack-laajennus mallintaa suojalaitteen toiminnallisuuden PCM600-kehyksen tarjoamia rajapintoja käyttäen. ConnPack-laajennuksen tarjoamaa datamallia kutsutaan *tyyppidataksi*. Tyyppidata mallintaa suojausohjelman konfiguroitavat parametrit ja kytkennät, sekä ohjelman toimilohkot. Toimilohko on suojalaitteen toiminnallinen osa, eli se vastaa jonkin itsenäisen suojaustoiminnon toteuttamisesta. Esimerkiksi kolmivaiheinen ylivirtasuoja on yksittäinen toimilohko. Toimilohkojen lisäksi tyyppidata sisältää ohjelman toiminnallisuuden rakentamiseen tarkoitettuja loogisia lohkoja, kuten AND- ja OR-lohkoja.

Tyyppidataan voi kuulua myös tietoja laitteen fyysisestä rakenteesta, kuten erillisten korttipaikkojen lukumäärä ja niiden tyyppitietoa. Asematietokonen tapauksessa fyysisiä ominaisuuksia ei mallinneta lainkaan, sillä siihen ei ole mahdollista liittää samanlaisia lisäkortteja kuten tavanomaisiin suojareleisiin.

Tyyppidata on puhtaasti datamalli, eikä se näy konkreettisesti PCM600-konfigurointityökalussa. Tyyppidatan datamallin perusteella ConnPack-laajennus luo *instanssidataa*, joka näkyy työkalussa konkreettisena objektina. Esimerkiksi käyttäjän muokkaamat toimilohkot esitetään ACT-työkalussa instanssidatana.

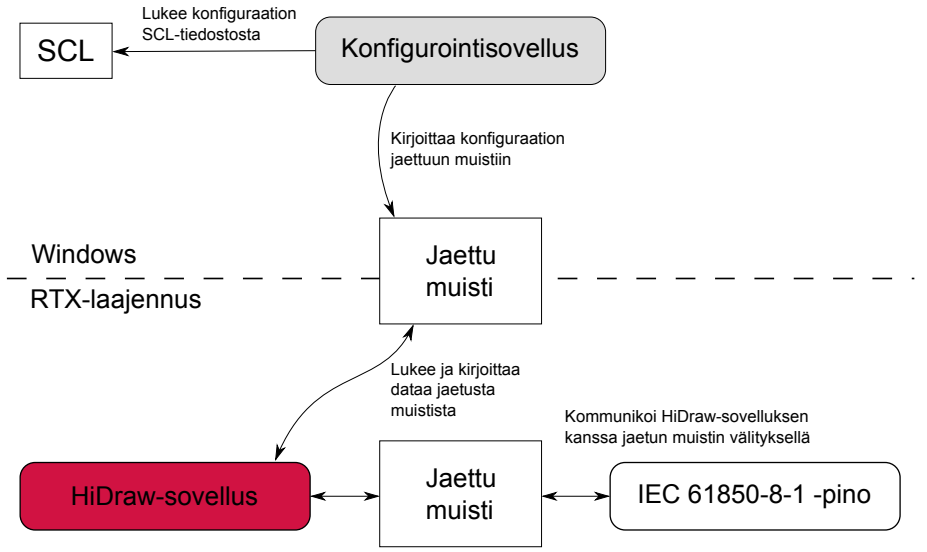
6 TOTEUTUS

Opinnäytetyössä kehitetty konfigurointijärjestelmä koostuu kahdesta eri osakomponentista: ajoympäristöstä ja konfigurointityökalusta. Komponentit sijaitsevat fyysisesti eri laitteissa ja tiedonsiirto niiden välillä tapahtuu SCL-tiedoston välityksellä. Konfigurointityökalu siirtää konfiguraatiotiedot sisältävän SCL-tiedoston asematietokoneelle FTP (File Transfer Protocol) -protokollaa käyttämällä. Asematietokoneen ajoympäristö lukee konfiguraation SCL-tiedostosta ja tekee tarvittavat toimenpiteet uuden konfiguraation käyttöönottamiseksi.

Karkeasti yksinkertaistettuna konfigurointityökalu voidaan nähdä ainoastaan SCL-tiedoston generointityökaluna. Samalla tiedostolla ohjataan ajoympäristön toimintaa. Ajoympäristö konfiguroidaan uudelleen aina, kun uusi SCL-tiedosto siirretään asematietokoneelle.

6.1 Ajoympäristön konfigurointi

Ajoympäristö koostuu kahdesta erillisesta osasta: konfigurointisovelluksesta ja HiDraw-työkalulla toteutetusta suojaussovelluksesta. Konfigurointisovellus on Windows-sovellus, joka muuntaa SCL-tiedoston sisältämän konfiguraation suojaussovelluksen ymmärtämään muotoon. Suojaussovellus toimii kokonaisuudessaan RTX-reaaliaikaympäristön alaisuudessa ja se lukee konfiguraation jaetun muistin välityksellä. Kuvassa 6.1 on havainnollistettu ajoympäristön rakennetta ja siihen liittyviä osakomponentteja.



Kuva 6.1. *Ajoympäristö ja sen osakomponentit.*

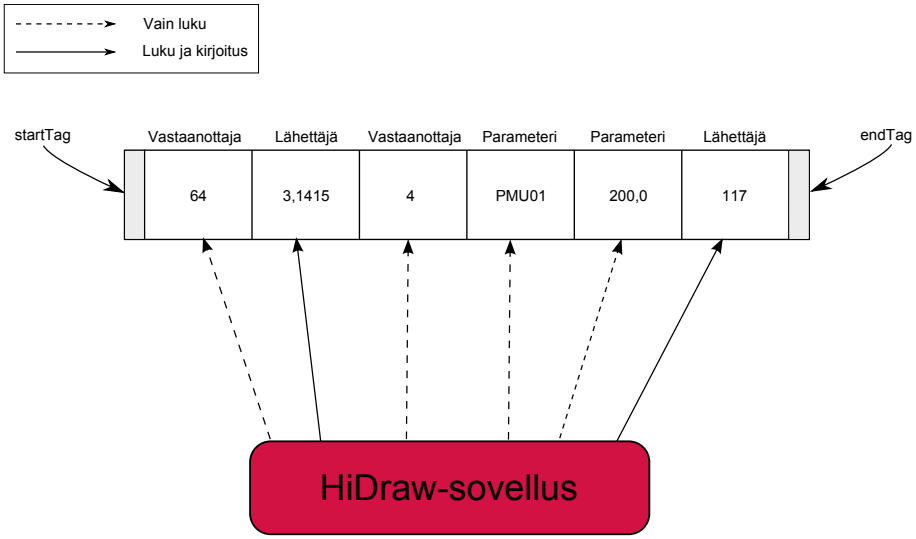
Ajoympäristön tuottamiseen käytetty HiDraw-työkalu asetti suurimmat haasteet konfiguroinnin toteuttamiselle. HiDraw generoi suojaussovelluksen ohjelmakoodin ja kääntää sen binaariksi. Ajoympäristöön liitetään mukaan myös vuorontajasovellus ja muut siihen liittyvät apusovellukset. HiDraw kääntää samalla myös erillisen binaarin konfigurointisovelluksesta. HiDraw-työkalun toimintaperiaatteesta johtuen konfiguroitavat asiat tuli ottaa huomioon jo käännösvaiheessa. Näin ollen konfiguroitavuus jouduttiin toteuttamaan suurimmaksi osaksi esikääntäjän direktiivejä hyödyntämällä.

6.1.1 Konfigurointi HiDraw-sovelluksen näkökulmasta

HiDraw-järjestelmä kokoaa ohjelmakoodin graafisten symbolien välityksellä ja kääntää lopuksi ohjelmakoodin binaariksi. Sen ensisijaiset ulostulot ovat siis ohjelmakoodi ja valmis ohjelmabinaari.

Suojaussovelluksen konfiguraatio koostuu asetteluista ja kytkennöistä. *Asettelut* eli parametrit ovat yksinkertaisia muuttujia, joiden arvo voi muuttua ohjelman suorituksen aikana. Kytkennät muodostuvat sen sijaan kahdesta osapuolesta: *vastaanottajasta* (input) ja *lähettäjästä* (output). Kytkennät säilyvät samana koko ohjelman eliniän.

Jokaista asettelua, vastaanottajaa ja lähettäjää edustaa yksi HiDraw'n graafinen symboli. Konfigurointi toteutettiin siten, että jokainen symboli kirjoittaa erilliseen tiedostoon tiedon siitä, millaista dataa se haluaa jaetussa muistissa olevan. Tieto kirjoitetaan C-makrona, jonka avulla esikääntäjä tuottaa tietueen, jossa on kenttä jokaisen symbolin vaatimalle datalle. Suojausohjelmasta jaettua muistia käsitellään suoraan tietueen kentän perusteella. Kuvassa 6.2 on havainnollistettu suojausohjelman suhdetta jaettuun muistiin.



Kuva 6.2. *Jaettu muisti HiDraw'n näkökulmasta. Muistialueen alussa ja lopussa on muuttumattomat kentät, joita käyttämällä muistin eheyttä voidaan tarkkailla myös ohjelman suorituksen aikana.*

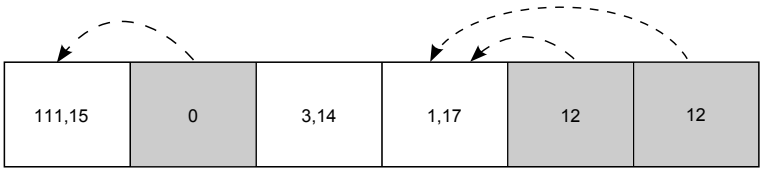
Asettelujen konfigurointi

Asettelut konfiguroidaan yksinkertaisesti alustamalla HiDraw-sovelluksen muuttuja sitä vastaavalla jaetun muistin arvolla. Asettelut luetaan jaetusta muistista säännöllisin väliajoin, joten asettelujen konfigurointi on mahdollista myös sovelluksen ajoaikana.

HiDraw-ohjelma lukitsee jaetun muistin käyttämällä RTX:n poissulkemismekanismia asettelujen lukemisen ajaksi. Jos konfigurointisovellus on lukinnut jaetun muistin ennen HiDraw-sovellusta, HiDraw-sovellus käyttää vanhoja asetteluja. Näin varsinainen suojaustoiminto on käytettävissä koko ajan.

Kytkentöjen konfigurointi

Kytkennät jaettiin konfiguroinnin vuoksi kahteen eri osaan: vastaanottajaan ja lähettäjään. Lähettäjä kirjoittaa kaiken datansa jaettuun muistiin välittämättä siitä, kuka vastaanottaja on. Vastaanottaja on puolestaan se osa kytkentää, jolla varsinainen konfigurointi mahdollistetaan. Jaettuun muistiin on varattu vastaanottajalle kenttä, joka sisältää suhteellisen muistiosoitteen. Kenttä jaetussa muistissa kertoo sen lähettäjän suhteellisen muistiosoitteen, johon vastaanottajan halutaan kytkeytyvän. Vastaanottaja laskee jaetun muistin alkuosoitteen ja suhteellisen muistiosoitteen perusteella lähettäjän sijainnin jaetussa muistissa ja lukee datan vakioviitteen välityksellä. Kuvassa 6.3 on havainnollistettu kytkentöjen konfigurointia.



Kuva 6.3. *Kytkentöjen konfigurointi. Harmaalla taustalla on merkitty vastaanottajan ja valkoisella lähettäjän dataa jaetussa muistissa.*

Vastaanottajaa ei ole myöskään pakko kytkeä mihinkään. Suojausohjelma käyttää tällöin oletusarvoa jaetun muistin sijaan. Jos kytkemätön signaali on suojauslohkon kannalta välttämätön, voidaan koko suojauslohko jättää suorittamatta. Näin suoritusaikaa jää enemmän muille toiminnoille.

6.1.2 Konfigurointisovellus

Konfigurointisovellus on Windowsin päällä toimiva sovellus, jonka tehtävänä on hallita suojausjärjestelmää ja siihen kuuluvia osakomponentteja. Suojausjärjestelmään kuuluu suojaussovelluksen lisäksi IEC 61850-8-1 -pino ja XEX-vuorontaja. Sovellus alustaa ensin jaetun muistin ja käynnistää lopuksi kommunikaatiopinon ja suojaussovelluksen. Suojaussovellus käynnistää tarvitsemansa vuorontajaprosessit automaattisesti.

Jaettu muisti alustetaan SCL-tiedostosta löytyvän konfiguraation perusteella. SCL-tiedostoon on liitetty HiDraw-sovelluksen muuttujanimiä, joiden perusteella alustetaan jaetun muistin sisältö.

Ajoaikainen konfigurointi

Sovellus päivittää jaetun muistin sisällön, mikäli SCL-tiedoston sisältö muuttuu järjestelmän ajoaikana. Koko järjestelmää ohjataan siis SCL-tiedoston ja sen sisällön perusteella. Mikäli parametrien havaitaan muuttuneen, voidaan ne päivittää sovelluksen suorituksen aikana.

Parametrien päivittämisessä hyödynnetään RTX-järjestelmän poissulkemismekanismeja. Suojaussovellus yrittää lukita jaetun muistin parametrien lukemisen ajaksi ja jos lukitseminen epäonnistuu, suojaussovelluksessa käytetään parametrien vanhoja arvoja. Vastaavasti, jos suojaussovellus on ehtinyt lukita jaetun muistin ennen konfigurointisovellusta, konfigurointisovellus odottaa, kunnes se saa varatun resurssin käyttöönsä.

Suojausjärjestelmän uudelleenkäynnistäminen

Suojausjärjestelmä tulee käynnistää uudestaan, mikäli jokin seuraavista muuttuu: kytkennät, järjestelmäparametri tai SCL-tiedoston GOOSE-konfiguraatio. Järjestelmäparametrilla tarkoitetaan parametria, jonka muuttaminen vaatii järjestelmän uudelleenkäynnistämisen. Kytkentöjen ja järjestelmäparamerien muuttuminen vaikuttaa vain suojausohjelman toimintaan, mutta GOOSE-konfiguraation muutokset vaikuttavat koko järjestelmään.

Kytkentöjen ja järjestelmäparametrien muuttaminen vaatii vain suojaussovelluksen uudelleenkäynnistämisen, mutta GOOSE-konfiguraation muuttuminen vaatii sekä suojausohjelman että kommunikaatiopinon uudelleenkäynnistämisen. Kommunikaatiopinon tulee käynnistyä kokonaisuudessaan, ennen kuin suojaussovellus voidaan käynnistää. Tästä johtuen konfiguraatiosovellus odottaa kommunikaatiopinolta viestiä siitä, että pino on alustettu ja suojaussovellus voidaan käynnistää.

6.1.3 Suunnitteluperiaatteet ja arkkitehtuuri

Suojaussovelluksen rakenne haluttiin pitää mahdollisimman yksinkertaisena ja siirtää monimutkainen logiikka konfigurointisovelluksen puolelle. Näin päädyttiin eräänlaiseen isäntä-renki-arkkitehtuuriin, jossa konfigurointisovellus ohjaa suojausjärjestelmää kokonaisuudessaan, eikä suojausjärjestelmä tiedä konfigurointisovelluksesta mitään.

Suunnitteluperiaatteissa noudatettiin varautuvan ohjelmoinnin (defensive programming) periaatteita. Konfigurointisovellus varmentaa aina konfiguraation eheyden niin tarkasti kuin mahdollista ja virhetilanteissa sammuttaa koko järjestelmän. Myös jaetun muistin eheyttä tarkkaillaan suojausohjelman suorituksen aikana, ja jos muistin havaitaan korruptoituneen, suojaussovellus sammutetaan välittömästi. Muistin eheyttä tarkkaillaan

kuvassa 6.2 esitettyjen *startTag*- ja *endTag*-kenttien avulla: jos niiden arvo muuttuu virheellisen muistiviittauksen takia, tiedetään muistin korruptoituneen.

Suojaussovelluksen arkkitehtuuri

Suojausohjelman arkkitehtuuri ja rakenne määräytyivät HiDraw-järjestelmän ehdoilla, joten siihen rakennettiin kommunikaatiomenetelmät, joita käyttämällä konfigurointiohjelma ohjaa järjestelmää. Suojaussovelluksen komponentit eivät tiedä konfigurointisovelluksen olemassaoloa lainkaan, vaan ne kirjoittavat ja lukevat dataa jaetusta muistista ilman tietoa konfigurointisovelluksesta.

Suojaussovelluksen arkkitehtuurin kannalta suurimmat muutokset olivat jaetun muistin käyttöönottaminen ja kytkentöjen jakaminen kahteen eri osakomponenttiin. Kytkentöjen molempien osapuolten kaikki data siirrettiin jaettuun muistiin, jotta kytkentöjen konfigurointi oli mahdollista. Kytkentöjen vastaanottajakomponentin toteutusta jouduttiin muokkaamaan siten, siinä otetaan huomioon jaetun muistin rakenne.

Parametrien konfiguroiminen oli mahdollista siirtämällä niiden data jaettuun muistiin. Ainoa arkkitehtuuriin vaikuttanut seikka oli poissulkemisongelman ratkaiseminen parametrien lukemisen yhteydessä. Poissulkemisongelma ratkaistiin lisäämällä lukitusmekanismi jaetun muistin suojaamiseksi.

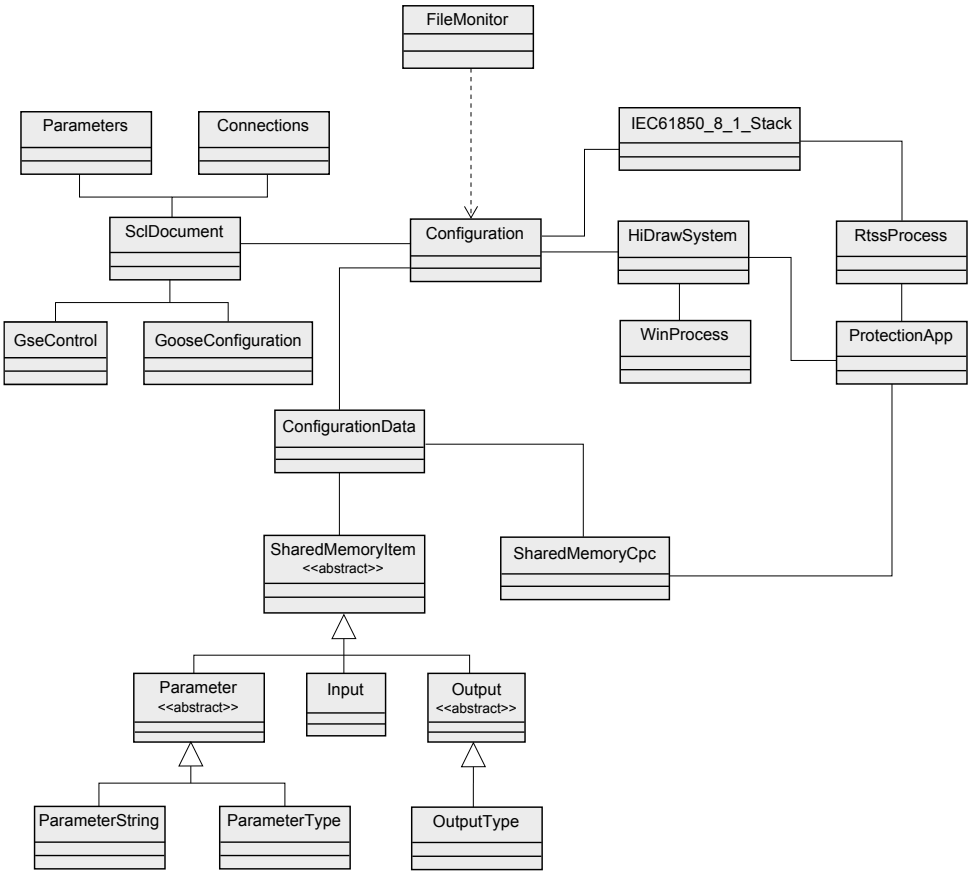
Konfigurointisovelluksen arkkitehtuuri

Konfigurointisovelluksen arkkitehtuuri koostuu neljästä erillisestä osakokonaisuudesta:

1. SCL-tiedoston konfiguraation sisältävästä komponentista,
2. suojausjärjestelmän eri komponentit sisältävästä osasta,
3. jaetun muistin sisällön mallintavasta osasta ja
4. SCL-tiedoston muutoksia ilmoittavasta komponentista.

Kuvassa 6.4 on esitetty luokkakaavio sovelluksen rakenteesta. Arkkitehtuurissa on pyritty pitämään järjestelmän eri osakokonaisuudet erillään toisistaan. Tämä mahdollistaa eri osakokonaisuuksien korvaamisen kokonaan toisenlaisilla komponenteilla. Esimerkiksi jaettu muisti voitaisiin tarvittaessa korvata esimerkiksi tietokannalla, eikä muuhun järjestelmään tarvitsisi tehdä isoja muutoksia.

Myös konfiguraation välittäminen eri ohjelman osilta toisille on pyritty eristämään osakomponenttien sisäisestä toteutuksesta. SCL-tiedoston sisältämän konfiguraation mallintava komponentti välittää tiedon konfiguraation sisällöstä luokkien *Parameters*, *Connections* ja *GooseConfiguration* välityksellä.



Kuva 6.4. *Luokkakaavio konfigurointisovelluksen rakenteesta.*

6.2 PCM600 Connectivity Package -laajennus

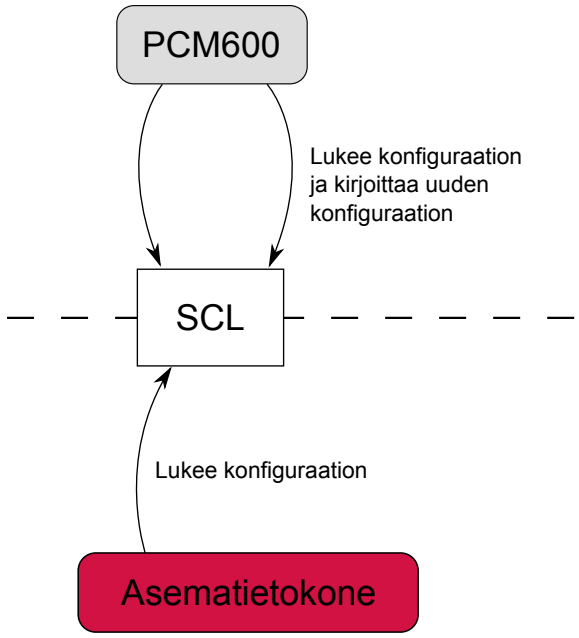
Connectivity Package -laajennusta käytetään konfiguraation kirjoittamiseen SCL-tiedostoon. Kuvassa 6.5 on esitetty kokonaiskuva ConnPack-laajennuksen käytöstä PCM600-työkalun välityksellä. ConnPack-laajennusta käyttämällä on mahdollista lukea olemassa oleva konfiguraatio työkaluun tai rakentaa konfiguraatio manuaalisesti eri toimilohkoista.

ConnPack-laajennus ei tiedä asematietokoneella toimivasta konfigurointisovelluksesta mitään, vaan sen ainoa tehtävä on kirjoittaa SCL-tiedosto ja siirtää se FTP-yhteyden välityksellä asematietokoneeseen. Näin koko järjestelmän molemmat osat ovat täysin riippumattomia toisistaan ja ainoa yhdistävä tekijä niiden välillä on standardoitu SCL-tiedosto.

6.2.1 Arkkitehtuuri ja suunnitteluperiaatteet

ConnPack-laajennus on vahvasti riippuvainen PCM600:n tarjoamasta rajapinnasta. Näin ollen suurin osa arkkitehtuurista määräytyy PCM600:n määrittelemän rajapinnan perusteella.

Asematietokoneen Connpack-laajennuksessa on kuitenkin muutamia poikkeavia ratkaisuja verrattuna aiemmin kehitettyihin ConnPack-laajennuksiin:



Kuva 6.5. *ConnPack-laajennuksen toimintaympäristö.*

– .NET 3.5 -ohjelmistokehyksen ominaisuuksien hyödyntäminen

– parametrien arvot kirjoitetaan SCL-tiedostoon

– SCL-tiedoston valmistajakohtaisia osia on käytetty aiempaa laajemmin.

.NET 3.5 -ohjelmistokehyksen hyödyntäminen helpottaa ennen kaikkea SCL-tiedoston käsittelyä, sillä .NET 3.5 sisältää kehittyneet työkalut XML-tiedostojen käsittelyyn. Näin ConnPack-laajennuksen logiikka on SCL:n käsittelyn kannalta erittäin suoraviivainen.

Parametrien arvojen kirjoittaminen SCL-tiedostoon mahdollistaa monipuolisen konfiguroinnin myös silloin, kun suojalaite on poissa ajossa. Aikaisemmin parametrien asettelu on suoritettu käytönaikaisesti käyttämällä esimerkiksi IEC 61850 -standardin määrittelemää parametrisointia.

6.2.2 Konfigurointiformaatti – IEC 61850 SCL

Konfigurointiformaatiksi valittiin kohdassa 3.3 kuvattu IEC 61850 -standardin mukainen SCL-tiedosto. Standardi sallii valmistajakohtaisten osioiden lisäämisen SCL-tiedostoon, joita työkaluketju käyttää konfiguraation välittämiseen. Kommunikaatiopino on ainoa osa järjestelmästä, joka ei tarvitse tietoa valmistajakohtaisista osioista.

Valmistajakohtainen osa

Järjestelmä käyttää kolmea eri SCL:n valmistajakohtaista osaa, joiden avulla PCM600 ja HiDraw-järjestelmä välittävät tietoa. Toimilohkot, parametrit ja kytkennät tallennetaan erilaisiin valmistajakohtaisiin osiin SCL-tiedostossa.

Parametrien tallentamiseen käytetty osa sisältää pääasiassa PCM600:n vaatimaa tyyppidataa, kuten parametrin ääriarvot ja datan tietotyypin. Listauksessa 6.1 on esimerkki parametrien tallentamiseen käytetystä osasta. Ajoympäristö ja PCM600 vaihtavat tietoa `CommonSA:Value`-elementin välityksellä. `HiDrawName`-attribuutin perusteella määritetään parametrin sijainti jaetussa muistissa. Alla on esimerkki PHIPTOC1-lohkon parametreista. Myös ConnPack-laajennus käyttää samaa tietoa PCM600-vaatiman tyyppidatan mallintamiseen.

```
<commonSA:Object type="Function">
  <commonSA:Node id="PHIPTOC1" guid="9ba61bb3-5f4c-..." />
  <commonSA:Parameters>
  <commonSA:Parameter name="RsDlTmms.setVal" guid="a70cba3c-597d-..."
      HiDrawName="PHIPTOC1_HIDRAW">
       <commonSA:Value>20</commonSA:Value>
       <commonSA:Values dataCategory="settingGroup" type="numerical">
         <commonSA:Min>20</commonSA:Min>
         <commonSA:Max>60000</commonSA:Max>
         <commonSA:Step>10</commonSA:Step>
       </commonSA:Values>
     </commonSA:Parameter>
  </commonSA:Parameters>
</commonSA:Object>
```

Listaus 6.1. *Parametrien tallentamiseen käytetty valmistajakohtainen osa SCL-tiedostossa.*

Toimilohkojen ominaisuudet on tallennettu erilliseen osaan SCL-tiedostossa. Toimilohkojen tallentamiseen käytetty osa sisältää myös PCM600:n tarvitsemaa tietoa, kuten signaalin tyypin, jolla voidaan estää laittomien kytkentöjen tekeminen kohdassa 5.1.3 kuvatussa ACT-työkalussa. Ajoympäristöä varten on tallennettu tieto signaalin HiDraw-nimestä, jotta sen sijainti voidaan määrittää jaetussa muistissa. Esimerkki toimilohkojen mallintamiseen käytetystä osasta on esitetty listauksessa 6.2.

```
<tACT:FB Name="PHIPTOC1" guid="9ba61bb3-5f4c-...">
  <tACT:IN Name="MEASUREMENTS" Pin="1" Type="MUMEAS" guid="43337301-
      d162-..." HiDrawName="SGN_IN_1_HIDRAW" />
  <tACT:OUT Name="OPERATE" Pin="1" Type="BOOLEAN" guid="fd5892a7-d6ae-
      ..." HiDrawName="SGN_OUT_1_HIDRAW" />
</tACT:FB>
```

Listaus 6.2. *Esimerkki toimilohkojen ominaisuuksien mallintamiseen käytetystä osasta.*

Kytkennät on tallennettu erilliseen logiikka-elementtiin ja kytkentöjen merkitsemiseen on käytetty listauksessa 6.2 esitettyjä signaalien `guid`-tunnisteita. Tunnisteiden avulla yhdistetään ajoympäristön lähettäjä (`s`-attribuutti) ja vastaanottaja (`t`-attribuutti) toisiinsa. Listauksessa 6.3 on esitetty SCL-tiedoston valmistajakohtainen osa, johon on tallennettu kytkennät eri toimilohkojen välillä.

```
<tACT:LOGIC WorkSheetInfo="MainApp">
  <tACT:OP s="72755917-8a6b-..." t="43337301-d162-..." />
</tACT:LOGIC>
```

Listaus 6.3. *Tehtyjen kytkentöjen talletamiseen käytetty SCL-tiedoston osa.*

Standardoitu osa

SCL-tiedostoon on mallinnettu standardin vaatimat datatyypit ja toimilohkot. Lisäksi myös GOOSE-viestien välittämiseen käytetyt daniput on mallinnettu SCL-tiedostossa. Datanippu sisältää tiedon eri toimilohkojen lähettämästä datasta. Esimerkki datanipun sisällöstä on esitetty listauksessa 6.4.

```
<DataSet name="prot">
  <FCDA ldInst="LD0" prefix="PHI" lnClass="PTOC" lnInst="1" doName="Op"
      daName="general" fc="ST" />
  <FCDA ldInst="LD0" prefix="PHI" lnClass="PTOC" lnInst="1" doName="Op"
      daName="q" fc="ST" />
  <FCDA ldInst="LD0" prefix="PHI" lnClass="PTOC" lnInst="1" doName="Str
     " daName="general" fc="ST" />
  <FCDA ldInst="LD0" prefix="PHI" lnClass="PTOC" lnInst="1" doName="Str
     " daName="q" fc="ST" />
</DataSet>
```

Listaus 6.4. *Esimerkki IEC 61850 -standardin määrittelemästä datanipusta ja sen sisällöstä.*

Listauksesta nähdään, että datanippu koostuu `FCDA`-elementeistä, joissa määritellään se instanssi, jonka dataa GOOSE-viestinä lähetetäään. Esimerkin nipusta nähdään, että kyse on loogisessa laitteessa `LD0` (`ldInst`-attribuutti) sijaisevasta `PHIPTOC1`-toimilohkosta. Toimilohkon nimi määräytyy attribuuttien `prefix`, `lnClass` ja `lnInst` perusteella. `FCDA`-elementtiin merkitään myös se dataobjekti, jonka dataa lähetetään. Dataan liittyvät yksityiskohdat on tallennettu attribuutteihin `doName`, `daName` ja `fc`.

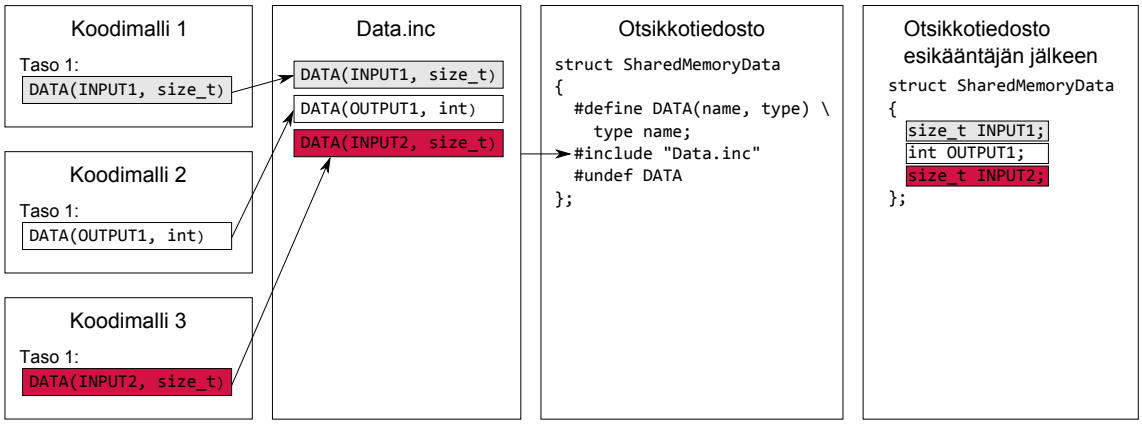
6.3 Toteutuksen arviointi

6.3.1 Ajoympäristön konfigurointi

Ajoympäristön konfigurointi perustuu vahvasti käännösaikaisen tietorakenteen ja jaetun muistin käyttämiseen. Käännösaikaisuus johtuu siitä, että järjestelmässä joudutaan käyttämään esikääntäjädirektiivejä, jotta jaettuun muistiin saadaan tarvittava määrä erityyppisiä kenttiä. Esikääntäjällä tuotetaan tietue, joka tallennetaan jaettuun muistiin. Käytännössä tämä toteutetaan siten, että jokainen HiDraw-piirrosmerkki kirjoittaa yhteiseen tiedostoon esikääntäjädirektiivin, jolla luodaan yksi kenttä jaettuun muistiin tallennettavaan tietueeseen. Kuvassa 6.6 on esitetty esimerkki esikääntäjän hyödyntämisestä jaetun muistin sisällön tuottamisessa. Kuvassa yhtä HiDraw-piirrosmerkkiä edustaa yksi koodimalli.

Käännösaikaisesta riippuvuudesta johtuen suojaussovellusta ja konfigurointisovellusta ei voi eriyttää toisistaan. IEC 61850 -kommunikaatiopino on ainoa osa järjestelmää, jolla ei ole käännösaikaista riippuuvussuhdetta muihin osakomponentteihin.

Esikääntäjädirektiivien avulla tuotettu tietorakenne on kuitenkin muistinkäytöltään optimaalinen, sillä jaettu muisti sisältää vain välttämättömät kentät. Konfigurointisovelluksen



Kuva 6.6. *Esikääntäjädirektiivejä käytetään jaetun muistin sisällön tuottamiseen HiDraw'n generoiman Data.inc-tiedoston välityksellä.*

tietorakenne mahdollistaa jaetun muistin joustavan käsittelemisen ja RTX:n poissulkemismekanismeja käyttämällä rinnakkaisuusongelmat on saatu ratkaistua.

Jaetun muistin käsitteleminen suojaussovelluksesta käsin on erittäin suoraviivaista, eikä se vaadi kuin yhden muistiviittauksen. Konfiguroitavuus ei näin ollen vaikuta merkittävästi reaaliaikavaatimuksiin ja järjestelmän suorituskykyyn.

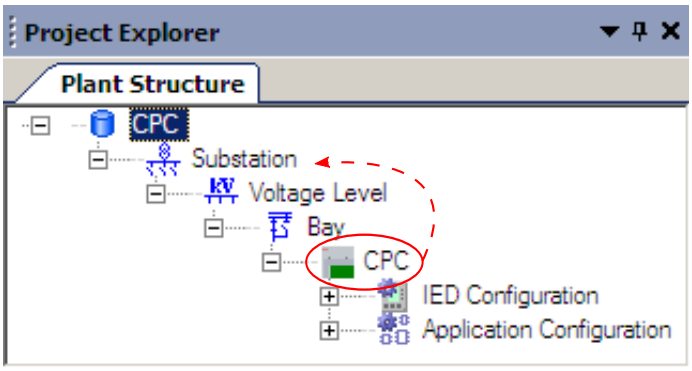
Konfigurointisovelluksen suorituskyky osoittautui hyväksi ja testien perusteella koko järjestelmän konfigurointiin menee aikaa noin 5000 ms ja suojaussovelluksen uudelleenkäynnistämiseen noin 1000 ms. Käytönaikaiseen konfigurointiin menee aikaa noin 5 ms. Vaikka konfigurointisovelluksen suorituskyvyllä ei ole suoraa vaikutusta suojaussovelluksen suorituskykyyn, vaikuttaa se merkittävästi järjestelmän käyttäjäkokemukseen. Hyvän käyttäjäkokemuksen kannalta on tärkeää, että järjestelmän toiminta on nopeaa, eikä sen käytössä ilmene ylimääräisiä viipeitä.

6.3.2 ConnPack-laajennus

Opinnäytetyössä toteutettu ConnPack-laajennus käyttää hyödykseen PCM600:n rajapintoja sekä IEC 61850 -standardin mukaista SCL-tiedostoa. Arkkitehtuuri määräytyi suurelta osin PCM600-sovelluskehyksen ehdoilla. SCL-tiedoston käyttäminen osoittautui .NET-sovelluskehyksen myötä hyväksi ratkaisuksi ja se takaa myös hyvän yhteensopivuuden tulevaisuutta ajatellen.

Suurimmat haasteet liittyivät SCL-tiedostossa käytettyjen valmistajakohtaisten osien rakentamiseen siten, että kaikki tarpeellinen tieto voidaan välittää ajoympäristölle. Toteutuksessa päädyttiin käyttämään ajoympäristön muuttujanimiä, joiden perusteella jaettu muisti kirjoitetaan. Tämä ratkaisu vaatii sovelluskehittäjältä manuaalista käsityötä, joka tekee järjestelmän kuvaavan konfiguraatiotiedoston tuottamisesta työlästä ja virhealtista.

PCM600:n rajoituksista johtuen asematietokone jouduttiin esittäämään tavallisena suojareleenä konfigurointinäkymässä. Looginen sijainti hiearkiassa olisi ollut asematasolla kahta tasoa korkeammalla. Kuvassa 6.7 on havainnollistettu ongelmaa.



Kuva 6.7. *PCM600-työkalun projektinäkymän puutteista johtuen asematietokoneen objekti jouduttiin sijoittamaan kennotasolle asematason sijaan.*

IEC 61850-9-2 -kommunikaation konfigurointi

Standardoidun IEC 61850-9-2 -liikenteen mallinnus PCM600-työkalussa toteutettiin yksinkertaisena toimilohkona. Toimilohko pitää sisällään *svID*-tunnisteen, jolla yksilöidään mittadataa lähettävä liittymisyksikkö.

Toimilohkon asettelut sisältävät standardin määrittelemiä ominaisuuksia, kuten lähetystaajuuden ja lähetettävän datan tietotyypit. IEC 61850-9-2 -kommunikaation mallintaminen toimilohkona on intuitiivinen ja helppokäyttöinen ratkaisu PCM600-työkalun käyttäjän näkökulmasta.

IEC 61850-8-1 -kommunikaation konfigurointi

GOOSE-kommunikaation konfiguraatio määritellään SCL-tiedostossa. PCM600-työkalu on ensisijaisesti suojalaitteen konfigurointityökalu, eikä siinä ole järjestelmätyökalussa vaadittuja ominaisuuksia. Tästä johtuen GOOSE-kommunikaation konfigurointi ei kokonaisuudessaan ole mahdollista PCM600-työkalussa.

GOOSE-kommunikaation konfiguroimiseksi päätettiin käyttää ABB:n kehittämää Communication Configuration Tool (CCT) -työkalua. CCT-työkalu on IEC 61850 -standardin kuvaaman konfigurointiprosessin mukainen järjestelmätyökalu. CCT-työkalulla on mahdollista konfiguroida suojalaitteiden välinen kommunikaatio ja sitä käytetään esimerkiksi GOOSE-kommunikaation konfiguroimiseen.

7 JOHTOPÄÄTÖKSET

Opinnäytetyössä määriteltiin vaatimukset asematietokoneen konfigurointityökalulle ja se toteutettiin laajennuksena ABB:n kehittämälle PCM600-työkalulle. Työssä määriteltiin myös konfigurointityökalun ja asematietokoneen välisessä kommunikaatiossa käytetty tiedostoformaatti. Lisäksi asematietokoneen ohjelmistoa muokattiin tukemaan konfigurointia.

Konfigurointityökalun toteutus pohjautuu ABB:n PCM600-työkalun päälle tehtyyn ConnPack-laajennukseen. PCM600-työkalun arkkitehtuuri tarjoaa yksinkertaisen ja monipuolisen rajapinnan ConnPack-laajennusten käytettäväksi. PCM600:n modulaarinen rakenne mahdollisti sen, että opinnäytetyössä voitiin toteuttaa vain tarvittavat työkalumoduulit. ConnPack-laajennuksen kehittäminen ja laajentaminen voidaan näin ollen tehdä hallitusti myös jatkossa.

Asematietokoneen ConnPack-laajennus mallintaa asematietokoneen ohjelmiston käyttäen IEC 61850 -standardin mukaista SCL-tiedostoa. SCL-tiedostoon on liitetty myös järjestelmäkohtaista tietoa, jota käyttämällä asematietokoneen ohjelmiston eri osat kommunikoivat keskenään. Stardardoidun SCL-tiedoston käyttäminen mahdollistaa hyvän yhteensopivuuden myös seuraavan sukupolven asematietokoneiden kanssa.

Asematietokoneen ajoympäristön konfigurointi osoittautui ennakoitua haasteellisemmaksi tehtäväksi. Ajoympäristö on vahvasti rippuvainen ABB:n kehittämästä HiDraw-työkalusta, jolla asematietokoneen ohjelmakoodi generoidaan ja käännetään binaariksi. Samalla työkalulla käännetään myös konfigurointisovelluksen ohjelmabinaari. Ajoympäristö ja konfigurointisovellus jakavat käännösaikaisesti tuotetun tietorakenteen, joka sitoo ne vahvasti toisiinsa jo käännösaikana.

Käännösaikainen riippuvuus vaikeuttaa ajoympäristön eri osien kehittämistä, sillä niitä ei voida kehittää erillään, vaan molemmat osat täytyy kääntää samalla työkalulla. Ongelma voitaisiin ratkaista käyttämällä tietokantapohjaista konfigurointia seuraavan sukupolven toteutuksessa. Tällöin riippuvuussuhde muuttuisi ajoaikaiseksi, joka mahdollistaisi ajoympäristön eri osien kehittämisen itsenäisinä sovelluksina.

Ajoympäristön ohjelmistoalustan soveltuvuus asematietokoneen ohjelmiston pohjaksi on myös hieman kyseenalainen. Ohjelmistoalustan puutteellisen dokumentaation vuoksi kaikkien havaittujen ongelmien syytä ei pystytty täysin selvittämään. Ongelmat pystyttiin kuitenkin osaksi kiertämään muita ratkaisuja käyttämällä. Ohjelmistoalustan käyttökelpoisuuden varmentamiseksi tulisi rakentaa kattava testiympäristö, jolla sen toimivuutta voitaisiin arvioida eri tilanteissa.

Opinnäytetyön lopputulokseksi kehitettiin toimiva konfigurointiympäristö, jota käyttämällä asematietokoneen ohjelmistoa on mahdollista konfiguroida. Asematietokoneen ajoympäristön kehittämisen aikana huomattiin, että ohjelmistoympäristön skaalautuvuus ei ole toivotulla tasolla ja sen käyttökelpoisuutta tulee arvioida lisää. Saavutettujen tulosten perusteella voidaan kuitenkin todeta, että keskitetty suojaus- ja ohjaustekniikka on käyttökelpoinen, mutta sen todelliset hyödyt nähdään vasta, kun se otetaan käyttöön isommassa mittakaavassa.

LÄHTEET

[1] ABB Ltd. Architecture Specification for PCM600. Internal document. ABB Powersystems, 2010.

[2] ABB Ltd. Centralized Protection and Control – Architecture and Design. Internal document. ZCRD Corporate Research, 2007.

[3] ABB Ltd. Code Generation in HiDraw. Internal document. ABB Power Systems, 1997.

[4] ABB Oy Distribution Automation. Protection and Control IED Manager PCM600 – User Manual. 2010.

[5] Ardence RTX. RTX SDK Documentation. Citrix Systems Inc., 2007.

[6] Baldinger, F., Jansen, T., van Riet, M. & Volberda, F. Nobody knows the future of Smart Grids, therefore separate the essentials in the secondary system, Manchester, 29/03/2010–01/04/2010. 2010, The Institution of Engineering and Technology.

[7] Brunner, C. IEC 61850 Process Bus – Challenges and Benefits. UTInnovation, Switzerland.

[8] Brunner, C., Lang, G., Leconte, F. & Steinhauser, F. Implementation Guideline for Digital Interface to Instrument Transformers using IEC 61850-9-2. 2004, UCA International Users Group. 21 p.

[9] Hakala-Ranta, A., Rintamäki, O. & Starck, J. Utilizing possibilities of IEC 61850 and GOOSE, Praque, 08/06/2009–11/06/2009. 2009, CIRED.

[10] IEC 61850-1. Communication networks and systems in substations – Part 1: Introduction and overview. 1st ed. 2004, International Electrotechnical Comission. 37 p.

[11] IEC 61850-2. Communication networks and systems in substations – Part 2: Glossary. 1st ed. 2003, International Electrotechnical Comission. 42 p.

[12] IEC 61850-5. Communication networks and systems in substations – Part 5: Communication requirements for functions and device models. 1st ed. 2004, International Electrotechnical Comission. 131 p.

[13] IEC 61850-6. Communication networks and systems in substations – Part 6: Configuration description language for communication in electrical substations related to IEDs. 1st ed. 2004, International Electrotechnical Comission. 144 p.

[14] IEC 61850-6. Communication networks and systems in substations – Part 6: Configuration description language for communication in electrical substations related to IEDs. 2nd ed. 2009, International Electrotechnical Comission. 215 p.

[15] IEC 61850-7-1. Communication networks and systems in substations – Part 7-1: Basic communication structure for substation and feeder equipment – Principles and models. 1st ed. 2003, International Electrotechnical Comission. 110 p.

[16] IEC 61850-7-2. Communication networks and systems in substations – Part 7-2: Basic communication structure for substation and feeder equipment – Abstract communication service interface (ACSI). 1st ed. 2003, International Electrotechnical Comission. 171 p.

[17] IEC 61850-7-4. Communication networks and systems in substations – Part 7-4: Basic communication structure for substation and feeder equipment – Compatible logical node classes and data classes. 1st ed. 2003, International Electrotechnical Comission. 104 p.

[18] IEC 61850-8-1. Communication networks and systems in substations – Part 8-1: Specific Communication Service Mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3. 1st ed. 2004, International Electrotechnical Comission. 133 p.

[19] IEC 61850-9-2. Communication networks and systems in substations – Part 9-2: Specific Communication Service Mapping (SCSM) – Sampled values over ISO/IEC 8802-3. 1st ed. 2004, International Electrotechnical Comission. 27 p.

[20] IntervalZero, Inc. Hard Real-Time with IntervalZero on the Windows Platform. 2009. [WWW]. [Cited 19/07/2010]. Available: http://www.intervalzero.com/rtx.htm.

[21] Koskimies, K. & Mikkonen, T. Ohjelmistoarkkitehtuurit. Helsinki 2005, Talentum Media Oy. 250 s.

[22] Kumpulainen, L. et al. Verkkovisio 2030 – Jakelu- ja alueverkkojen teknologiavisio. 2006. [WWW]. [Viitattu 27.01.2011]. Saatavissa: http://www.vtt.fi/inf/pdf/tiedotteet/2006/T2361.pdf.

[23] Locamation B.V. SASensor – The Power of Simplicity. 2010. [WWW]. [Cited 08/03/2010]. Available: http://www.locamation.nl/folders_pdf/loc_sas.pdf.

[24] Mackiewicz, R. Overview of IEC 61850 and Benefits. [WWW]. [Cited 23/02/2011]. Available: http://140.98.193.141/portal/cms_docs_pes/pes/subpages/meetings-folder/T_D_2005_2006/tuesday/pn10/05TD0235.pdf.

[25] Mörsky, J. Relesuojaustekniikka. 2. painos. Hämeenlinna 1993, Otatieto. 459 s.

[26] Odnegård, G. HiDraw, the Tool for Functional Block Programming. Internal document. ABB Powersystems, 2005.

[27] Valtari, J., Hakola, T. & Verho, P. Station Level Functionality in Future Smart Substations. 2010, Nordac.

[28] Valtari, J., Verho, P., Hakala-Ranta, A. & Saarinen, J. Increasing Cost-efficiency of Substation Automation Systems by Centralised Protection Functions, Praque, 08/06/2009–11/06/2009. 2009, CIRED.

[29] Volberda, F., van Riet, M. & Pikkert, A. The Power of Simplicity, Vienna, 21/05/2007–24/05/2007. 2007, CIRED.

LIITE 1: ESIMERKKI HIDRAW-PIIRROKSESTA