Aku-Petteri Partanen

KÄYTTÖTAPARYHMIEN OHJELMOINNIN AUTOMATISOINTI SIEMENS TIA PORTAL -YMPÄRISTÖSSÄ

Sähkö- ja automaatiotekniikan koulutusohjelma
2019

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

KÄYTTÖTAPARYHMIEN OHJELMOINNIN AUTOMATISOINTI SIEMENS TIA PORTAL -YMPÄRISTÖSSÄ

Partanen, Aku-Petteri
Satakunnan ammattikorkeakoulu
Sähkö- ja automaatiotekniikan koulutusohjelma
Heinäkuu 2019
Sivumäärä: 56

Liitteitä: 0

Asiasanat: Openness, Siemens, TIA Portal, Automatisointi, PLC

Työn tarkoituksena oli tutkia mahdollisuutta logiikkaohjelmistokehityksessä käytettävien ohjelmistorakenteiden automaattista tuottamista Siemens TIA Portal -ympäristössä.

Tarve tutkimukselle tuli yrityksen ohjelmistopuolen edustajien halusta vähentää käsin tehtävää kopiointityötä projekteissa. Täten voitaisiin vapauttaa työvoimaa mielekkäämpien tehtävien pariin, sekä vastata tulevaisuuden, yhä laajempien projektien asettamiin haasteisiin.

Työssä perehdyttiin kattavasti ja monipuolisesti logiikkavalmistajan tarjoamiin mahdollisuuksiin sekä työkaluihin ohjelmistotuotannon automatisointiin, sekä näiden käyttöä vaativien teknologioiden ja metodien käyttöön. Saatujen tutkimustulosten perusteella kehitettiin sovellus demokäyttöön, jonka rungon avulla on mahdollista lähteä toteuttamaan laajempimittaista hanketta ohjelmistotuotannon automatisointiin liittyen yrityksessä.

Tuloksena tästä opinnäytetyöstä saatiin tutkimusmateriaalin lisäksi selkeä pohja tulevaisuuden hankkeita ajatellen, sekä todettiin Siemens Openness -rajapinnan hyödynnettävyys sovelluksessa. Näiden lisäksi selvitettiin yrityksen nykyisten toimintamallien ja dokumentaation sopivuutta sovellukselle automatisointiin, sekä tunnistettiin kohdat, joihin vaaditaan lisäkehitystyötä yrityksessä.

SOFTWARE DEVELOPMENT AUTOMATISATION OF
OPERATION MODE GROUPS IN SIEMENS TIA PORTAL

Partanen, Aku-Petteri
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Electrical and automation engineering
July 2019
Number of pages: 56
Appendices: 0

Keywords: Openness, Siemens, TIA Portal, Automation, PLC

The purpose of this thesis was to investigate the feasibility to use automated production of software structures in PLC (programmable logic controller) software development in the Siemens TIA Portal environment.

The company's software representatives expressed a desire to reduce the amount of manual entry work currently needed in projects, therefore freeing employees to work on more pressing issues. A positive outcome of this research would also aid the company in meeting the challenges of increasingly broader scopes of projects in the future.

This thesis comprehensively explores the possibilities offered by the logic manufacturer, the tools for automating software production and the use of technologies and methods required for their use. Based on the results of the research, an application was developed for demonstration purposes, the framework of which acted as a prelude to a larger-scale project related to automation of software production in the company.

In addition to receiving the results of this thesis, the company also received a detailed template for future projects and the baseline suitability of Siemens Openness was established. In addition, the company's current operating models and documentation methods were reviewed to determine suitability for software automation and areas where further development was required were identified.

SISÄLLYS

1 JOHDANTO

Opinnäytetyö käsittelee Siemensin PLC -ohjelmistotuotannon automatisointiin käytettäviä työkaluja ja metodeja, ja sen keskiössä olevaa Siemens Openness -rajapintaa. Työssä sivutaan myös ohjelmistotuotannon ja ohjelmakoodin standardisointia sen liittyessä vahvasti myös edellä mainittuun automatisointiin.

Standardisoimalla ohjelmakoodia voidaan jo pelkästään isoissa ohjelmistoprojekteissa säästää tuhansia tunteja työaikaa vuodessa, ja näin vapauttaa työvoimaa mielekkäämpiin ja haastavampiin työtehtäviin.
Kuitenkaan aivan kaikkea ohjelmaa ei voida projekteissa standardisoida, vaan vielä jää jäljelle paljon ”kopioi – liitä” työtä. Tällainen itseään toistava, rutiinityö on omiaan automatisoitavaksi, ja eri logiikkavalmistajilla tähän onkin kehitetty erilaisia ratkaisuja.

Siemensin ratkaisu ongelmaan on Openness-rajapinta, joka tarjoaa käyttäjälleen keinon päästä TIA Portal -ympäristöön muokkaamaan ja tuottamaan ohjelmakoodia ulkoisen sovelluksen kautta. Työssä käsitellään tämän sovelluksen kehittämiseen vaadittavia teknologioita sekä työkaluja kattavasti, ja siinä vertaillaan myös eri vaihtoehtoja sovelluksen toteuttamiselle.

Työn tavoitteena on tutkia Openness -rajapinnan suomia mahdollisuuksia ja sen sopivuutta yrityksen käyttötaparyhmien ohjelmoinnin automatisointiin. Tutkimuksen tueksi kehitetään demokäyttöön sovellus jonka avulla olisi mahdollista pystyä tuottamaan ainakin pääpiirteittäin yksinkertaisia käyttötaparyhmälohkoja. Työssä selvitetään myös jatkokehityksen kannalta oleellisia tarpeita sekä haastetaan yrityksen ohjelmistotuotannon nykyistä toimintamallia.

2 CIMCORP OY

Cimcorp Oy on automaatiojärjestelmien kansainvälinen toimittaja, ja sen pääkonttori sijaitsee Ulvilassa. Tytäryhtiöitä Cimcorpilla on Kanadassa, Yhdysvalloissa ja Intiassa, sekä kolme huoltoa tarjoavaa toimipistettä Suomessa. Rengasteollisuuden saralla Cimcorp on maailman johtavia automaatiojärjestelmien toimittajia, ja sillä on myös vahva jalansija postinjakelun ja vähittäiskaupan keräilyjärjestelmien toimittajana. (Cimcorp-konserni pähkinänkuoressa)

Cimcorpin toiminta alkoi vuonna 1975 Rosenlewin työkalutehtaan automaatio-osastona. Modulaarisen robotiikkajärjestelmän kehitys sai alkunsa Valcon kuvaputkitehtaan suurtilauksesta, johon Rosenlew valittiin toimittajaksi materiaalinkäsittelylaitteille ja roboteille. Vuonna 1986 automaatio-osastosta tuli Wärtsilän tytäryhtiö, Cimcorp Oy. Yrityskauppojen myötä vuonna 1996 yritys siirtyi Swisslogin omistukseen, ja muutti nimensä Swisslog Oy:ksi. Loppuvuodesta 2003 yhtiö siirtyi nykyisen johdon omistukseen, ja on toiminut vuoden vaihteen jälkeen eteenpäin jälleen nimellä Cimcorp Oy. (Robotiikkaa vuodesta 1975)

2000-luvulla alkoi liiketoiminta siirtyä uusille alueille, ja Cimcorp alkoi automatisoida elintarviketeollisuuden lähettämöitä ja rengasteollisuutta. Vuonna 2010 Cimcorp laajentui maailmalle ostamalla Kanadalaisen RMT Roboticsin, ja 2014 japanilainen Murata Machinery Ltd taas osti Cimcorpin koko osakekannan, jolloin suomalainen yritys sai taakseen huomattavan vankat hartiat. Järjestelmien kasvaessa materiaalivirran hallinta on saanut entistä suuremman roolin, ja tähän Cimcorp on vastannut omilla valmistuksenohjausohjelmistoilla, sekä tällä hetkellä käynnissä olevalla tuotantotilojen laajennuksella. (Robotiikkaa vuodesta 1975)

3 SIEMENS TIA PORTAL

3.1 TIA Portal V15

TIA Portal (Totally Integrated Automation Portal) on Siemensin automaatiolaitteistojen ohjelmointiin tarkoitettu integroitu kehitysympäristö. Aiemmin logiikan, käyttöliittymän ja liikkeenohjauksen ohjelmistokehitykseen on käytetty erillisiä kehitysympäristöjä, mutta TIA Portalin myötä nämä kaikki on saatu yhdistettyä saman ohjelmiston sisälle, joka helpottaa huomattavasti automaatio-ohjelmistokehitystyötä. TIA Portal on myös ensimmäinen työkalu, joka mahdollistaa yhden kehitysympäristön kaikille automaatiotoiminnoille. (Siemens: Teollisuuden tuotteet ja ratkaisut)

TIA Portal V15 sisältää uusia toimintoja jotka mahdollistavat korkeamman suunnittelutehokkuuden saavuttamisen. Tärkeimpiä ominaisuuksia ovat sovellusmahdollisuuksien laajennukset, kuten Multifunctional Platform, 2D- ja 4D-kinematiikan integrointi S7-1500-logiikkaohjaimiin ja robottien liitettävyys ohjelmointiin. (Fokus sovelluksissa, digitalisaatiossa ja tehokkuudessa)

Multifunctional Platform kuuluu Simatic S7-1500 tuoteperheeseen, joka mahdollistaa logiikkaohjelmien tekemisen käyttäen korkeamman tason ohjelmointikieltä (C/C++) perinteisten PLC-ohjelmointikielien lisäksi. (Fokus sovelluksissa, digitalisaatiossa ja tehokkuudessa)

Teollisuus 4.0 -ilmiön myötä digitalisaatio on kasvava trendi teollisuudessa, ja tuotantoprosesseja pyritään optimoimaan automaatiosuunnittelussa muun muassa virtuaalimaailman ja integroinnin siivittämänä (Teollisuus 4.0 – ”Suomen oltava kilpailukykyinen vaihtoehto, kun teollisuuden paluumuutto Aasiasta Eurooppaan alkaa”, 2014 -9-27). Siemens on vastannut tarpeeseen sisällyttämällä TIA Portal V15 -ympäristöön OPC-UA -toiminnallisuuden sekä mahdollisuuden virtuaaliseen käyttöönottoon simulaatiomallien avulla. S7-1500 -tuoteperheen OPC-UA -ominaisuudet helpottavat koneiden välistä standardisointia sekä kommunikointia tehtaan ylempien ohjaus- ja hallintajärjestelmien kanssa. (Fokus sovelluksissa, digitalisaatiossa ja tehokkuudessa)

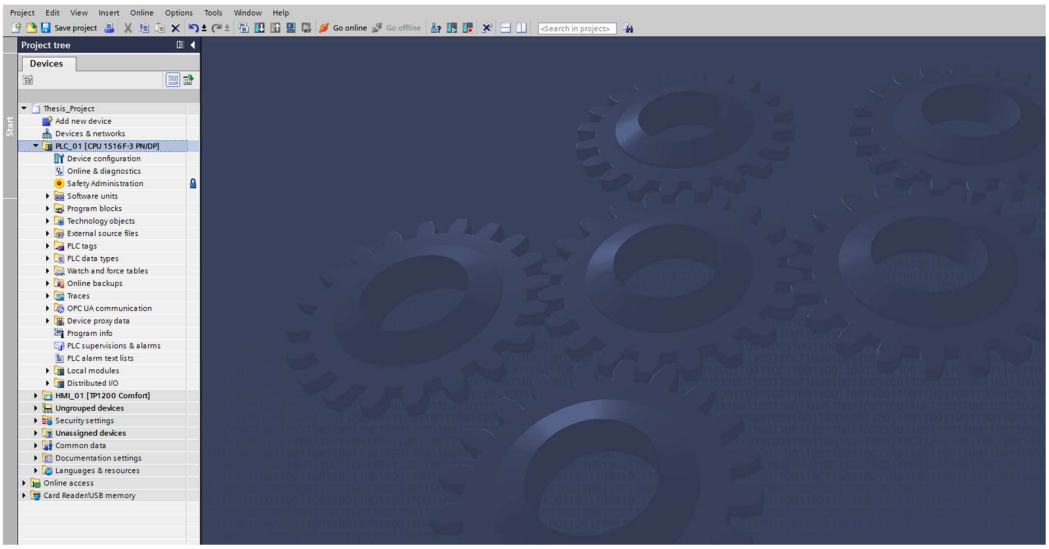
Virtuaalisen käyttöönoton myötä voidaan jo ennen asennusvaihetta testata PLC-ohjelman ja mekaanisen mallin toimintaa yhdessä, joka säästää aikaa käyttöönottovaiheessa sekä vähentää virheitä suunnittelussa. Tämän mahdollistaa Siemensin oma virtuaalinen S7-1500-ohjain, Simatic S7-PLCSIM Advanced -ohjelmisto. (Fokus sovelluksissa, digitalisaatiossa ja tehokkuudessa)

3.2 Simatic Step7

Simatic Step7 toimii keskeisenä suunnitteluohjelmistona ja on perustana kaikkien Siemensin logiikoiden ohjelmoinnissa. Se on osana TIA Portal -kehitysympäristöä, ja vanhaan Step7 Classic -ympäristöön verrattuna moni asia on muuttunut ja kehittynyt. (Siemens: Teollisuuden tuotteet ja ratkaisut)

Uutta Step7-ympäristöä kehitettäessä on huomioitu erityisesti käyttöliittymän helppokäyttöisyys ja opittavuus, sekä ennen kaikkea suunnittelun tehokkuus. Uusia ominaisuuksia on esimerkiksi valokuvamainen laitekonfiguraatio, älykäs drag/drop-toiminnallisuus sekä helppokäyttöiset uudistetut työkalut. (TIA Portal SIMATIC STEP 7)

TIA Portal Step7 tukee kaikkia PLCopen-standardin IEC 61131-3 määrittelemiä ohjelmointikieliä, jotka ovat FBD, LAD, IL, GRAPH (SFC) ja SCL (ST). Muuttujien hallinnointi on tehty aikaisempaa helpommaksi yhteisen muuttujataulun avulla ja konfiguraatioon kuuluvat laitteet ovat selkeästi löydettävissä vasemmalta projektihakemistoluettelosta. Uudessa Step7-käyttöliittymässä on myös selkeästi havaittavissa piirteitä ohjelmointiympäristön lähestymisestä muiden valmistajien ympäristöjä. (Kuva 1) (TIA Portal SIMATIC STEP 7)



Kuva 1. TIA Portal -projektihakemisto ja ulkoasu

3.3 Kirjastointi

Uutta TIA Portal -ympäristössä verrattuna vanhaan Step7 Classic -ympäristöön on vakiolohkoille ja komponenteille tarkoitettu kirjasto. Vanhassa ympäristössä jouduttiin kopioimaan koodia projektista toiseen, jonka seurauksena myös mahdolliset virheet kopioituvat ja näin ollen kertautuvat. (Siemens: Tuhansien tuntien säästö)

Käytännössä kirjastointi tarkoittaa ohjelmalohkojen ja komponenttien, kuten funktioiden (FC) ja toimilohkojen (FB) arkistointia. Valmiita projektipohjia voidaan myös kirjastoida, ja näitä pohjia kutsutaan ”master copies”, eli isäntäkopioiksi. Isäntäkopioihin voidaan tallentaa melkein mitä tahansa objekteja. Näitä isäntäkopioita voidaan käyttää projekteissa itsenäisenä objektina, tai pohjana uudelle projektille. Isäntä kopioihin voidaan tallentaa esimerkiksi laitteita, jotka sisältävät laitekonfiguraation, muuttujatauluja, PLC-datatyyppejä (UDT), teknologiaobjekteja ja paljon muuta. (Kuva 2) (Guideline for Library Handling, 68 - 69)

Library element	Type	Master copies
SIMATIC PLC		
OB	-	X
FB	X	X
FC	X	X
DB (global)	-	X
Technology objects	-	X
PLC tags	-	X
PLC data types	X	X
Watch and force tables	-	X
Traces	-	X
Text lists	-	X
SIMATIC HMI		
Screens	X	X
Faceplates	X	-
Templates (of screens)	-	X
Pop-up screens	-	X
Slide-in screens	-	-
HMI tags	-	X
Scripts	X	X
Protocols	-	X
HMI UDT	X	-
HMI style	X	-
HMI style sheet	X	-
User administration: User	-	X
User administration: Groups	-	X

Kuva 2. ”Master copies” ja ”Types”: Sallitut kirjastoelementit. (Guideline for Library Handling, 69)

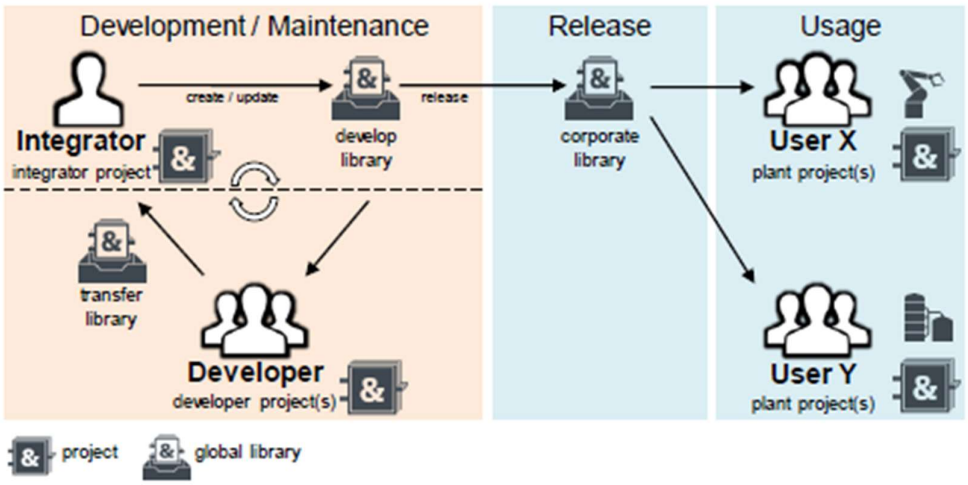
Yksittäiset kirjastokomponentit, kuten edellä mainitut ohjelmalohkot FB ja FC sijoittuvat kirjastokansioissa ”types” kansioon, eli tyyppeihin. Tyyppeihin on hyvä tuoda objekteja, joita tarvitaan itse ohjelman suoritukseen, eli toisin sanoen parametroitavia ohjelmalohkoja sekä tietueita. Tyypit ovat versioitavissa, eli muutoksia tehtäessä myös aikaisemmat tyyppiversiot ovat edelleen käytettävissä ja löydettävissä. Projektit, joissa on vanhoja tyyppiversioita käytössä ohjelmassa, ovat helposti päivitettävissä. (Guideline for Library Handling, 5–7)

TIA Portal -ympäristössä voi olla käytössä kahdenlaisia kirjastoja: globaaleja kirjastoja ja paikallisia, eli projektikirjastoja. Globaalit kirjastot sijaitsevat tyypillisesti yrityksen palvelimella, ja ne ovat sieltä käsin hyödynnettävissä kaikissa projekteissa. Paikalliset, projektikohtaiset kirjastot ovat taas nimensä mukaan projektikohtaisia.

Projekteissa on usein laitteita, jotka ovat toiminnaltaan niin spesifejä, ettei globaalin kirjaston komponentit niiden ohjelmointiin käy, ja toisaalta ei ole järkevää sellaista globaaliin kirjastoon luodakaan. Tällaisessa tilanteessa projektikirjasto on hyvä paikka komponentille, jos sitä käytetään projektissa useammassa paikassa. (Guideline for Library Handling, 5–7)

Suurimpana etuna kirjastoinnissa on versionhallinta. Kun kirjastokomponenttia päivitetään, voidaan se helposti päivittää koko projektiin, ja näin ollen välttää vanhojen versioiden olemassaolo projektissa. Versioinnin myötä myös kirjaston ylläpito on systemaattisempaa, ja sitä varten kannattaa yrityksessä nimetä tietyt vastuuhenkilöt ja jakaa roolit kirjaston ylläpidossa. (Guideline for Library Handling, 5–7)

Tehtävät voidaan jakaa kolmeen toimenkuvaan. Integraattori toimii vastuuhenkilönä kirjastojen ylläpidossa, ja huolehtii siitä, että kirjastokomponenttien uusimmat versiot ovat koko ajan käyttäjien saatavilla. Näiden globaalien kirjastokomponenttien kehittämisestä vastaa kehitystiimi, joka saattaa uusimmat versiot integraattorille. Käyttäjät käyttävät näitä globaaleja kirjastoja, vastaavat uusien versioiden päivittämisestä projekteihin sekä projektikohtaisten kirjastojen ylläpidosta (Kuva 3). (Guideline for Library Handling, 15-16)



Kuva 3. Roolit ja vastuut organisaation kirjastojen ylläpidosta (Guideline for Library Handling, 15)

4 SIEMENS OPENNESS

4.1 Avoimet ohjelmointirajapinnat

API (Application Programming Interface), eli ohjelmointirajapinta toimii ohjelmiston rajapintana ulkopuolisten ohjelmistojen ja tietojärjestelmien välillä ja määrittelee, miten se tarjoaa tietoa ja palveluita näille. Rajapinta voi toimia pelkkänä datarajapintana mahdollistaen ainoastaan tiedon lukemisen palvelusta, tai se voi myös tarjota toiminallisuuksia, joiden avulla voidaan palvelun tietoja muuttaa ja laskenta-algoritmeja käyttää rajapinnan kautta. (Avoimen rajapinnan määritelmä)

Rajapintaa voidaan kutsua avoimeksi, jos se on avoimesti dokumentoitu, käyttöönotettavissa ja testattavissa. Rajapinnan tulee siis olla määritelty ja sen on oltava verkon yli vapaasti kaikkien saatavilla. Dokumentaation tulee olla julkisesti riittävää rajapinnan vaivattoman käyttöönoton mahdollistamiseksi ja sen tulee myös olla käyttöönotettavissa ilman ylläpitäjän tai järjestelmän toimia virka-ajan ulkopuolellakin. Rajapinnan testausta varten tulee olla tarjolla vähintään testiaineisto, ja testaus voidaan toteuttaa esimerkiksi avoimella pääsyllä tuotantojärjestelmään tai testijärjestelmään, joka voi mahdollisesti olla myös vapaasti ladattavissa omaan käyttöön. (Avoimen rajapinnan määritelmä)

Näistä määrittelyistä huolimatta rajapinnan kautta saatavan datan ei tarvitse olla avointa dataa, ja tuotantojärjestelmä voi olla kokonaan irti internetistä ja vapaalta pääsyltä. Esimerkiksi potilastietokannan tiedot eivät voi olla kaikkien saatavilla, vaikka potilastietojärjestelmään itsessään voi olla avoin rajapinta.

4.2 Openness rajapintana TIA Portal -ympäristöön

Siemens Openness on Siemensin kehittämä API-rajapinta, joka mahdollistaa ulkoisten sovellusten datan siirron ja toimintojen suorittamisen TIA Portal -ympäristöön. Rajapinta sisältyy normaaliin TIA Portal -asennukseen jo versiosta V12 lähtien, ja on vapaasti käyttöönotettavissa. Openness-rajapinnan käyttöönottoa varten pitää Window-

sin yksi käyttäjätileistä määritellä ”Siemens Openness user” ryhmään tietokoneen asetuksista, joka antaa käyttäjän tekemille sovelluksille oikeuden käyttää rajapintaa. (TIA Portal Openness: Introduction and Demo Application, 4)

Openness-rajapintaa hyödyntäviä sovelluksia voidaan ohjelmoida käyttäen Microsoft Visual Studio -integroitua kehitysympäristöä, ja valittavina ohjelmointikielistä on C# tai VisualBasic.Net. Rajapinta itsessään tarjoaa lukuisia funktioita, joita hyödyntämällä käyttäjä voi luoda sovelluksia joilla automatisoida TIA Portal -projektin ohjelmointia tuomalla ohjelmalohkoja sekä muita objekteja projektiin. Openness-rajapinta mahdollistaa siis ohjelmistokehityksen automatisoinnin. (TIA Portal Openness: Introduction and Demo Application, 4-8)

Perinteinen tapa tuoda ja generoida ohjelmakoodia on pitkälti käsityötä, ja ainoastaan tekstikielellä ohjelmoitujen ohjelmalohkojen osalla mahdollista. Käyttäjä luo tekstitiedoston, joka on syntaksiltaan ja tallennusformaatiltaan oikeaoppinen, ja tuo sen TIA Portal -ympäristössä ”External Sources” kansioon. Tämän jälkeen pitää kyseinen lähdekoodi vielä generoida ohjelmalohkoiksi projektiin projektihakemistosta.

Openness-rajapinnan ansiosta käyttäjän on mahdollista tuoda projektiin ohjelmalohkoja XML-tiedosto -formaatissa, ja kuvauskieltä käytettäessä ei ole asetettu rajoitteita juurikaan ohjelmalohkon sisältämälle ohjelmointikielelle, esimerkiksi LAD, eli tikapuukaavio kielisen lohkon tuominen on tällöin täysin mahdollista. Ohjelmalohko voidaan generoida automaattisesti käyttäjän luomalla sovelluksella projektiin Openness-rajapinnan ansiosta. Pienessä mittakaavassa tämä ei vielä tuota merkittävää etua, mutta jo hieman isommassa projektissa generoitavia lohkoja voi olla satoja, jolloin voidaan säästää useita työtunteja. (TIA Portal Openness: Introduction and Demo Application, 28-29)

4.3 Rajapinnan tarjoamat toiminnallisuudet

Openness-rajapinnan mukanaan tuomia mahdollisuuksia on kehitetty Siemensin toimesta aina TIA Portalin uusien versioiden julkaisun myötä. Eteenpäin ollaan tultu paljon eri ohjelmointikieliä sisältävien lohkojen käsittelyssä, itse logiikan hallinnassa sekä sovellusten tekemisen joustavuudessa. (SIMATIC Openness: Automating creation of projects, 23-24)

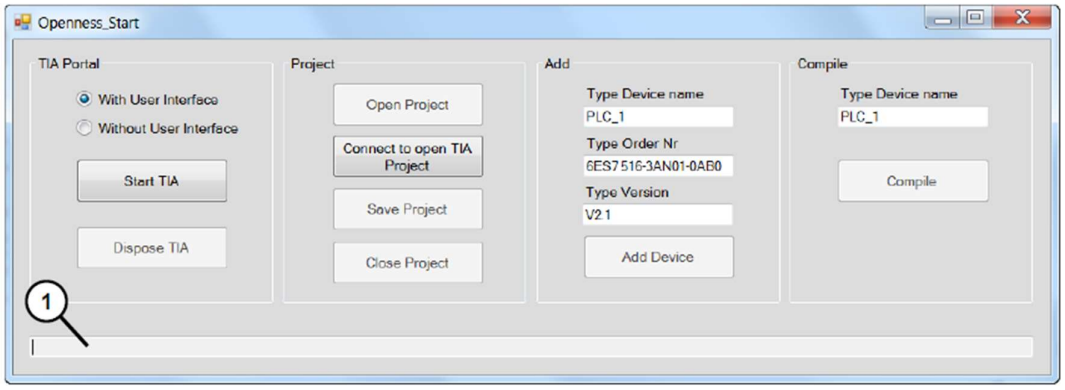
Openness-rajapinnalla voidaan luoda, muokata, lukea ja poistaa projektidataa, ja luettua dataa voidaan hyödyntää myös muissa projekteissa. Itse rajapinnan tarjoamia funktioita on satoja, ja ne on kaikki dokumentoitu Siemensin API-dokumentteihin, ja ne ovat vapaasti ladattavissa valmistajan kotisivuilta. (TIA Portal Openness: Introduction and Demo Application, 7-8)

Openness-rajapinnan ”import/export”, eli objektien vienti ja tuonti, perustuu niiden muuntamiseen XML-formaattiin ja XML-formaatista takaisin objektiksi. Tätä toimenpidettä voidaan hyödyntää muun muassa toimilohkoihin (FB), funktioihin (FC), tietueisiin (DB), verkkotopologiaan ja lukuisiin muihin komponentteihin. Myös kirjastokomponentteja, niin tyyppikirjastoissa kuin ”master”-kirjastoissa voidaan rajapintaa hyödyntäen tuoda projektiin, sekä viedä projektista. (TIA Portal Openness: Introduction and Demo Application, 28-29)

Rajapinnan kautta voidaan myös käskyttää itse TIA Portal -ympäristöä. Ympäristö voidaan avata rajapinnan kautta, ja siihen voidaan avata joko olemassa oleva projekti, tai luoda kokonaan uusi projekti. Ympäristö voidaan avata joko ilman graafista käyttöliittymää tai käyttöliittymän kanssa. (TIA Portal Openness: Introduction and Demo Application, 24-26)

Dokumentaation lisäksi Siemens on kehittänyt joitakin esimerkkisovelluksia, Openness-rajapinnan funktioiden havainnollistamiseksi. Näitä esimerkkisovelluksia on ladattavissa Siemens Support -nettisivuilta lähdekoodeineen. (TIA Portal Openness: Introduction and Demo Application, 9)

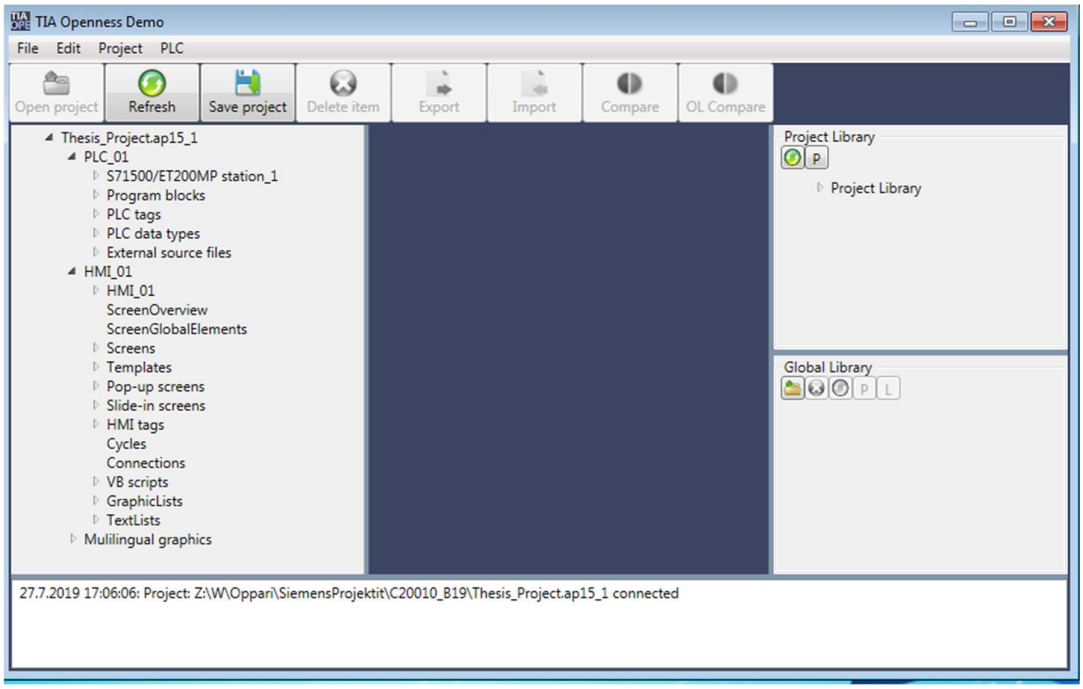
Yksinkertaisin esimerkkisovellus, ja ensimmäinen mihin rajapintaa tutkiessa kannattaa perehtyä on ”StartOpenness”-sovellus. Lähdekoodi on pidetty yksinkertaisena, ja siihen on tuotu helppoja esimerkkejä rajapintafunktioiden käytöstä yksinkertaisissa toimenpiteissä, kuten TIA Portal -käynnistys, projektin tallennus, projektin avaus sekä projektin sulkeminen. Sovelluksella on myös mahdollista määritellä ja nimetä laite, joka voidaan generoida käyttöliittymästä kohdeprojektin laitekonfiguraatioon. Alla kuva sovelluksen käyttöliittymästä. (Kuva 4) (TIA Portal Openness: Introduction and Demo Application, 14-19)



Kuva 4. StartOpenness-sovelluksen käyttöliittymä. (TIA Portal Openness: Introduction and Demo Application, 14)

Toinen Siemensin tuottama esimerkkisovellus on “TIA Openness Demo”, joka sisältää jo valtaosan toiminnoista Openness-rajapinnassa. Lähdekoodi on huomattavasti monimutkaisempaa ja laajempaa kuin edeltävässä esimerkkisovelluksessa, ja sen ymmärrys vaatii ainakin perustiedot olio-ohjelmoinnista C#-kielellä. Tämäkin sovellus sisältää graafisen käyttöliittymän, jonka kautta voidaan avata ja tarkastella TIA Portal -projektihakemiston sisältöä, sekä muokata sitä.

Tähän sovellusesimerkkiin perehtymällä on mahdollista saada jo todella kattavan kuvan Openness-rajapinnasta ja sen käytöstä sovelluksissa. Alla on esitetty kuva TIA Opennes Demo -sovelluksen käyttöliittymän pääsivusta (Kuva 5). (TIA Portal Openness: Introduction and Demo Application, 20-33)



Kuva 5. TIA Openness Demo käyttöliittymän pääsivu.

4.4 Openness ja kirjastointi

Openness-rajapinta tarjoaa myös useita kirjastointiin liittyviä toimintoja. Rajapinnan kautta voidaan päivittää ja luoda kirjastoja sekä sisällyttää kirjastokomponentteja projektiin. (TIA Portal Openness: Introduction and Demo Application, 30)

Kirjastokomponenttien tuonti ja vienti projektiin onkin Openness-rajapinnan kautta huomattavasti helpompaa kuin lohkojen luonti. Käytännössä jos käyttäjä haluaa luoda lohkon, joka ei ole standardilohko, pitää ensin kirjoittaa tai tuoda projektista XML-tiedosto, joka pitää sisällään kaikki lohkon komponentit, muuttujat ja funktiokutsut. Openness-rajapinnan kautta ei ole mahdollista generoida yksittäisiä ohjelmarivejä TIA Portal projektin -ohjelmalohkoon, vaan ainoastaan kokonainen lohko kerrallaan projektihakemistoon. (TIA Portal Openness: Introduction and Demo Application, 30)

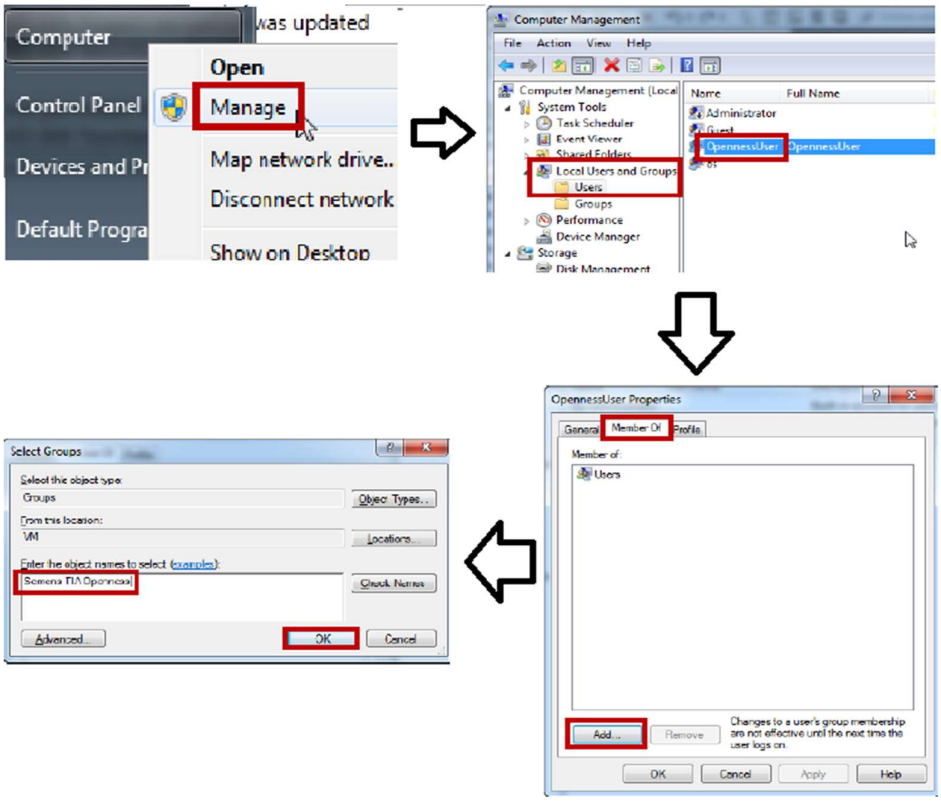
Käytettäessä kirjastokomponentteja, on Openness-rajapintaa hyödyntäen mahdollista tuoda rajaton määrä komponentteja halutuilla nimillä projektiin, ja näin generoida par-

haassa tapauksessa 90 prosenttia projektista automaattisesti. Tämä kuitenkin vaatii onnistuakseen PLC-ohjelmistoarkkitehtuurilta täydellistä standardilohkojen käyttöä joka paikassa. (TIA Portal Openness: Introduction and Demo Application, 30)

4.5 Openness-rajapinnan käyttö ohjelmoinnissa

Openness-rajapintafunktioita voidaan hyödyntää ohjelmakoodissa, ja valittavina ohjelmointikielinä ovat Microsoftin C# ja VB.net ohjelmointikielet. Ohjelmointiympäristönä voidaan käyttää ainakin Microsoft Visual Studio integroitua kehitysympäristöä, ja C#-ohjelmointikielen ollessa kyseessä myös Visual Studion tarjoamaa graafista käyttöliittymän suunnittelutyökalua. (SIMATIC Openness: Automating creation of projects, 25)

Jotta rajapintaa on mahdollista käyttää, tulee Windowsin käyttäjätilien asetuksista antaa haluamalleen käyttäjätilille oikeudet siihen. Käytännössä tämä tapahtuu käyttäjätilien asetuksista, ”Local Users and Groups” valikosta. Valikosta valitaan ”OpennessUser” niminen ryhmä, ja lisätään haluttu käyttäjätili ryhmään. Lopuksi tulee käyttäjän kirjautua vielä uudelleen käyttäjätilille muutosten voimaan astumiseksi. Kuva käyttöönottoprosessista alla (Kuva 6). (TIA Portal Openness: Introduction and Demo Application, 10-11)



Kuva 6. Openness-käyttäjätilin määrittäminen Windowsissa.

Ensimmäistä Openness-projektia luotaessa, pitää Visual Studio -ympäristössä vielä tuoda oikeat referenssit projektiin. ”Siemens.Engineering.dll” niminen rajapintatiedosto löytyy Siemensin asennuskansioista, ja se tulee asettaa referenssiksi projektiin, jossa halutaan Openness-rajapintafunktioita käyttää. Näiden toimenpiteiden jälkeen Openness-rajapinnan tulisi olla hyödynnettävissä, ja rajapintaluokkien tuotavissa ohjelmakoodiin. (TIA Portal Openness: Introduction and Demo Application, 12)

Kun käyttäjän luomaa sovellusta käytetään ensimmäisen kerran, ilmoittaa TIA Portal -ympäristö ponnahdusikkunalla käyttäjälle, että ulkopuolinen sovellus yrittää ottaa yhteyttä siihen. Valmiin sovelluksen kohdalla, kysyy TIA Portal tätä vain kerran, mutta ajettaessa sovellusta Visual Studio -ympäristössä kysyy se tätä jokaisella ohjelman ajokerralla. Sovelluksen toiminta edellyttää, että sillä on lupa yhdistää TIA Portal -ympäristöön. (TIA Portal Openness: Introduction and Demo Application, 13)

5 OPENNESS SCRIPTER

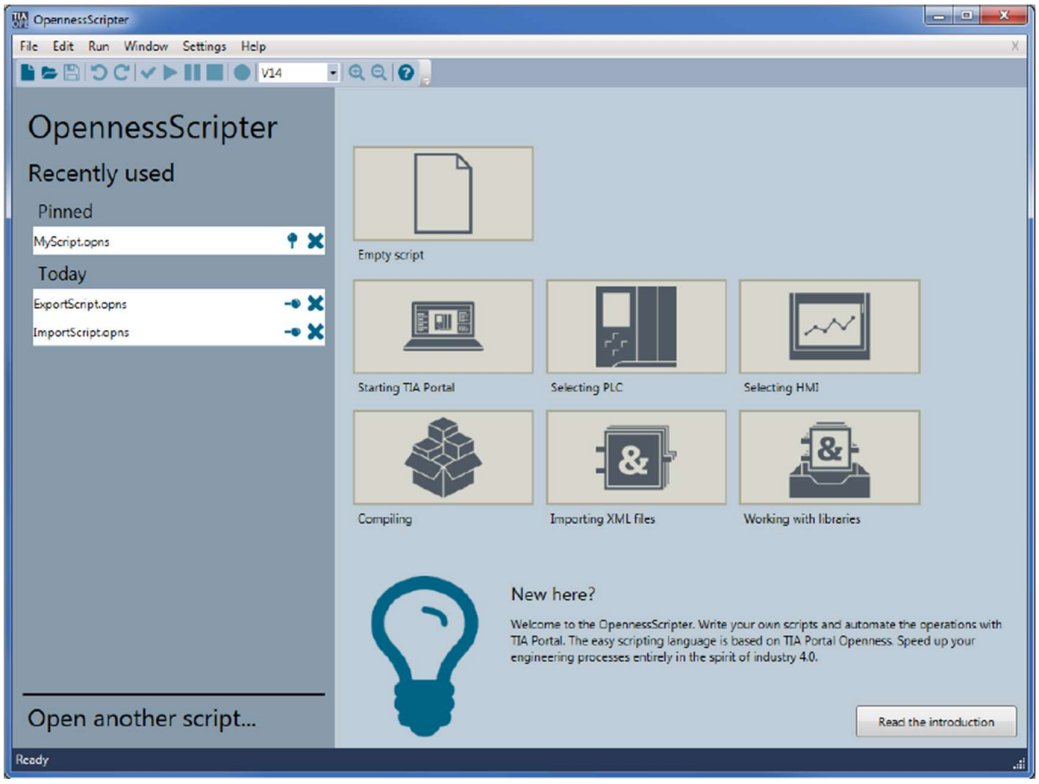
5.1 OpennessScripter apuna Openness-rajapinnan hyödyntämiseen

OpennessScripter on Siemensin tekemä apuväline Openness-rajapinnan toiminallisuuksien käyttöön. Scripter on yksinkertaisuudessaan Openness-rajapintaa hyödyntävä sovellus, joka tekee rajapinnan toiminallisuuksien käyttämisestä huomattavasti helpompaa. Scripteriä voidaan käyttää kolmella eri tavalla: Windows DOS -komentojonolla, Scripterin omaa kehitysympäristöä hyödyntäen tai vaihtoehtoisesti linkittää se TIA Portal ”external applications”-kirjastoon. (OpennessScripter: Introduction, 4)

Scripter helpottaa Openness-rajapinnan käyttöä, koska monimutkaisen ohjelmoinnin ja mittavien ohjelmistorakenteiden sijaan rajapintaa voidaan käskyttää käyttäen yksinkertaisia käskylauseita, eli metakieltä. Tämä vähentää ohjelman kirjoitustyötä merkittävästi, josta esimerkkinä Siemensin omassa dokumentissa 3016 kirjainta sisältävä koodi on voitu lyhentää Scripterin ansioista 115 kirjaimeen, toisin sanoen neljään ohjelmariviin. Tässä tapauksessa Scripterin käyttö poisti siis 95 prosenttia kirjoitustyöstä. (OpennessScripter: Introduction, 6)

Scripter on vapaasti ladattavissa Siemensin kotisivuilta, ja se on täysin ilmainen sovellus. Scripter vaatii toimiakseen käytännössä samat edellytykset kuin Openness-rajapinta, eli käyttäjätili tulee tässäkin tapauksessa olla lisättynä ”Siemens Openness User”-ryhmään. Kun ohjelma on asennettu, voidaan aloittaa skriptin kirjoittaminen. (OpennessScripter: Introduction, 7-8)

Scripterin käyttöliittymässä on valittavissa kuusi eri esimerkkipohjaa skriptille, joihin on valmiiksi kirjoitettu tiettyjä toiminnallisuuksia käytön aloitusta helpottamaan ja havainnollistamaan. Näitä ovat esimerkiksi TIA Portal -ympäristön käynnistys, XML-tiedostojen tuonti projektiin sekä HMI:n valinta. Valittavana on myös tyhjä skriptipohja, jossa käyttäjä pääsee aloittamaan puhtaalta pohjalta skriptin tekemisen. Alla kuvassa esitetty OpennessScripter-päänäkymän käyttöliittymä (Kuva 7). (OpennessScripter: Introduction, 9-10)



Kuva 7. OpennessScripter-päänäkymän käyttöliittymä. (OpennessScripter: Introduction, 9)

Skriptien ajaminen tapahtuu Scripterin käyttöliittymän kautta, ja valmiita skriptejä on mahdollista tallentaa muistiin myöhempää käyttöä varten.

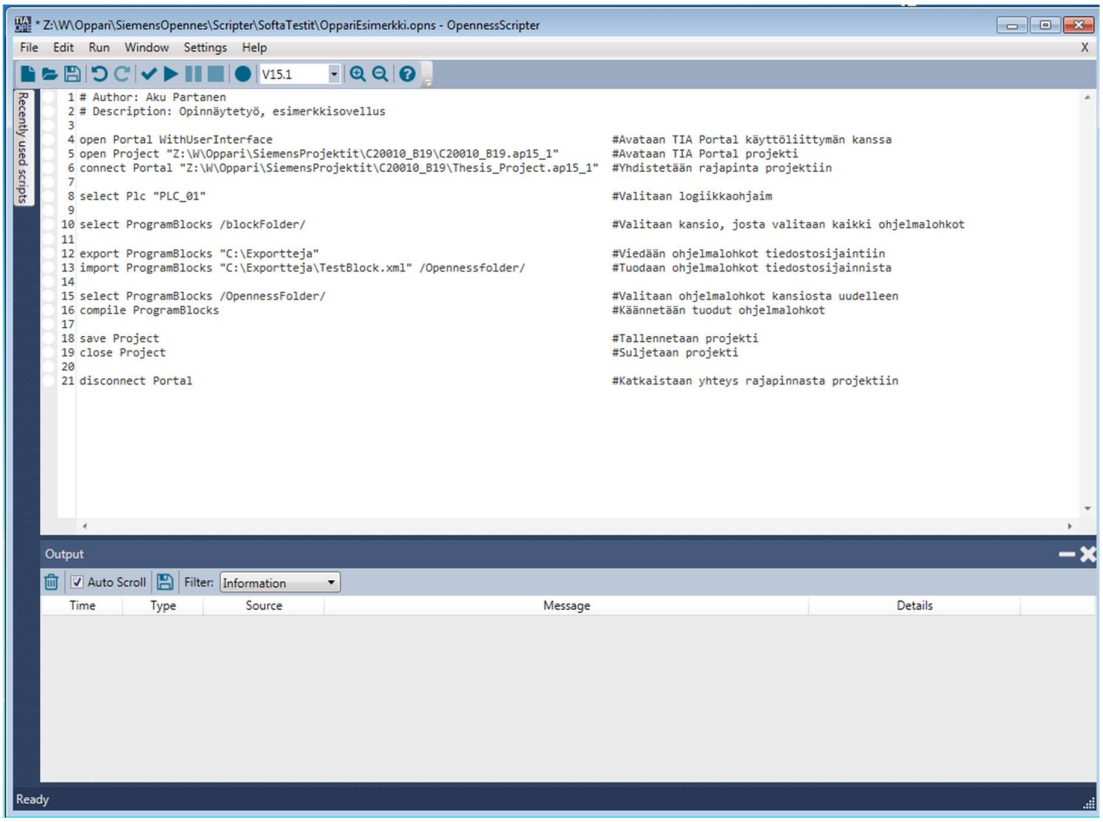
5.2 Sovelluksen ominaisuudet

Käytännössä Scripteriin on sisällytetty suurin osa Openness-rajapinnan tarjoamista toiminallisuuksista. XML-tiedostoja voidaan viedä ja tuoda, TIA Portal -ympäristöä voidaan tietyiltä osin hallita, kirjastoja voidaan päivittää ja kopioida, grafiikoita voidaan viedä ja tuoda, sekä paljon muuta.

Käytännössä Scripter on todella hyvä apuväline Openness-funktioiden kokeiluun sekä yksinkertaisten, käskynomaisten toimintojen ajamiseen TIA Portal -ympäristössä. (OpennessScripter: Detailed Documentation, 5)

Yksi mielenkiintoisimmista ominaisuuksista tätä opinnäytetyötä ajatellen on OpennessScripterin kyky tuoda XML-formaatissa olevia tiedostoja TIA Portal -projektiin, ja generoida niistä ohjelmalohkoja. Myös ohjelmalohkojen vienti projektista XML-formaattiin tiedostoksi on mahdollista ohjelmalohkon ohjelmointikielestä riippumatta. Nämä ominaisuudet mahdollistavat käytännössä ohjelmalohkon muokkaamisen TIA Portal -ympäristön ulkopuolella, joka taas on perusedellytys logiikkaohjelmoinnin automatisoinnille. (OpennessScripter: Detailed Documentation, 23-25)

Jotta ohjelmalohko voidaan viedä XML-formaatissa ulos TIA Portalista, pitää rajapinnan tietää mikä lohko on kyseessä, eli lohko pitää ensin valita. Kun haluttu lohko on valittu, voidaan se viedä määriteltyyn tiedostosijaintiin käyttäjän tietokoneelle. XML-formaatissa olevan ohjelmalohkon tuonti taas tapahtuu asettamalla funktiolle parametriksi lähde tiedostopolku, sekä kohdekansio, johon se projektissa tuodaan. Esimerkki tämän tempun tekevästä skriptistä kuvassa alla (Kuva 8). (OpennessScripter: Detailed Documentation, 23-25)



Kuva 7. Esimerkki skripti OpennessScripter-sovelluksessa, ohjelmalohkon vienti ja tuonti.

5.3 Sovelluksen rajoitukset

Yksinkertaisuudestaan johtuen Scripter ei sovellu vaativampien sovellusten kokonaisvaltaiseen kehittämiseen. Scripterin tarkoitus on nimenomaan toimia alustana käskylistalle, eikä sillä voida toteuttaa itsenäisesti minkäänlaisia ohjelmistorakenteita, kuten ehtolauseita, silmukkarakenteita, tai vastaavia toimintoja.

Scripterillä voidaan tehdä ainoastaan käskylistoja, joita voidaan suorittaa systemaattisesti järjestyksessä ylhäältä alas, ja joissa ohjelman suoritus on mahdollista saattaa aina loppuun, eikä virheitä esiinny. Sovellukset, jotka vaativat ehtorakenteiden käyttöä, tietojen ja tiedostojen käsittelyä, muokkausta tai muuta vastaavaa TIA Portal -ympäristöstä erillään olevaa toimintaa eivät Scripterillä yksinään ole toteutettavissa, ja näin ollen vaatii se ulkoisen sovelluksen tuekseen. (OpennessScripter: Detailed Documentation, 13-14)

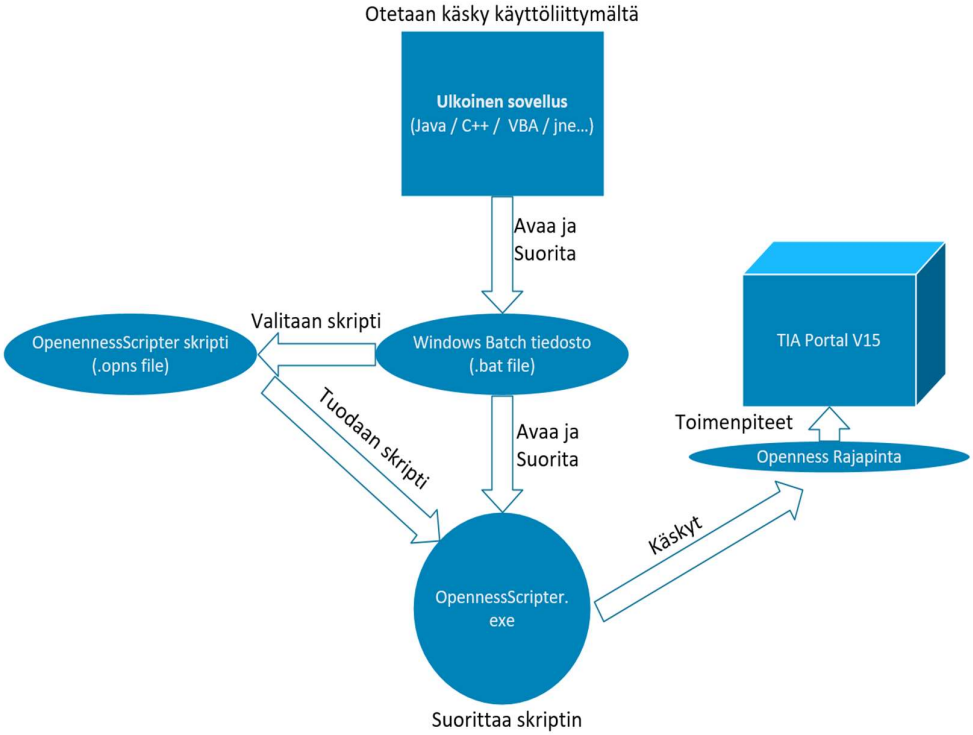
Scripteriä on mahdollista käyttää ulkoisen MS-DOS komentojono ohjelman kautta, eli Windows batch (.bat)-tiedostoformaatissa tallennettua ohjelmaa ajamalla. Tämä ominaisuus mahdollistaa Scripterin käskyttämisen ulkopuolelta, joka mahdollistaa skriptien automaattisen ajamisen, ja näin ollen muiden toiminallisuuksien, kuten ehtolauseiden ja rakenteiden käytön ulkoisen ohjelman toteuttamana. (OpennessScripter: Detailed Documentation, 10)

5.4 OpennessScripter sovelluksen osana

Kuten edellisen kappaleen lopussa mainittiin, on OpennessScripter sovellusta mahdollista käyttää komentojonolla. Komentojonoon pitää määritellä tiedostopolku, jossa OpennessScripter-sovellus sijaitsee sekä skripti mitä Scripterissä halutaan ajaa. Nämä kaksi argumenttia riittävät skriptin ajamiseen, mutta näiden lisäksi käyttäjä voi halutessaan lisätä muuta toiminallisuutta komentojonoon, kuten tulostuksia, ehtorakenteita, muuttujia ja muuta komentojonon mahdollistamaa toimintaa. (OpennessScripter: Detailed Documentation, 10)

Komentojonon, eli Windows batch -tiedoston käyttäminen mahdollistaa ulkoisen sovelluksen yhdistämisen Scripteriin, ja se tuo Scripterin käyttöön uusia ominaisuuksia, kuten tietojenkäsittelytoimintoja, graafisen käyttöliittymän mahdollisuuden, ja muita hyödyllisiä ominaisuuksia joita ylemmän tason ohjelmointikielet mahdollistavat.

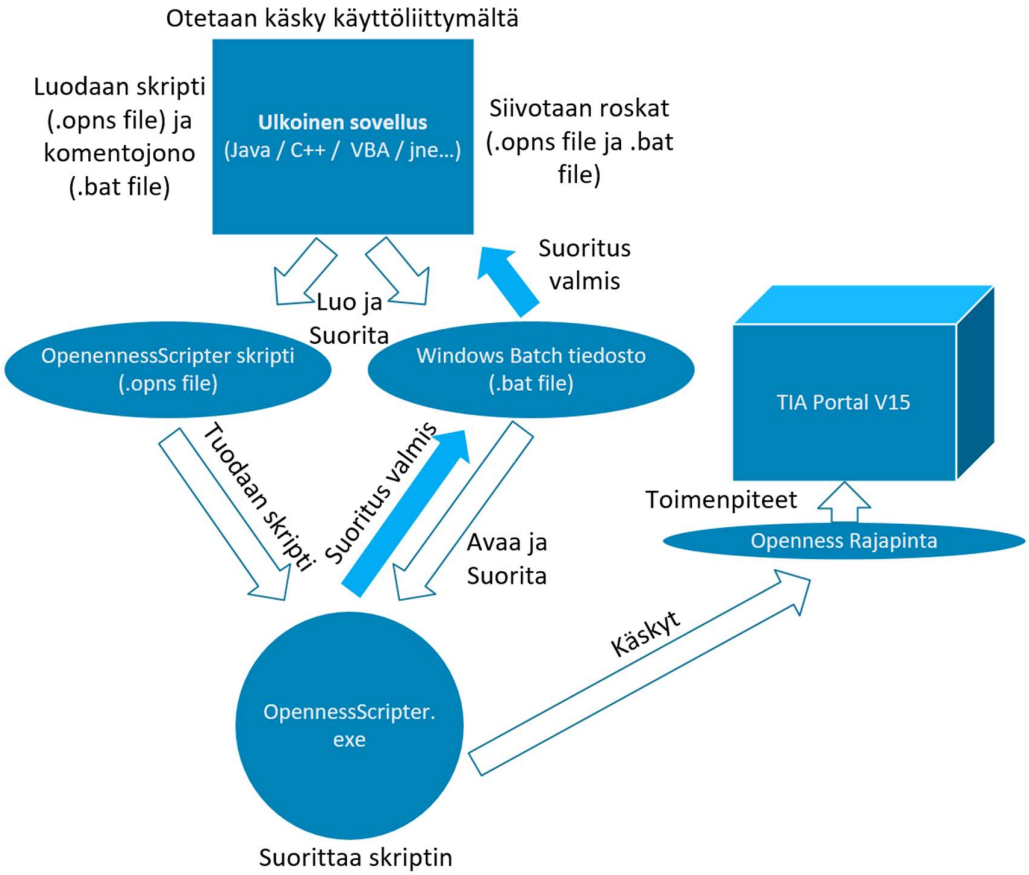
Esimerkkinä sovellus voisi toimia seuraavasti: Ulkoinen sovellus, joka on ohjelmoitu käyttäjän määrittelemällä ohjelmointikielellä (esimerkiksi Java) avaa Windows batch -tiedoston, johon on sisällytetty tarvittavat tiedostopolut ja rakenteet. MS-DOS -komentojono avaa OpennessScripterin ja syöttää batch-tiedostossa määritellyn skriptin sille. Scripter suorittaa halutut komennot TIA Portal -ympäristössä. Ohessa toimintaa havainnollistava kuva (Kuva 8).



Kuva 8. OpennessScripter sovelluksen osana, periaatekaavio.

Edellä kuvattu toimintamalli ei vielä itsessään tarjoa paljon joustoa ohjelmiston toimintaan johtuen tiedostopolun staattisesta määrittelystä batch-tiedostossa, ja skriptin staattisesta rakenteesta OpennessScripter (.opns) tiedostossa. Siksi kyseiset kaksi tiedostoa onkin hyvä luoda väliaikaistiedostoiksi ulkoisen sovelluksen toimesta, ja tällöin

sekä tiedostopolut, että itse skriptin toiminta voidaan määritellä aina tapauskohtaisesti sovellusta suorittaessa, eikä suorituksen jälkeen synny käyttäjän koneelle ”roskia”. Toiminta havainnollistettu alla kuvassa (Kuva 9).



Kuva 9. OpennessScripter sovelluksen osana, ajonaikainen tiedostojen luonti ja roskien siivous.

6 OHJELMISTOKEHITYKSEN TYÖKALUT

6.1 Microsoft Visual Studio

Microsoft Visual Studio on Microsoftin kehittämä ja ylläpitämä integroitu ohjelmiston kehitysympäristö, eli IDE (Integrated Development Environment). Visual Studio sisältää monipuoliset ammattitason kehitystyökalut ja –palvelut minkä tahansa alustan sovelluskehitykseen, kuten Windows, iOS, Android, verkko ja pilvi ympäristöille. Visual Studiolla on mahdollista suunnitella ja toteuttaa monimutkaisia sovelluksia, jakaa koodia ja seurata töitä. (Ammattitason ohjelmistot)

Visual Studio ei ole ohjelmointikieliriippuvainen IDE, vaan sitä voidaan käyttää monilla eri ohjelmointikielillä sovelluksia ohjelmoidessa. Se tukee yli 36 eri kieltä, joita esimerkkinä ovat C#, C++, VB (Visual Basic), Python, JS (JavaScript) ja monia muita kieliä. Visual Studio on saatavissa Windows- ja MacOS-käyttöjärjestelmille. (Introduction to Visual Studio)

Ensimmäinen versio Visual Studiosta on julkaistu jo vuonna 1997, jonka versio numero oli 5.0 ja se tarjosi vielä täyden tuen Java-kielellä ohjelmointiin, toisin kuin uudemmat versiot, 2017-versiosta lähtien. Tämän hetken uusin versio on Visual Studio 2019. (Introduction to Visual Studio)

Visual Studiosta on saatavilla kolme eri julkaisua, joista Community on ainoa ilmainen versio. Visual Studio Professional ja Enterprise ovatkin tarkoitettuja enemmän yritysten ja työelämän käyttötarkoituksiin, kun taas Community kevyempään harrastekäyttöön. Pelkästään Community versio on kehitysympäristöksi jo melko raskas ohjelmisto, ja se kykenee jokseenkin vaativampaankin sovelluskehitykseen. Sekä Professional että Enterprise-julkaisuissa kummassakin on määräaikainen ilmainen kokeilujakso, jonka jälkeen niiden käyttö muuttuu maksulliseksi. (Introduction to Visual Studio)

Ladatessa Visual Studiota käyttäjä saa valita, mitä kielipaketteja mukana asennetaan. Kukin paketti vie muutamasta sadasta megatavusta kymmeniin gigatavuihin tilaa tietokoneen muistista, joten aluksi kannattaa asentaa vain ne paketit, joita oikeasti voidaan tarvita. (Introduction to Visual Studio)

6.2 Olio-ohjelmointi

Olio-ohjelmointi perustuu itsenäisten olioiden yhteistoimintaan, ja se ei ole pelkästään ohjelmointia, vaan myös ajatusmalli. Olio on tietorakenne, jolla pyritään kuvaamaan reaalimaailman asioita ohjelmoinnissa selkeällä ja ymmärrettävällä tavalla. Ne pitävät sisällään tilatietoa, sekä niillä on oma käyttäytymismalli. (Peltomäki 2014, 66)

Olio-ohjelmoinnin ideana on rakentaa sovellus itsenäisistä, toisistaan riippumattomista palasista eli olioista, joilla on tietyt vastuualueet ja toiminnot, joita niiden tulee suorittaa. Olio-ohjelmointi mahdollistaa ohjelmistokehityksen jakamisen siis pieniin palasiin, ja näin helpottaa ohjelmointityön jakamista useammalle henkilölle. Tässä tapauksessa ollaan kiinnostuneita tyypillisesti vain olion tarjoamasta julkisesta käyttörajapinnasta. (Peltomäki 2014, 66)

Ennen kuin olioita voidaan alkaa määritellä, on oltava olemassa luokka. Oliota voidaan synnyttää luokista, periaatteessa rajaton määrä. Jokainen olio siis kuuluu aina yhteen luokkaan, ja jotkut luokat voivat taas koostua toisen luokan olioista. Luokka siis kuvaa saman tyyppisten olioiden joukkoa, josta esimerkkinä voidaan ajatella käsitettä ”auto” luokkana. Autoja voi olla monen merkkisiä ja mallisia, ja ne voidaan ajatella olioina. Esimerkiksi luokasta ”auto” voidaan synnyttää kolme oliota, Toyota, Saab, Nissan. Nämä ovat luokan ilmentymiä, eli olioita. (Peltomäki 2014, 66)

6.3 C#-ohjelmointikieli

C#, puhekielessä C charp on moderni, olio-orientoitunut korkeamman tason ohjelmointikieli, ja sitä kehitettäessä on yhtenä tavoitteena ollut yhdistää MS Visual Basicin tuottavuus ja C++:n tehokkuus. Kielenä se on symbioottinen .NET-alustan kanssa, ja suurin osa .NET-luokkakirjastoista on kehitetty kieltä käyttäen. (Sivonen 2004, 1)

C++-kielestä poiketen, pointterit eli osoitintyypit eivät varsinaisesti ole C#-kielen perustyyppejä, mutta ne ovat silti sisällytettynä kieleen C/C++ yhteensopivuuden takaamiseksi. Monelta osalta ja ulkoasultaan C# muistuttaa Javaa, mutta poikkeuksen tästä tekevät esimerkiksi sijoitus- ja ehto-operaattorit. (Sivonen 2004, 1-2)

Ohjelma C#-kielessä koostuu järjestetyistä Unicode-sarjoista, eli lähdetiedostoista. Kielessä varatut sanat kirjoitetaan Javan ja muiden C-kielien tapaan pienillä kirjaimilla ja isot ja pienet kirjaimet tulkitaan eri merkeiksi. Jossain määrin C# tukee C++:n kaltaisesti esiprosessikäskyjä, mutta erona voidaan mainita esimerkiksi esiprosessien suoritus vasta kielioppia tarkistaessa. (Sivonen 2004, 1-2)

C#-kielisiä ohjelmia voidaan kirjoittaa millä tahansa tekstieditorilla, ja valittavina on laaja kirjo erilaisia editoreja. Osa tekstieditoreista on suunniteltu varta vasten ohjelmointia ajatellen, ja esimerkiksi aikaisemmassa kappaleessa esitelty Microsoft Visual Studio on C#-kielen ohjelmointiin monipuolinen työkalu. Tämän lisäksi ohjelmoijien suosimia työkaluja ovat Notepad++, Vim ja ConTEXT. (Hyvönen, lappalainen & Lakanen 2013, 3)

Ohjelman kääntäminen ja ajaminen vaativat onnistuakseen jonkun C#-sovelluskehittimen, eli kääntäjän. Visual Studiossa tällainen kääntäjä ominaisuus on itsessään, ja ohjelman testaaminen ympäristössä on suhteellisen yksinkertaista. (Hyvönen ym. 2013, 3)

6.4 Microsoft Excel

Excel on Microsoftin kehittämä, graafinen taulukkolaskenta ja tietojen analysointi sovellus. Excel sisältää myös erikoisgrafiikkaominaisuuden, tietokantaominaisuuksia sekä sillä on mahdollista ohjelmoida makroja käyttäen Visual Basic for Application (VBA) –ohjelmointikieltä. (Keinonen 2010, 8)

VBA:lla on mahdollista ohjelmoida makroja, jotka suorittavat esimerkiksi usein käytettyjä toimintosarjoja, kuten tietojen muokkausta ja analysointia helposti vain yhdellä

klikkauksella. VBA on myös olio-orientoitunut kieli, eli se taipuu myös monimutkaisempien oliorakenteisten ohjelmien kirjoittamiseen, ja sillä voidaan myös tehdä esimerkiksi omia laskentafunktioita. (Keinonen 2010, 8)

Taulukkolaskenta on hyödyllinen metodi, kun tuotetaan usein toistuvia laskentamalleja, esimerkiksi budjetointia, laskutusta sekä muuta vastaavaa toimintaa. Taulukkolaskentaohjelma, kuten Excel, on apuväline laskennassa, ja ruutupaperimaisesti Excelin laskenta-arkki jakautuu riveihin ja sarakkeisiin. Laskenta toteutetaan yhdistämällä erilaisia tietojoukkoja toisiinsa, ja Excelin avulla niistä voidaan myös tehdä graafisia kuvaajia eli diagrammeja. (Keinonen 2010, 8)

Kuten edellä mainittu, Exceliin on mahdollista kirjoittaa makroja. Makrot ovat automaattisia toimintosarjoja, ja niitä voidaan ohjelmoida Exceliin joko nauhoittamalla omaa toimintaansa, ohjelmoimalla perinteisesti tai näiden yhdistelmällä. Makroja sisältävät Excel-dokumentit pitää tallentaa oikeaan muotoon makrojen toimivuuden takaamiseksi, eli tallennettaessa ensimmäistä kertaa tulee muistaa asettaa dokumentti ”makro enabled”-muotoon. (Keinonen 2010, 116)

6.5 Excel-ohjelmointi ja VBA

Excel-ohjelmointi, toisin sanoen makrojen tekeminen tapahtuu Microsoft Excel-sovelluksella ”Developper”-välilehdeltä. Makrokielenä Excel käyttää VisualBasic for Applications, eli lyhyemmin VBA-ohjelmointikieltä. (Keinonen 2010, 116)

Makrojen ohjelmointi on tapa automatisoida Excelin käyttöä. Excelin lisäksi VBA-ohjelmointikieltä on mahdollista käyttää Microsoftin Access, Power Point ja Word-dokumenttien makrojen kirjoittamiseen, ja näin automatisoida haluttuja toimintoja. (Taanila 2013, 1)

VBA on Microsoftin 90-luvulla kehittämään Visual Basic -kieleen perustuva, sovelluksille tarkoitettu ohjelmointikieli. Perinteisten Visual Basic -kielen ominaisuuksien lisäksi, voidaan siihen sisällyttää sovelluskohtaisia funktioita ja tietojen analysointia, esimerkiksi Excelissä solukohtaisia viitauksia ja toimintoja. Tästä esimerkkinä Range-

luokan funktiot, joilla voidaan käsitellä tietyn alueen sisällä olevia soluja ja niiden tietoja, tai esimerkiksi hävittää ne. (Lesson one: What is VBA?)

Visual Basic hyödyntää komponenttipohjaista ohjelmointitapaa, ja tukee myös olio-pohjaista toteutusmallia, vaikkei se täydellinen olio-ohjelmointikieli olekaan. Visual Basicista puuttuu esimerkiksi olio-kielille tyypillinen periytyminen. Komponenttien hyödyntämisessä Visual Basic kuitenkin mahdollistaa käytön mille tahansa COM-rajapintaa tukevien komponenttien käytölle. Javasta ja tavallisesta Basic-kielestä poiketen Visual Basic ei ole tulkattavissa, vaan se käännetään suoraan, kuten C-kieli. (Vitikka 2000, 1)

6.6 XML-merkintäkieli

XML on World Wide Web Consortiumin kehittämä merkintäkieli tai standardi, joka on tarkoitettu käytettäväksi formaatiksi tiedonvälitykseen eri järjestelmien välille ja dokumenttien tallentamiseen. XML lyhenne tulee englannin kielen sanoista ”extensible markup language”, ja sen tarkoituksena on selkeyttää laajojen tietomäärien jäsentelyä. (web-opas, Mikä on XML?)

XML-muotoisten dokumenttien käsittelyyn on tarjolla paljon erilaisia työkaluja, ja mainittakoon että Microsoft Excel tukee myös XML-formaatin käsittelyä taulukoissa sekä VBA-sovellusten kehittämisessä. Työkalut eivät ota kantaa kuvaako XML-dokumentti WWW-sivua, vai jotain muuta XML-formaatissa tallennettua dokumenttia, ja toimivat näin ollen samalla tavalla. (XML-tietojen tuominen, ei pvm), (web-opas, Mikä on XML?)

Ulkoasultaan tekstimuotoinen XML-kieli muistuttaa HTML-kieltä, jolla kirjoitetaan WWW-sivuja. Kuitenkin HTML:stä eroten, XML ei ole tarkoitettu internet sivunkuvauskieleksi, vaan sillä kuvataan tiedon rakennetta ilman ennalta määrättyjä koodeja, ja sillä voidaan muodostaa uusia koodeja, joilla luoda dokumentteja erinäisiin käyttötarkoituksiin. (web-opas, Mikä on XML?)

Ohjelmoitavien logiikoiden maailmassa XML on tullut tutuksi PLCopen-organisaation luotua XML-standardin IEC 61131-10. Siinä missä IEC 61131-3-standardi määrittelee PLC-ohjelmoinnissa käytettävät ohjelmointikielet, määrittelee tämä standardi yhteisen rakenteen XML-dokumenteille, joilla näitä standardiohjelmalohkoja ja projekteja voidaan siirtää logiikkavalmistajan ympäristöstä toisen valmistajan ympäristöön. (PLCopen, XML Exchange)

Monet valmistajat, kuten Siemens eivät tätä IEC 61131-10:n määrittelemää toimintatapaa vielä kuitenkaan noudata, ja esimerkiksi Siemens on kehittänyt oman XML-skeeman ohjelmalohkojen vientiin ja tuontiin. Siemensin tapauksessa tämäkin toimenpide onnistuu ainoastaan Openness-rajapinnan kautta, ja on käännettävissä ainoastaan valmistajan omassa TIA Portal -ympäristössä ohjelmakoodiksi. (Siemens: How is XML file structured for blocks?)

6.7 MS-DOS komentojonot

DOS-komentojono on komennoista, eli tekstistä koostuva tiedosto, joka tavallisesti nimetään päätteellä .BAT, eli tallennetaan Batch-tiedostoformaattiin. Komentojonotiedostossa jokainen yksittäinen rivi sisältää yhden komennon, joka suoritetaan komentorivillä, ja näistä riveistä koostuvalla tiedostolla voidaan näin ollen ajaa monimutkaisempiakin rutiineja tietokoneella. (Miten komentojono tehdään?)

Komentojonoilla voidaan toteuttaa ja automatisoida usein toistuvia toimintoja, kuten varmuuskopiointia tai vaikka avata ohjelmistoja. Muutamia ohjelmia voidaan myös käyttää komentojonoilla, ja niille on annettavissa useita parametreja, joilla voidaan ohjata ohjelman toimintaa. Esimerkiksi tässä opinnäytetyössä komentojonoa käytetään OpennessScripter.exe-ohjelman avaamiseen, ja sille annetaan parametrina skripti, joka sen tulee suorittaa. (Miten komentojono tehdään?)

Komentojonoon voidaan kirjoittaa myös yksinkertaisia ohjelmistorakenteita, kuten toistolauseita eli ”luuppeja”, käyttää ehtorakenteita, sekä muuttaa ohjelman suorituksen kulkua ”GOTO”-käskyllä. Komentojono voi myös tulostaa käyttäjälle tietoa

”ECHO”-käskyllä, sekä se voidaan laittaa paussille, jolloin se mahdollistaa interaktiivisten ohjelmien tekemisen esimerkiksi yksinkertaisella tekstikäyttöliittymällä. (Komentojonossa käytettävät komennot)

Komentojonoa voidaan käyttää myös sovellusten väliseen tiedonvaihtoon, jos se syystä tai toisesta ei muuten ole mahdollista. Komentorivitiedosto voidaan kirjoittaa ja tallentaa esimerkiksi käyttäen jotain korkeamman tason ohjelmointikieltä, ja myös ajaa tämän sovelluksen suorituksen yhteydessä. Tässä opinnäytetyössä esimerkiksi komentorivitiedosto (.bat) on tärkeässä roolissa sovelluksen ja Siemens OpennessScripterin yhteistoiminnan kannalta, ja kuten edellä mainittiin, se luodaan ajon aikana, sekä myös tuhotaan. Alla kuva esimerkki komentojonotiedostosta. (Kuva 10)

```
1  set exe="C:\Program Files (x86)\Siemens\Automation\OpennessScripter\OpennessScripter.exe"
2  set script="Z:\W\Oppari\SiemensOpennes\Scripter\SoftaTestit\OpenTIA_Portal_Test.opns"
3  echo "working..."
4  start "TIA Openness" /wait %exe% /silent /file:%script%
5  echo "done!"
6  exit
```

Kuva 10. Esimerkki DOS-komentojonotiedosto.

Kuvan esimerkki komentojonossa tehdään vaadittavat asiat OpennessScripter skriptin ajamiseen tietystä tiedostosijainnista. Rivillä yksi ja kaksi asetetaan muuttujiin arvot, mikä ohjelma halutaan ajaa, ja mikä tiedosto sille syötetään parametrina. Muuttujien alustuksen jälkeen käynnistetään haluttu ohjelma ”START” komennolla, ja syötetään parametrina aiemmin alustettu muuttuja, joka pitää sisällään tiedoston. Parametrina annetaan myös käsky odottaa, kunnes ohjelman suoritus on valmis, ja ilmoittaa siitä käyttäjälle. Lopuksi poistutaan komentojonon suorituksesta käskyllä ”EXIT”.

7 PLC-OHJELMISTOARKKITEHTUURI

7.1 Ohjausjärjestelmän rakenne

Ohjattavan järjestelmän laitteet jaetaan käyttötarkoituksiltaan ja ominaisuuksiltaan erilaisiin ryhmiin ja vyöhykkeisiin, näitä ovat hätäpysäytysvyöhyke, turvapysäytysvyöhyke ja käyttötaparyhmät. Ohjausjärjestelmä jaetaan vyöhykkeisiin, jotta niiden käyttäminen, valvonta ja huoltaminen olisi mahdollisimman yksinkertaista ja turvallista ja näin ollen tarkoitukseensa sopivaa. Vyöhykkeiden väliset riippuvuudet ja yhteydet kuvataan ohjausjärjestelmää koskevissa suunnitteludokumenteissa, ja ne laaditaan aluilleen jo projektin tarjousvaiheessa. (Suvela, 3)

Kaksi isointa yksittäistä vyöhykettä ovat hätäpysäytys- ja turvapysäytysvyöhyke. Hätäpysäytysvyöhyke tarkoittaa laitteista koostuvaa kokonaisuutta tai vyöhykettä, jolla on yhteinen hätäpysäytystoiminto ja yksi laite voi kuulua ainoastaan yhteen hätäpysäytysvyöhykkeeseen. Hätäpysäytysvyöhykkeen sisällä voi olla myös erillisiä turvapysäytysvyöhykkeitä, jotka on mahdollista saattaa turvalliseen tilaan ilman koko hätäpysäytysvyöhykkeen kattavaa pysäytystä. (Suvela, 3)

Hätäpysäyttimet voivat olla joko vain vyöhykekohtaisia, tai useamman vyöhykkeen yhteisiä, jolloin pysäytys toiminto ulottuu yhtä vyöhykettä laajemmalle alueelle. Jos hätäpysäytysvyöhykkeellä on turvapysäytys vyöhykkeitä, saa hätäpysäytys myös aikaan turvapysäytyksen näille vyöhykkeille, ja hätäpysäytyksen kuittaamisen jälkeen pitää ne vielä erikseen kuitata omilta vyöhykkeiltään. (Suvela, 3)

Turvapysäytystoiminto on hätäpysäytystä ”lievempi” tapa saattaa tietyt laitteet ja laitekokonaisuuden turvalliseen tilaan, esimerkiksi operaattorin päästämiseksi alueelle. Turvallisuusvyöhykkeen turvalaite pysäyttää kaikki vyöhykkeeseen kuuluvat laitteet, mutta ei vaikuta muihin hätäpysäytysvyöhykkeellä oleviin laitteisiin, jolloin tuotanto voi muulta osin jatkua esimerkiksi lyhyen huoltotyön aikana. Koko tuotantolinja pysähtyy vasta tuotantopysähdyksessä, kun materiaalivirta ei turvapysäytetyn laitteiston

vuoksi enää pääse siirtymään eteenpäin. Tuolloin laitteet pysähtyvät ohjaimen ohjaamana, jolloin ne on helppo saattaa jälleen toiminnalliseen tilaan pysähdyksen jälkeen, ja vältytään näin laajemmilta toimenpiteiltä tuotannon jatkamiseksi. (Suvela, 4)

7.2 Käyttötaparyhmä

Käyttötaparyhmä on ohjausjärjestelmän vyöhykkeistä pienin yksittäinen kokonaisuus, ja niille ominaista on itsenäiset tilat. Yksittäinen automaatiolaite voi kuulua aina vain yhteen käyttötaparyhmään kerrallaan, ja käyttötaparyhmällä on aina yhteiset toimintatilat. Toimintatiloja voi olla esimerkiksi automaattiajo, käsiajo, kotiin-ajo ja seis-tila. Yhden käyttötaparyhmän tilan vaihtuessa esimerkiksi ”seis”-tilaan ei tarvitse koko tuotantoa pysäyttää, vaan muut käyttötaparyhmät pystyvät edelleen toimimaan oman tilansa mukaan. Tämä jaottelu vähentää tuotantopysähdyksiä ja helpottaa niistä toipumista. (Suvela, 6)

Käyttötaparyhmän eri toimintatilat ovat ohjelmallisia ja mahdollistavat sen laitteiden käyttämisen eri käyttötarkoituksiin. Toimintatilat ovat käytössä vain, kun käyttötaparyhmän vyöhykkeen hätäpysäytys ei ole aktiivisena, tai se on kuitattu pois. Toimitilojen valinta voidaan toteuttaa käyttötaparyhmäkohtaisesti esimerkiksi kääntökytkimellä, tai vaikkapa kosketusnäytöllä, jolla on mahdollista asettaa myös kaikki käyttötaparyhmät tiettyyn tilaan. (Suvela, 9)

Perinteisten tilojen lisäksi käyttötaparyhmällä voi olla paljon muitakin hyödyllisiä tiloja, ja yksi yleisimmistä on tuotantopysäytys. Tuotantopysäytys ei pysäytä laitteita välittömästi, vaan odottaa että laitteet ovat turvapysäytykselle otollisessa kohdassa. Laitteet on mahdollista käynnistää jatkuvaan automaattiajoon, kun turvapysäytys on kuitattu. Tuotantopysäytys on suunniteltu niin, että tuotteet jäävät prosessissa sellaisiin paikkoihin, etteivät ne aiheuta vaaraa ja ovat sellaisessa paikassa, mistä on helppo jatkaa automaattiajolla tuotantoa eteenpäin. (Suvela, 10)

Muita, harvinaisempia tiloja ovat ”Tauko”, ”Työkierron lopetus”, ”Ulkoinen pysäytys” ja ”Askellusajo”. Näitä tiloja ohjelmoidaan ohjausjärjestelmään tarpeen mukaan, esimerkiksi asiakkaan vaatimuksesta. (Suvela, 6 - 12)

8 PROJEKTIN LÄHTÖKOHDAT

8.1 Openness-sovelluksen hyödyntäminen ohjelmistokehityksessä

Openness-sovelluksella on tarkoitus tämän työn osalta kyetä automatisoimaan kuljetinkäyttötaparyhmien ohjelmoinnin toistuva työ, joka on yrityksessä tällä hetkellä käsityötä. Automatisoitavaksi työn tekee sen kaavamainen toistuvuus, ja käyttötaparyhmien samankaltainen olemus.

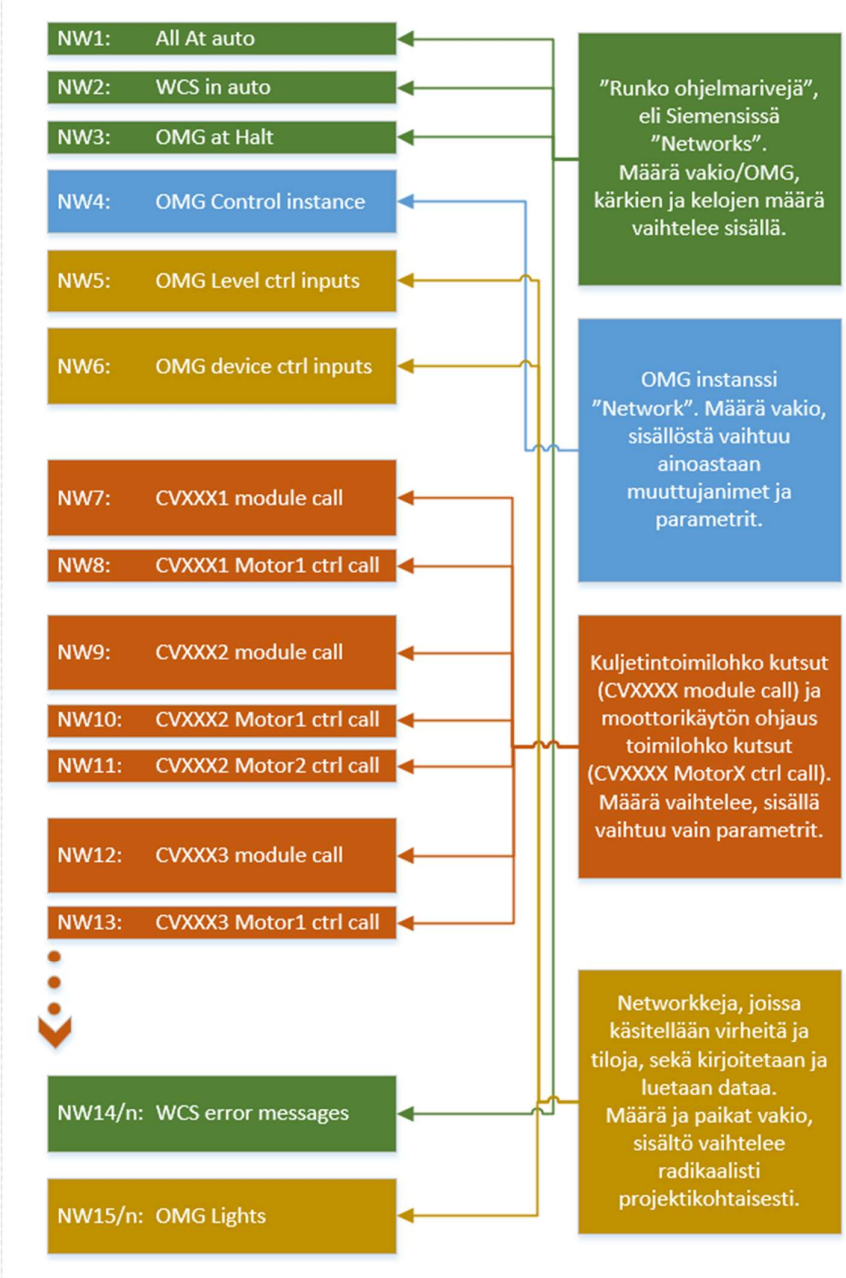
Työn toivottu tulos on helppokäyttöinen, ja koko tiimin käytettävissä oleva sovellus, joka kykenee generoimaan suurimman osan käyttötaparyhmä lohkojen sisällöstä funktiokutsuineen ja alustuksineen valmiiksi projektiin. Ihmisen tekemä työ saatetaan minimiin suunnittelemalla lähdetietodokumentit siten, että kaikki lohkojen generointiin tarvittava tieto on saatavilla ohjelmallisesti. Valmis sovellus halutaan yrityksen edustajan toiveesta liittää osaksi projektidokumenttia, joka sisältää nämä tiedot, mutta silti rakentaa sille erillinen ja selkeä graafinen käyttöliittymä.

Työn täydellinen ja lopullinen toteutus vaatii esimerkiksi sähkösuunnitteluosastolta sen nykyisten toimintamallien kehittämistä ja standardoimista nykyistä paremmalle tasolle, sekä sovelluksen kehittämiseen enemmän resursseja. Tämän vuoksi työ on päätetty rajata sovellusprototyyppiin, jonka ei ole tarkoitus pystyä generoimaan kaikkea ohjelmakoodia, vaan ainoastaan standardiosuus siitä. Työtä tehdessä otetaan kuitenkin huomioon pidemmän aikavälin jatkokehitys ja lopullinen päämäärä täysin automatisoidusta käyttötaparyhmäohjelmoinnista.

8.2 Kuljetinkäyttötaparyhmän rakenne

Työn kannalta yksi merkittävimmistä tekijöistä on yrityksen käyttämä kuljetinkäyttötaparyhmien ohjelmistorakenne. Vaikka yrityksessä PLC-ohjelmistokehitys on jo pitkään ennen uutta Siemens TIA Portal -ympäristön käyttöönottoa perustunut vahvasti kirjastointiin ja standarditoimilohkojen käyttöön, ei samanlaista rakennetta ole vielä kyetty luomaan käyttötaparyhmälohkoille.

Käytännössä yksi kuljetinkäyttötaparyhmälohko pitää sisällään kaikki kyseiseen käyttötaparyhmään kuuluvat kuljetintoimilohkokutsut, moottorikäytön ohjaus toimilohko kutsut, sekä useita ”runko-ohjelmarivejä”. Tällä hetkellä mahdottomaksi kuljetinkäyttötaparyhmälohkon standardoimisen tekee sen vaihtuvan rakenne, jossa vaihtelevat niin kuljettimien määrä käyttötaparyhmittäin, sekä ”runko-ohjelmarivien” elementtien määrä ohjelmarivin sisällä. Kuvassa alla on esitetty kuljetinkäyttötaparyhmän rakenne yksinkertaistettuna. (Kuva 11)



Kuva 11. Kuljetinkäyttötaparyhmän rakenne yksinkertaistettuna.

Kuten edellä olevassa kuvassa käy ilmi, vaihtelee kuljettimien määrä käyttötaparyhmittäin. Kuljetintyypistä riippuen ohjattavien moottorikäyttöjen määrä vaihtelee, yleisimmin yhdestä kolmeen. Ohjelmateknisesti ajatellen yksi kuljetinmoduuli tarvitsee toimiakseen aina yhden kuljetintoimilohkokutsun, ja jokainen moottorikäyttö kuljetinmoduulissa tarvitsee oman moottorinohjaustoimilohkokutsun.

Vihreällä värillä kuvassa (Kuva 11) merkityt ohjelmarivit (Networks) ovat määrältään aina vakioita, mutta niiden sisällä olevien LAD-kielisten kontaktien määrä vaihtelee riippuen kuljettimien määrästä, ja osassa ulkoisen WCS-ohjauksen sisältymisestä riippuen. Keltaisella värillä merkityt (Kuva 11) ohjelmarivit taas ovat vaikeammin automatisoitavissa sovelluksessa, johtuen niiden projektikohtaisista muutoksista, sekä monimutkaisemmasta ohjelmarakenteesta.

8.3 Yrityksen standarditoimilohkokirjasto

Yrityksen standarditoimilohkokirjasto on ollut olemassa jo ennen uuden Siemens TIA Portal -ympäristön käyttöönottoa yrityksen automaatiosuunnittelussa, ja TIA Portal onkin vielä suhteellisen uusi ympäristö käytössä. Ennen TIA Portal -ympäristön käyttöönottoa, kirjastoa pidettiin yllä lähinnä automaatiosuunnittelutiimin toimesta, ja standarditoimilohkoja käytännössä kopioitiin projektista toiseen. Uuden ympäristön tultua, on kirjaston ylläpitoon nimitetty erikseen siitä vastaava toimija yrityksen sisältä, ja myös versionhallinta on näiden muutosten myötä helpottunut.

Standarditoimilohkokirjastoon, eli Siemensin kielellä globaaliin kirjastoon on koottu yleisimpiä kuljetintoimilohkotyyppejä, joista jokainen perustuu niin sanottuun perus kuljetintyyppiin, M100-toimilohkoon. Kirjastoituja toimilohkoja tuodaan projektiin aina tarpeen mukaan, ja projektin niin vaatiessa niiden pohjalta kehitellään variaatioita, jotka taas sisällytetään projektikohtaiseen kirjastoon.

Tällä hetkellä yrityksessä ei ole vielä käytössä Master Copy -kirjastoja, johon periaatteessa olisi mahdollisuus tehdä esimerkiksi projektipohjia. Sen sijaan projektipohjat kopioidaan edelleen vanhoista projekteista, ja tällä tavalla pyritään ohjelmistoprojek-

tien yhdenmukaiseen rakenteeseen, sekä helpottamaan ja nopeuttamaan ohjelmointityötä. Tämä toimintamalli on tosin kyseenalaistettavissa TIA Portal -kirjastoinnin myötä, ja tulevaisuudessa voikin tulla ajankohtaiseksi siirtyä käyttämään tyyppikirjastojen lisäksi myös niin sanottua ”isäntämallia”, eli Master Copy -projektipohjia.

Edellä mainittujen kuljetintoimilohkojen lisäksi, yrityksessä on lukuisia muitakin kirjastoituja objekteja. Näitä ovat esimerkiksi käyttäjän määrittelemät tietotyypit, erilaiset datan käsittelyyn, kirjoittamiseen ja lukemiseen liittyvät funktiot, sekä väyläkommunikaatioon tarkoitetut funktiot. Käyttäjän määrittelemiä tietotyyppejä yrityksessä käytetään paljon, ja esimerkiksi jokaiselle kuljetintoimilohkolle on oma parametritietotyyppinsä, jonka välityksellä parametreja voidaan välittää lohkoon ja lohkosta tallentaa tiedostoihin luettavaksi.

8.4 Käyttötaparyhmän ohjelmoinnin esitiedot ja dokumentit

Jotta käyttötaparyhmien ohjelmointi voidaan automatisoida, pitää tarvittavien esitietojen olla saatavilla ohjelmallisesti, ja formaatissa jossa ne on mahdollista lukea. Tähän mennessä käsin tehtävä käyttötaparyhmäohjelmointi tapahtuu täysin yrityksen ohjelmistopuolen dokumenttien pohjalta, jotka ovat Excel-tiedostossa välilehdillä.

Kyseinen Excel-dokumentti pitää sisällään myös kaikki automaatiosuunnitteluosaston käyttämät Excel-makrot, joiden avulla ollaan esimerkiksi pystytty tuottamaan virhelistoja, IO-listoja sekä tiettyjä lähde-SCL-tiedostoja, joita ollaan voitu makron käyttämisen jälkeen ajaa TIA Portal-ympäristöön ”External Sources”-valikosta, tai IO-listojen ollessa kyseessä ”Tags”-valikosta kohdasta ”Import”.

Makrojen tämänhetkinen toiminta perustuu puhtaasti dokumentin sisältämään tietoon, johon on koottu kaikki kuljetinkäyttötaparyhmien ohjelmointiin vaadittavat tiedot, kuten rajapintaindeksit, kuljetinmoduulityypit, kuljettimien nimet sekä niiltä saatavat virheet. Dokumenttiin myös päivitetään standardimoduulien lisäksi kaikki projektikohtaiset moduulit, eli variaatiot omille välilehdilleen, ja niihin sisällytetään tarvittava tieto esimerkiksi virheistä, joita moduuli kirjoittaa virhetietueeseen.

Kokonaisuudessaan dokumentti on kattava, ja se sisältää käytännössä kaiken tiedon mitä käyttötaparyhmien ohjelmointiin vaaditaan. Kuitenkin jos tämä prosessi halutaan suorittaa ohjelmallisesti, eli automatisoida, tiettyjä asioita pitää vielä lisätä, jotta automatisointi olisi luonteeltaan kannattavaa ja tarkoituksenmukaista.

Haasteen ohjelmallisessa suorittamisessa muodostaa tällä hetkellä kuljetintoimilohkoille syötettävät tulo- ja lähtöparametrit. Näitä parametreja ei tällä hetkellä ole millään tavalla esitelty kyseisessä dokumentissa, vaan siinä ollaan luotettu ihmisen tekemään työhön ja ajatteluun. Parametrit tuleekin uudessa dokumenttiversioissa esitellä niin, että jokaisen kuljetinmoduulityypin parametria vastaa jokin ulkoinen, sille syötettävä parametri. Parametrit jotka moduuleille syötetään, tulee kaikkien suunnitteluosastojen puolesta standardisoida niin, että nimi on vakio, ja ainoastaan muuttujina toimivat kuljetinnumerot, jotka ovat aina myös esitettyinä parametrin nimessä. Alla kuvassa havainnollistetaan mahdollista ratkaisumallia. (Kuva 12)

F	G	H	I	J	K	L
DataType	Inputs				Outputs	Data Type
		M100 CV1Dir				
UDT_Control	DeviceControl.OmgXX	i_Control		o_Device	DeviceStatus.CvXXXXXX	UDT_BasePositionStatus
UDT_HMI_In	DeviceHMI.Omg[XX].Position[X]	i_HMI		o_StatusToUi	DeviceHMIStatus.Omg[XX].Position[X]	DInt
UDT_MhuTaskBase	DeviceMHUTask.CvXXXXXX	i_MHU		o_DriveControl	DriveControl.CvXXXXXX	UDT_DriveControlBits
Bool	F_IF_Status.XXXXX.FunctionEnable	i_SafetyStopOk		o_ErrorActive	DeviceHMIStatus.Omg[XX].ErrorActive	Bool
Bool	FALSE	i_LocalControl		o_ER_Receive	Errors.OmgXX.CVXXXXXX_Receive	Bool
Bool	i_XXXXXPrdAtEnd	i_ProductAtEnd		o_ER_Send	Errors.OmgXX.CVXXXXXX_Send	Bool
Bool	DriveStatus.DrvXXXXXXmcnv.State.DriveOperational	i_DriveOperationalCv		o_ER_Sensor	Errors.OmgXX.CVXXXXXX_Sensor	Bool
Bool	DriveStatus.DrvXXXXXXmcnv.State.DriveRunning	i_DriveRunning		o_ER_Timeout	Errors.OmgXX.CVXXXXXX_Timeout	Bool

Kuva 12. Kuljetinmoduulitoimilohkon esittely lähdetietodokumentissa.

Kuvassa oleva esittely tulee suorittaa standarditoimilohkojen lisäksi kaikille projektikohtaisille variaatioille ohjelmallisen suorittamisen onnistumiseksi. Standarditoimilohkojen kohdalla riittää, että ne on kerran esitelty dokumentissa, ja kulkeutuvat näin aina pohjadokumentin mukana projektista toiseen.

Käytännössä sovelluksen näkökulmasta riittäisi, että ainoastaan kuljetintoimilohkojen projektikohtaiset variaatiot esitellään edellä mainitulla tavalla lähdedokumentissa, ja parametrit voitaisiin näin ollen ”kovakoodata”, mutta tämä tuo mukanaan ongelmia, kun vakiolohkoja päivitetään, ja jossain vaiheessa saavutaan tilanteeseen, ettei ohjelma enää osaa tehdä uusinta vakiotoimilohkoa, esimerkiksi IO-rajapinnan muuttuessa. Näillä perusteilla on siis järkevää, että sovellus osaa hakea dynaamisesti myös vakiolohkot lähdedokumenteista, joissa niiden rajapintaesittelyä voidaan ylläpitää samalla tavalla kuin toimilohkokirjastoja, esimerkiksi automaatiosuunnittelutiimin toimesta, tai erillisen nimetyn vastuuhenkilön toimesta.

9 OHJELMISTOSUUNNITTELU

9.1 Toiminnan kuvaus ja sovelluksen toteutuksen suunnittelu

Lopullisen ohjelmistoprojektin tarkoitus on tuottaa ohjelmakoodi korkeamman tason ohjelmointikielellä, joka generoi pohjatietodokumenttien pohjalta käyttötaparyhmälohkot Siemens PLC -projektiin. Vaatimuksena tässä opinnäytetyössä ei ole tuottaa valmista tuotetta, joka kykenisi generoimaan kaiken PLC-ohjelmakoodin käyttötaparyhmissä, vaan sen sijaan tutkia menetelmät, joilla tämä on mahdollista toteuttaa yrityksessä, sekä mitä resursseja ja muutoksia nykyiseen malliin se vaatii toimiakseen.

Kuitenkin opinnäytetyöhön kuuluu demosovelluksen kehittäminen, joka tuotetaan silmällä pitäen mahdollista jatkokehittämistä. Jatkokehittämisen mahdollistaminen edellyttää ohjelmistokehityksessä kattavaa dokumentaatiota ja koodin kommentointia, sekä ohjelmistorakenteen suunnittelua ja toteutusta mahdollisimman monikäyttöiseen malliin.

Edellisissä kappaleissa on koottu kattavasti tietoa erilaisten ohjelmistokehitysympäristöjen, kielien ja työkalujen käytöstä. Tämä on nähty tarpeelliseksi työssä esitettäväksi johtuen ohjelmistoprojektin kehittämiseen käytettävien työkalujen laajasta valikoimasta sekä monista eri toteutustavoista. Kuitenkaan tässä työssä ei ole lähdetty selvittämään kaikkia mahdollisia tapoja toteuttaa projektia, vaan ainoastaan avaamaan ja vertailemaan kahta eri toteutusmallia, jotka ovat tilanteeseen ja käyttötarkoitukseen nähden työn tekijän mielestä järkeviä.

9.1.1 Toteutusmalli 1: Microsoft Visual Studio ja C#

Ensimmäinen mahdollinen tapa on toteuttaa Openness-sovellus niin sanotusti käyttämällä suoraan rajapinnan tarjoamia funktioita C# tai Visual Basic -ohjelmointikielillä. Tämä sovellusmalli edellyttää Microsoft Visual Studion käyttöä projektin kehitysympäristönä, jossa on mahdollista päästä suoraan käsiksi Siemensin rajapintafunktioihin. Rajapinnan käyttöönotto Visual Studiossa edellyttää rajapintatiedoston tuomisen referenssiksi projektiin, ja vaatii myös ohjelmoijalta tarkempaa perehtymistä rajapinnan

luokka- ja oliorakenteeseen. Tämä malli ei vaadi toimiakseen muita ulkopuolisia sovelluksia, ja sillä on mahdollista säilyttää yhteys dynaamisesti koko ohjelman suorituksen ajan TIA Portal -ympäristöön, ja näin ollen tarjoaa toteutuksena laajan pohjan erilaisille toiminallisuuksille.

Käyttäjän näkökulmasta sovellus olisi vain yksi sovellus muiden joukossa käyttäjän tietokoneelle asennettuna, ja sitä olisi mahdollista hyödyntää dynaamisesti kaikissa projekteissa. Sovellus olisi myös erillään projektin dokumenteista, joka toisi mukanaan mahdollisuuden käyttäjälle valita mitä dokumenttia käytetään kunkin projektin osalla.

Toteutukseltaan tämä sovellusmalli vaatii ohjelmoijalta suhteellisen syvällistä perehtymistä Microsoftin ohjelmointikieliin, sekä olio-ohjelmoinnin maailmaan. Näiden lisäksi tekijän tulee olla myös hyvin perillä rajapintafunktioiden käytöstä, joka luo omia haasteitaan kattavasta dokumentaatiosta huolimatta. Nämä haasteet on myös huomattu yrityksen edustajan taholla, ja nähty tietyllä tapaa ongelmallisiksi sovelluksen jatkokehittämisen ja ylläpitämisen näkökulmasta, kuten myös se, ettei sovellus kulkeudu projektidokumenttien ohessa projektista toiseen, kuten aiemmin on totuttu.

9.1.2 Toteutusmalli 2: Excel ja Visual Basic for Applications

Toinen toteutusmalli on toteuttaa sovellus osaksi lähdedokumenttia, ja tavallaan “piilottaa” se dokumentin taustalle. Yrityksessä on jo pitkään ollut käytössä Excel-tiedosto, joka kulkee pohjana aina projektista toiseen yrityksen omalla verkkolevyllä. Tähän lähdedokumenttiin kerätään käytännössä kaikki tieto mitä PLC-ohjelmistoprojektin toteuttamiseen vaaditaan, kuten kuljetintoimilohkojen esittely, IO-lista, kuljetinlista sekä kuljettimien tyypit ja käyttötaparyhmät. Edellä mainittujen tietojen lisäksi dokumentti pitää sisällään paljon muuta tietoa, sekä näiden tietojen käsittelyyn käytettäviä VBA-makroja.

Toteutusmallin ajatuksena on sovelluksen kehittäminen pohjadokumenttiin ”makroksi”, joka siis kulkeutuisi aina käyttäjälleen tämän ladatessa dokumentin omalle tie-

tokoneelleen verkkolevyltä, ja näin ollen se olisi myös kaikkien muokattavissa ja ylläpidettävissä kaiken aikaa. Tämä tapa ollaan nähty ylläpidettävyyden lisäksi järkeväksi myös integroinnin näkökannasta, sen mahdollistaessa myös vanhempien makrojen siirtämisen uuden käyttöliittymän alle, ja selkeyttäisi dokumenttia käyttäjän näkökulmasta tuoden kaiken toiminallisuuden käyttöliittymään, jättäen datan Excel-taulukkoon. Käytännössä pidettäisiin Excel-taulukot jatkossa tietokantana, ja sovellus sovelluksena, mutta silti kaikki ovat samassa dokumentissa koko projektiryhmän saatavilla.

Heikkoutena tässä toteutusmallissa on sovelluksen vaatimat välikappaleet, joita vaaditaan TIA Portal -ympäristöön yhteyden muodostamiseksi. Excel VBA -ohjelmakoodista ei ole suoraan mahdollista viitata Siemens-rajapintafunktioihin, vaan siihen tulee käyttää OpennessScripter.exe-sovellusta, ja sen avaamiseen sekä suorittamiseen komentojonotiedostoa. Näiden yhteistoimintaa on käsitelty aikaisemmissa kappaleissa, mutta yhtenä miinuksena voidaan vielä mainita, ettei tällä tavalla ole mahdollista luoda dynaamista yhteyttä sovelluksen ja TIA Portal -ympäristön välille, kuten toisessa toteutusmallissa on.

Tämän toteutusmallin nojatessa vahvasti OpennessScripter.exe-sovelluksen käyttöön, rajautuu toteutuksessa myös huomattava määrä ominaisuuksia ja mahdollisuuksia pois käytöstä, ja näin ollen myös toteutuksen mielikuvituksellisuus on rajallisempaa. Kuitenkin tämän projektin kohdalla ainoa haluttu ominaisuus liittyy ohjelmalohkojen generointiin Siemens TIA Portal -projektissa, johon OpennessScripter.exe sovelluksen toiminallisuudet riittävät hyvin.

9.1.3 Toteutusmallin valinta ja perustelut

Sovelluksen toteutukseen valitaan projektissa toteutustapa 2, eli Excel VBA -sovelluksen kehittäminen. Toteutusmalli tukee vahvasti ajatusta sovelluksen jatkokehittämisen helppoudesta, ja on luonteeltaan huomattavasti helpompi lähestyttävä projektiryhmän näkökulmasta, heidän omatessaan jo kokemusta Excel-makrojen käytöstä ja kehityksestä.

Valintaa tukee myös ajatus integraatiosta, joka tarkoittaisi Excel-dokumentin jatkokehittämistä helpommin käytettävään suuntaan myös vanhojen makrojen osalta. Yhtenä tärkeänä tekijänä tässä on myös se, että yrityksessä tämän työn tekijän kanssa samaan aikaan toimii toinen opinnäytetyön tekijä, joka toteuttaa saman kaltaista sovellusta Rockwell-logiikkamerkin ohjelmistoprojektien automatisointiin liittyen. Myös tämä Juho Keskisen suunnittelema sovellus liitettäisiin samaan lähdedokumenttiin, joka mahdollistaa myös samojen funktiokirjastojen käytön ohjelmistokehityksessä sekä käyttöliittymien yhdenmukaistamisen.

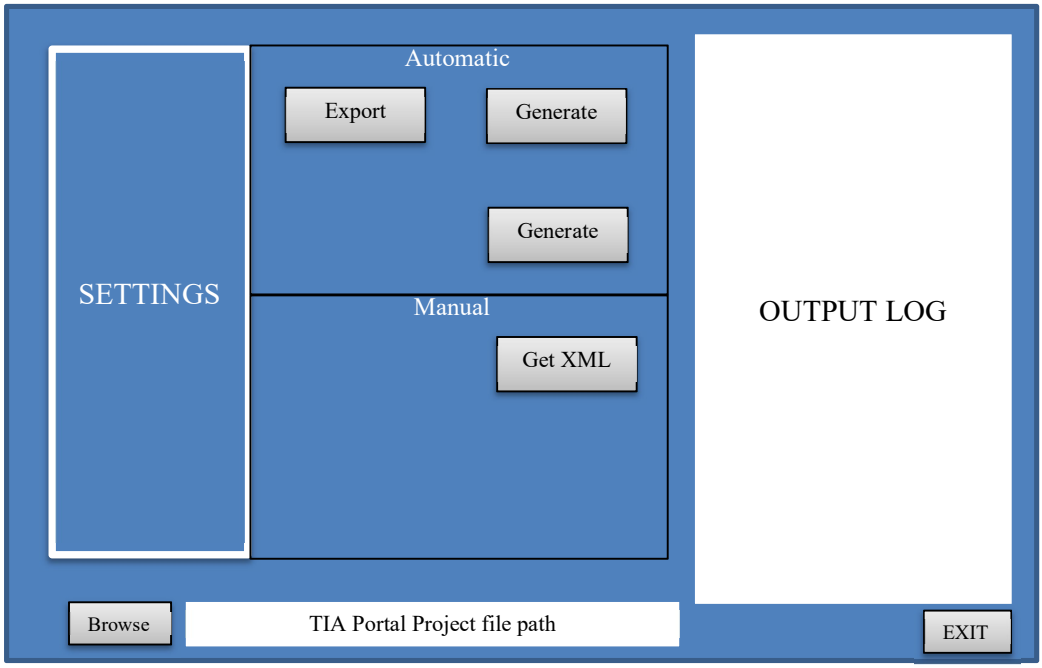
9.2 Käyttöliittymän suunnittelu

Sovelluksen käytettävyyden näkökulmasta on tarpeellista tehdä sille oma käyttöliittymänsä. Jotta käyttäjän kokemus sovelluksesta olisi hyvä ja käyttäminen helppoa, tulee käyttöliittymän olla graafinen, kuvaava ja selkeästi käyttötarkoitukseensa sopiva. Käyttöliittymän tulee mahdollistaa tiettyjen ohjelman suoritukseen liittyvien parametrien vastaanottaminen käyttäjältä, ja käyttäjälle on informoitava mahdollisista poikkeuksista ja väärin tehdyistä toimenpiteistä, kuitenkaan haittaa aiheuttamatta.

Excel-dokumentin auettua käyttöliittymän ei tule vielä ilmaantua esille, vaan käyttäjälle pitää antaa mahdollisuus muokata Excel-taulukkoa ilman häiriötekijöitä, ja sen tulee olla ensisijainen vaihtoehto. Käyttäjä voi halutessaan käynnistää käyttöliittymän erillisestä painikkeesta, joka löytyy Excel-dokumentin tietyltä välilehdeltä. Käyttöliittymän auettua sovellus piilottaa Excel-taulukon käyttäjältä, luoden näin selkeämmän alustan sovelluksen käyttämiselle. Käyttäjän painettua ”Exit” tai ”X” painiketta, käyttöliittymä sulkeutuu, ja Excel-taulukko palautuu näkyville.

Käyttöliittymässä sovelluksen toiminallisuudet on sisällytetty painikkeisiin, jotka ovat selkeästi eritelty käyttötarkoitustensa mukaan ja ne on nimetty kuvaavasti. Asetukset ohjelman toiminnan hallinnointiin sijoitetaan selkeästi erilleen, ja niitä voidaan muokata ennen sovelluksen toiminnallisuuden käynnistystä. Käyttöliittymään sijoitetaan myös kenttä lokiteksteille, jossa ohjelman suorituksen ajan annetaan tietoa sovelluksen ohjelmakierron etenemisestä ja virheistä sekä opastetaan käyttäjää toimimaan oikein

jokaisessa tilanteessa. Ohessa kuva käyttöliittymän graafisesta hahmottelusta (Kuva 13).



Kuva 13. Graafisen käyttöliittymän hahmotelma.

Käyttöliittymä jaotellaan selkeyden vuoksi kolmeen osaan: Asetukset, Automaattinen toiminta ja Manuaalinen toiminta. Asetuksissa on käyttäjällä mahdollisuus muokata parametreja liittyen ohjelmalohkojen generointiin. Käyttäjän tulee voida päättää generoitavien ohjelmalohkojen ja tietueiden aloitusindeksi, sekä haluttu kansio johon ne generoidaan projektissa. Käyttäjälle annetaan myös mahdollisuus valita, suoritetaanko toimenpiteet taustalla, vai avataanko TIA Portal -käyttöliittymä, jolloin käyttäjän on mahdollista havainnoida toimintaa.

Jotta projektiin on mahdollista tuoda ohjelmalohkoja, pitää käyttäjän myös antaa tiedostopolku projektin sijainnista. Tämä toteutetaan ”Browse”-painikkeella, joka avaa tiedostovalitsimen Windowsissa, ja poistaa siten käsin kirjoittamisen ja muistamisen vaivan käyttäjältä.

Käyttöliittymään tuodaan ”Automaattinen toiminta”-alueelle kolme painonappia, joista käyttäjä voi valita haluaako hän generoida käyttötaparyhmätlohkot ja kuljetintietueet, vai moottorinohjaustoimilohkojen tietueet projektiin. Käyttäjän on myös mahdollista tuoda projektista ”Export”-painonapilla kaikki projektien generoimiseen vaadittava tieto tiedostosijaintiin käyttäjän tietokoneelle. Tämä toiminto pitää suorittaa ainakin kerran projektissa, koska tietueiden generointi vaatii vakiotoimilohkojen sisältämän tiedon käyttöä, eikä sitä haluta ”kovakoodata” projektiin toimilohkojen päivitysten mahdollistamiseksi.

Viimeiseksi käyttöliittymään lisätään lokitekstikenttä, josta käyttäjälle näytetään tietoa ohjelmassa tapahtuvista poikkeuksista, sekä opastetaan käyttäjää toimimaan oikein joka tilanteessa. Lokitekstikenttä saa myös tietoa, jos Openness-rajapinnassa tapahtuu poikkeuksia, esimerkiksi ohjelmalohkojen ollessa jo projektissa, tai projektin ollessa auki muualla.

9.3 Sovelluksen arkkitehtuuri ja rakenne

Vanhoissa, jo yrityksessä käytössä olevissa makroissa ei olla jouduttu vielä suuremmin miettimään ohjelmiston arkkitehtuuria, niiden ollessa vielä suhteellisen suppeita ja helppolukuisia. Tätä sovellusta kehitettäessä on kuitenkin hyvä keskittyä tarkemmin suunnittelemaan ohjelmistorakennetta, sovelluksen monimutkaisuuden ja projektin laajuuden vuoksi.

Selkeyden vuoksi on valittu ohjelmointiparadigmaksi olio-ohjelmointi, joka soveltuu ajatusmalliltaan parhaiten suoritettaviin toimenpiteisiin. Kuten tämän opinnäytetyön kappaleessa 6.2 on mainittu, olio-ohjelmointi keskittyy kuvaamaan reaalimaailman asioita ohjelmoinnin sanoin. Sovelluksen ohjelmointi voidaankin jakaa karkeasti seuraavasti: Halutaan tuottaa Siemens PLC -ohjelmalohkoja: käyttötaparyhmiä. Ohjelmalohkokansio koostuu käyttötaparyhmistä sekä tietueista, ja käyttötaparyhmä koostuu ohjelmariveistä (Networks).

Näistä neljästä tehdään omat luokkamoduulinsa, joista voidaan synnyttää olioita vastaamaan aina tietystä osa-alueesta ohjelman suorituksessa. Niiden lisäksi tarvitaan

paljon muita moduuleja, jotka pitävät sisällään tiedostojen hallintaa sekä muita aputoimintoja.

Tapa, jolla ohjelmisto kykenee generoimaan ohjelmakoodia TIA Portal -projektiin, perustuu XML-tiedoston kirjoittamiseen. Yksi XML-tiedosto pitää sisällään aina joko yhden ohjelmalohkon (FC) tai tietueen (DB). Näitä XML-tiedostoja generoidaan tarvittava määrä ohjelman suorituksen aikana, jotka ajetaan projektiin ohjelmalohkoiksi. Siemensin käyttämä XML-kieli on verraten monimutkaista luettavaa, mutta siinä on perusrakenteita, joiden mukaan sitä on helpompi lähteä työstämään, jos sen kirjoitus jaetaan ohjelmassa oikein. Alla taulukossa on esitelty Siemensin käyttämää XML-rakennetta, ja selitetty tärkeimpiä osia ja niiden tarkoitusta (Taulukko 1)

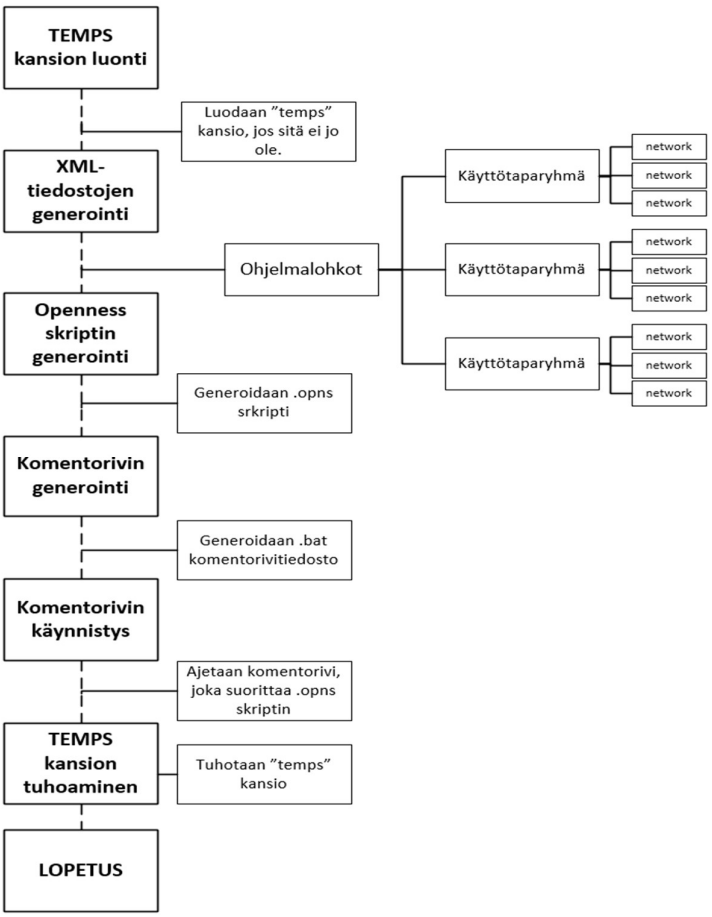
<SW.Blocks.FC>	FC-lohkon määrittely, pitää sisällään yhden lohkon.
<SW.Blocks.CompileUnit>	Network-määrittely, pitää sisällään yhden networkin kaikki tiedot.
<Parts>	Networkin osat. Pitää sisällään Networkissa käytettävät muuttujat ja niiden määrittelyt, sekä mahdolliset LAD-kieliset elementit kuten kärjet ja kelat.
<Access>	Yhden muuttujan määrittely <Parts> sisällä
<Part…….>	Yhden LAD-kielisen elementin määrittely <Parts> sisällä.
<Wires>	Pitää sisällään tiedot, joiden avulla osat linkitetään toisiinsa Networkissa.
<Wire>	Yksi yhdistys <Wires> sisällä

Taulukko 1. Siemens XML -kuvauskielen elementtejä.

Yhdessä käyttötaparyhmälohkoa kuvaavassa XML-kielisessä dokumentissa on noin yksitoistatuhatta riviä tekstiä, joka osaltaan tekee sen lukemisen ja ymmärtämisen hankalaksi. XML-tiedosto kannattaakin näiden edellä mainittujen ”osien” avulla jakaa hieman pienempiin palasiin. Huomattavaa on myös, että yhtä Networkia kuvaavassa

osassa indeksoinnit ovat sisäisiä, ja ne aloittavat aina numerosta 21. Networkien ulkopuoliset indeksit taas juoksevat numerosta 1 alkaen, välittämättä Networkien sisäisitä indekseistä, ja toisin päin.

Kuten edellä on mainittu, XML-tiedostot generoidaan, ja sen jälkeen tuodaan TIA Portal -projektiin käyttämällä OpennessScripter.exe-sovellusta. Sovellus generoi tätä varten oman skriptin, jonka rakenne ja pituus riippuvat sovelluksella tehtävästä toimenpiteestä. Tämä skripti tallennetaan XML-tiedostojen tavoin ”temps”-kansioon, ja sieltä se osoitetaan OpennessScripterin ajettavaksi. Tämä osoitus tapahtuu käynnistämällä generoitu komentorivitiedosto, joka on myös tallennettuna ajonaikaisiin tiedostoihin ”temps”-kansioon. Komentorivi käskyttää avaamaan OpennessScripter.exe-sovelluksen, ja sen ajamaan generoidun skriptin, joka tuo edellä luodut XML-tiedostot TIA Portal -projektiin ohjelmalohkoiksi. Alla kuvassa esitetty sekvenssikaavio ohjelman toiminnasta (Kuva 14).



Kuva 14. Sovelluksen toimintasekvenssi.

10 TULOKSET, TIA OPENNESS -SOVELLUS

10.1 Sovelluksen toiminta ja käyttö

Sovellus toteutettiin Excel-dokumentin taustalle, kiinteäksi osaksi lähdedokumenttia. Itse lähdetietodokumenttia jouduttiin muokkaamaan siltä osin, kun oli tarpeellista, esimerkiksi lisäämällä yhdelle välilehdelle vielä jokaisen projektissa mukana olevan SEW Movimot -väyläohjatun moottorikäytön tunnukset, sekä tarkentamaan tietoja yrityksessä käytettävien vakiokuljetintoimilohkojen osalta.

Sovellus osaa hakea tiedot automaattisesti lähdetietodokumentista, ja näiden pohjalta luoda halutun laista XML-koodia, jonka se generoi automaattisesti tai TIA Portal -projektiin ohjelmalohkoiksi. Sovelluksessa käyttäjälle annetaan mahdollisuus päättää suunnitteluvaiheessa määritellyt ominaisuudet, sekä valita projekti, johon ohjelmalohkot generoidaan. Käyttäjä voi halutessaan generoida pelkästään XML-tiedostot haluamaansa tiedostosijaintiin, tai vaihtoehtoisesti automaattisesti suoraan projektiin.

Ohjelma itsessään on koostaan ja tuottamastaan datamäärästä huolimatta yllättävän nopea, mutta generointi TIA Portal -ympäristön päässä kestää aikansa. Tästä syystä päädyttiin ohjelmoidessa ratkaisuun, jossa käyttäjälle annetaan kaksi erillistä painonappia käyttöliittymään, joista toisesta voidaan generoida käyttötaparyhmien ohjelmakoodi ja tietueet, ja toisesta taas erikseen moottoriohjaustietueet (DB).

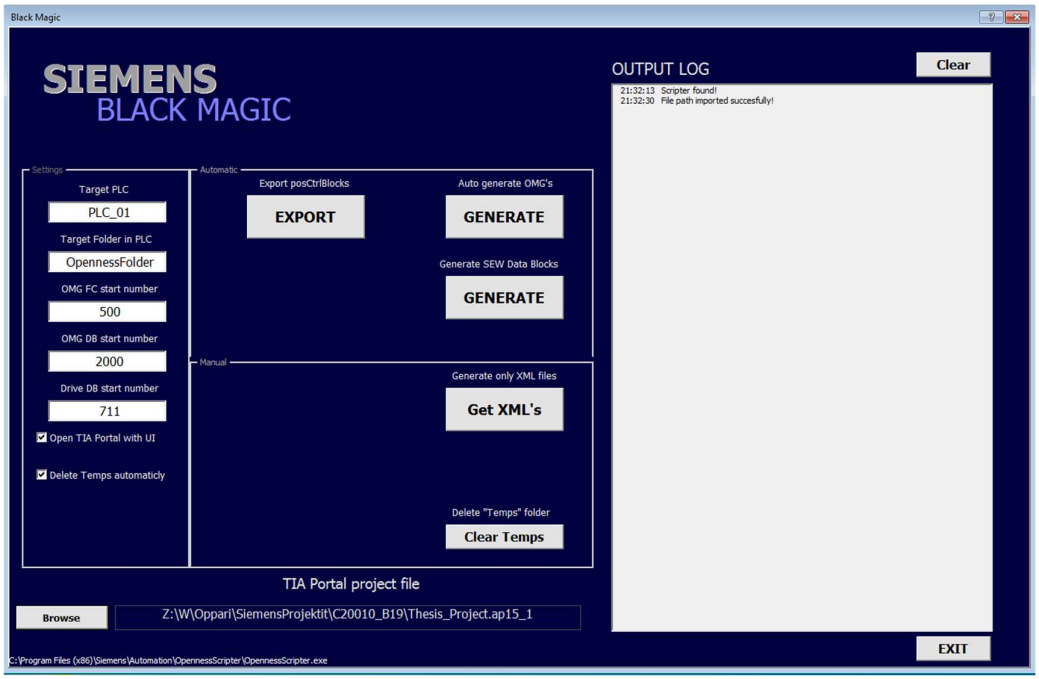
Sovellus pyrittiin tekemään niin, että suurin osa XML-koodista on dynaamista ja ohjelman kirjoittamaa. Kopiointia tuoduista mallilohkoista pyrittiin välttämään, jotta jatkossa ei tarvitsisi ylläpitää ohjelmalohkokirjastoja niin TIA Portal -kirjastossa kuin ohjelman mallilohkojen osalta. Ainoa lohko, jonka kutsu ja ohjelmarivi lohkossa on tuotu valmiista mallista, on moottoriohjaustoimilohko, sen sisältäessä hankalasti muokattavaa XML-koodia. Tämä on kuitenkin muokattavissa dynaamiseksi toteutustavaksi jatkokehitettäessä.

Jotta ohjelma pystyy generoimaan tietueet (DB), pitää käyttäjän kunkin projektin alkaessa tuoda kerran projektitiedostosta kaikki siinä käytettävät vakiolohkot, niin globaalit kuin projektikohtaiset. Tämä prosessi on myös automatisoitu, eikä käyttäjän tarvitse kuin painaa ”Export”-painonappia, jolloin sovellus hoitaa prosessin alusta loppuun. Huomioitavaa on, ettei näitä lohkoja tuoda kopioimistarkoituksessa, vaan koska ne sisältävät tietueisiin vaadittavaa dataa, jota ei ole mahdollista saada muualta, tai sen tuottaminen on tarpeettoman työlästä.

Kokonaisuudessaan ohjelma on hyvällä mallilla, ja sen osalta ollaan päästy asetettuihin tavoitteisiin. Se ei vielä generoi käyttötaparyhmää alusta loppuun, mutta tekee sille rungon, joka pitää sisällään jokaisen kuljetinohjaustoimilohko- ja moottorinohjaustoimilohkokutsun, parametroituna mahdollisin osin, sekä käyttötaparyhmänlohkon alussa olevat kolme ohjelmariviä (Networks), joissa alustetaan tilatiedot LAD-kielisillä kärjillä.

10.2 Käyttöliittymä

Käyttöliittymän osalta pysyttiin pitkälti suunnitelmissa, joskin tiettyjä toiminnallisuuksia lisättiin matkalla niiden osoittautuessa hyödylliseksi. Näitä ovat esimerkiksi ”Temps”-kansion tuhoaminen painonapilla, sekä valinta sille halutaanko kyseinen kansio poistaa automaattisesti ohjelmakierron lopussa. Tietyissä tapauksissa kansio voidaan haluta säilyttää, esimerkiksi sovelluksen kehitysvaiheessa on hyvä kehittäjän päästä tarkastelemaan ja etsimään mahdollisia ohjelmassa tapahtuvia vikoja. Tämä on mahdotonta, jos kansio tuhotaan aina automaattisesti. Käyttäjälle lisättiin myös painonappi lokikentän tyhjentämistä varten. Alla kuva valmiista käyttöliittymästä (Kuva 15).



Kuva 15. Toteutunut käyttöliittymä.

10.3 Sovelluksella saavutettu hyöty yrityksessä

Vaikka sovellus on vasta kehitysvaiheessa, voidaan sillä jo nyt saavuttaa monien tuntien säästöä yrityksen isommissa ohjelmistoprojekteissa. Lohkot, joita sovellus generoi projektiin automaattisesti, ovat pitkälti olleet käsityötä, ja tylsää ”kopioi-liitä”-toimintaa, josta pyritään pääsemään eroon.

Suurin hyöty sovelluksen kehittämisestä yritykselle on kuitenkin tutkimusmielessä. Vastaavaa sovellusta ei ole yritetty kehittää yrityksessä ennen, ja Openness-rajapinta tuntuu olevan vielä muutenkin hyvin vähän käytetty ominaisuus, eikä siitä ole hirveästi esimerkinomaisia ratkaisuja netistä löydettävissä. Kun runko sovellukselle on tehty, ja se vapautetaan yrityksessä muiden käyttöön, on todennäköistä, että myös ideoita ja käyttökohteita vastaavanlaiselle sovellukselle löytyy lisää, ja rajapintaa opitaan hyödyntämään laajemmin PLC-ohjelmistokehityksessä.

Kolmas hyöty on sovelluksen kehittämisen aiheuttama paine standardisoida muita osia ohjelmistokehityksen saralla, sekä sen vaatimia oheistoimintoja, kuten IO-listojen

tuottamista sähkösuunnitteluosastolla. Jatkuva kehitystyö myös kannustaa yritystä kehittämään omia toimintamallejaan ketterämpään suuntaan, ja tuotteistamaan toimintojaan entistä pidemmälle tämän tarjoaman taloudellisen hyödyn voimin. Jatkossa työvoimaa voitaisiin vapauttaa kopiointityön sijaan enemmän ajattelua vaativiin, mielekkäämpiin tehtäviin, ja voidaan jättää tarpeelliset, mutta tylsät työt koneen tehtäväksi.

10.4 Käyttöönotto ja käyttöohjeet

Jotta sovellus voidaan vapauttaa yleiseen käyttöön projektiryhmälle, vaatii se tietynlaisten toimenpiteiden suorittamisen ennen tätä. Ensimmäiseksi jokaiselle sovellusta käyttävän virtuaalitietokoneelle tulee asentaa Excel, ja käyttäjätili on määriteltävä kappaleessa 4.5 selostettuun malliin OpennessUser-ryhmään. Yhteensopivuuden takaamiseksi olisi suotavaa asentaa Excelistä versio 2013 tai uudempi, mutta tarpeen tullen myös aikaisemmat versiot saadaan toimintaan.

Excel-lähdedokumentin pohjadokumentti, johon projektit luodaan, tulee päivittää sisältämään sovelluksen ohjelmakoodi, ja siihen tulee päivittää vaadittavat tiedot, jotka määriteltiin kappaleessa 10.1. Lisäksi dokumenttia täytettäessä projektin tiedoilla, on kiinnitettävä entistä enemmän huomiota, ettei tiedoissa esiinny ristiriitoja, ja niiden on oltava tarkemmin määriteltyjä.

Excel-dokumentin täyttämiseen tarvittavat käyttöohjeet sisällytetään Excel-dokumenttiin, jotta ne ovat jokaiselle helposti saatavilla. Sovelluksen käyttöohjeet taas luodaan paikallisesti käyttöliittymän kysymysmerkki-painikkeen taakse, sekä kattavampi dokumentaatio tehdään vielä lisäksi saataville yrityksen omaan järjestelmään.

Edellä mainittujen toimenpiteiden lisäksi, käyttäjille tulee suorittaa perusperehdytys ohjelmiston käyttöön sekä sen mahdolliseen muokkaamiseen. Ohjelmakoodissa itsessään on kattava kommentointi englannin kielellä, jonka tarkoituksena on helpottaa sovelluksen jatkokehittämistä ja avata toimintaa muillekin kuin tekijälle.

11 JATKOKEHITTÄMINEN

Sovelluksen ollessa vasta pelkkä runko, on jatkokehitettävää runsaasti. Kuitenkin ensimmäinen asia joka tulee kehittää, on suunnitteluosastojen välinen standardimalli tiedon esittämiseen ja dokumentointiin, projektista riippumatta. Ilman standardimallia on mahdotonta automatisoida mitään, automaation ja ohjelmiston luottaessa aina samanlaiseen tiedonesitysmalliin.

Sovelluksessa itsessään kehitettävää olisi ainakin käyttötaparyhmien generoimisessa puuttuvien ohjelmarivien lisääminen generointiin, joita ovat esimerkiksi ohjelmalohkon loppupään rivit (Networks), joissa määritellään ”Halt”-virheet sekä valomajakoiden toiminta. Näiden lisäksi ohjelmasta puuttuu vielä alkupäästä ohjelmalohkoa ohjelmarivejä, jotka sisältävät instanssikutsun sekä tilojen asettelua käyttötaparyhmälle.

Myös käyttöliittymää voidaan kehittää yhdessä sovelluksen kanssa vastaamaan yrityksen muuttuvia tarpeita, ja siihen voidaan lisätä mitä mielikuvituksellisimpia toimintoja, esimerkiksi ohjelmalohkojen muokkausta ”export - find and replace – import”-toiminnallisuus, joka mahdollistaisi esimerkiksi muutaman arvon vaihtamisen samaan tyyliin koko projektissa, tai vastaavaa.

Sovelluksessa on myös pyritty ottamaan mahdollisia käyttäjästä aiheutuvia virhetilanteita huomioon, mutta varmasti siinä saralla on vielä paljon kehitettävää, kun käyttäjäkokemuksia ja poikkeustilanteita saadaan raportoiduksi.

Yksi tärkeimmistä kehityskohteista olisi kuitenkin integrointi. Kuten edellä on mainittu, sisältää nykyinen Excel-lähdedokumentti useita makroja, ja myös useita painonappeja sijoitettuna pitkin dokumenttia vailla järjestystä. Nämä kaikki voitaisiin jatkossa asettaa saman käyttöliittymän alle, joka selkeyttäisi Excel-dokumenttia, ja helpottaisi tällä hetkellä jo sekavaa makrojen suorittamista. Myös Juho Keskisen tekemä ohjelmistoprojekti voidaan integroida samaa ympäristöön, jolloin yhteistoiminnan tuloksena on mahdollista saada kaikkia käyttäjiä tyydyttävästi palveleva sovellus, sekä järjestyksessä oleva lähdetietodokumentti.

12 YHTEENVETO

Työn tavoitteena oli tutkia Siemens Openness -rajapinnan hyödyntämistä PLC-ohjelmistokehityksen automatisoinnissa, sekä tuottaa demo-sovellus toimimaan tulevaisuudessa laajamittaisemman sovelluksen kehittämisen runkona. Openness-sovelluksen tarkoituksena oli generoida automaattisesti kuljetinkäyttötaparyhmien ohjelmalohkot Siemens TIA Portal -projektiin, jolloin käsin tehtävä kopiointityö voitaisiin saattaa yrityksessä minimiin, ja säästää kymmeniä tunteja PLC-ohjelmoinnissa.

Järkevimmäksi tavaksi yrityksessä koettiin sovelluksen toteuttaminen jo olemassa olevaan Excel-lähdetietodokumenttiin makroksi, jolloin sovelluksen jatkokehittäminen olisi mahdollista kaikille, ja se voitaisiin integroida jo olemassa olevien makrojen, sekä toisen opinnäytetyön tiimoilla sovellusta kehittävän henkilön ohjelmakoodin kanssa samaan dokumenttiin.

Sovelluksen ohjelmoinnin tapahtuessa korkeamman tason ohjelmointikieltä (VBA) käyttäen oli sitä tehdessä hyötyä koulussa opituista ohjelmointitaidoista, joiden perusteet rakennettiin ohjelmointikursseilla sekä syvennettiin myöhemmin mielenkiintoisten opiskelijaprojektien parissa. Työn liittyessä vahvasti PLC-ohjelmointiin ja logiikkaohjelmien arkkitehtuuriin hyödylliseksi koettiin myös erityisesti Automaation ohjausjärjestelmät -moduulin kursseilla opitut perustaidot Siemens PLC -ohjelmoinnin saralla.

Työn valmistuttua yrityksessä voidaan säästää työtunteja jo olemassa olevan sovelluksen käyttämisellä, mutta mikä tärkeintä, on saatu tutkimustuloksia ja runko myöhempää sovelluksen kehittämistä silmällä pitäen. Jatkossa yrityksen PLC-ohjelmointia voitaisiin automatisoida tuloksien pohjalta huomattava määrä, mikä taas antaa yrityksen ohjelmistopuolen osastolle kykyä vastata yhä laajempien projektien asettamiin aikataulu-ja resurssihaasteisiin.

LÄHTEET

Ammattitason ohjelmistot. Viitattu 28.7.2019. https://www.microsoft.com/fi-fi/store/b/software

Avoimen rajapinnan määritelmä. Viitattu 27.7.2019. http://avoinrajapinta.fi/

Cimcorp-konserni pähkinänkuoressa. Viitattu 21.7.2019. https://www.cimcorp.com/fi/cimcorp/cimcorp-konserni

Fokus sovelluksissa, digitalisaatiossa ja tehokkuudessa. Viitattu 23.7.2019. https://www.pjc.fi/ajankohtaista/2018/02/05/siemensin-tia-portal-v15

Guideline for Library Handling. Viitattu 25.7.2019. https://support.industry.siemens.com/cs/document/109747503/guideline-on-library-handling-in-tia-portal-?dti=0&lc=en-WW

Hyvönen, M., Lappalainen, V. & Lakanen, A. 2013. Ohjelmointi 1 C#. Luentomoniste. Jyväskylän yliopisto. Viitattu 28.7.2019. https://kurssit.it.jyu.fi/ITKP102/monistecs/pdf/

Introduction to Visual Studio. Viitattu 28.7.2019. https://www.geeksforgeeks.org/introduction-to-visual-studio/

Keinonen, K. J. 2010. Microsoft Excel Edistynyt käyttö. Helsinki: Ornanet.

Komentojonossa käytettävät komennot. Viitattu 14.8.2019. http://appro.mit.jyu.fi/doc/komentojonot/index6.html

Lesson one: What is VBA? Viitattu 28.7.2019. https://www.thespreadsheetguru.com/vba-intro/what-is-vba

Miten komentojono tehdään? Viitattu 14.8.2019. http://appro.mit.jyu.fi/doc/komentojonot/index4.html

OpennessScripter: Detailed Documentation. Viitattu 27.7.2019. https://support.industry.siemens.com/cs/document/109742322/tool-f%C3%BCr-eine-einfachere-nutzung-der-tia-portal-openness-schnittstelle-(openness-scripter)?dti=0&lc=de-WW

OpennessScripter: Introduction. Viitattu 27.7.2019. https://support.industry.siemens.com/cs/document/109742322/tool-f%C3%BCr-eine-einfachere-nutzung-der-tia-portal-openness-schnittstelle-(openness-scripter)?dti=0&lc=de-WW

Peltomäki, J. 2014. Pieni Java 8 -kirja Helsinki: Books on Demand.

PLCopen, XML Exchange. Viitattu 14.8.2019. https://www.plcopen.org/technical-activities/xml-exchange

Robotiikkaa vuodesta 1975. Viitattu 21.7.2019. https://www.cimcorp.com/fi/cimcorp/historia

Siemens: Teollisuuden tuotteet ja ratkaisut. Viitattu 23.7.2019. http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/tia_portal.php

Siemens: Tuhansien tuntien säästö. Viitattu 25.7.2019. https://new.siemens.com/fi/fi/tuotteet/teollisuusautomaatio/referenssit/tuhansien-tuntien-saasto.html

Siemens: How is XML file structured for blocks? Viitattu 14.8.2019. https://support.industry.siemens.com/cs/document/109480446/how-is-the-xml-file-structured-for-blocks-(export-import-in-tia-portal-openness-v13-sp1)-?dti=0&pnid=14667&lc=en-WW

SIMATIC Openness: Automating creation of projects. Viitattu 27.7.2019. https://support.industry.siemens.com/cs/document/109477163/simatic-openness%3A-automating-creation-of-projects?dti=0&lc=en-AE

Sivonen, V. 2004. Ohjelmointikielten periaatteet: C# -kieli. Seminaariesitelmä. Helsingin yliopisto. Viitattu 28.7.2019. https://www.cs.helsinki.fi/u/pohjalai/k04/ohpe/seminar/Sivonen-CSharp.pdf

Suvela, T. Käyttötaparyhmä, toimintatilat. Luentomoniste. Satakunnan Ammattikorkeakoulu. Viitattu 14.8.2019. https://docplayer.fi/26173474-Kayttotaparyhma-toimintatilat.html

Taanila, A. 2013. Excel VBA-ohjelmointi. Luentomoniste. Haaga-Helia Ammattikorkeakoulu. Viitattu 28.7.2019. http://myy.haaga-helia.fi/~taaak/vba/vba.pdf

Teollisuus 4.0 – ”Suomen oltava kilpailukykyinen vaihtoehto, kun teollisuuden paluumuutto Aasiasta Eurooppaan alkaa”. Viitattu 23.7.2019. https://www.tekniikkatalous.fi/uutiset/teollisuus-40-suomen-oltava-kilpailukykyinen-vaihtoehto-kun-teollisuuden-paluumuutto-aasiasta-eurooppaan-alkaa/73193f56-c36f-372c-b61e-40ad27b80144

TIA Portal Openness: Introduction and Demo Application. Viitattu 27.7.2019. https://support.industry.siemens.com/cs/document/108716692/tia-portal-openness%3A-introduction-and-demo-application?dti=0&lc=en-WW

TIA Portal SIMATIC STEP 7. Viitattu 23.7.2019. http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/ohjelmoitavat_logiikat_simatic/ohjelmistot/tia_portal_step7.htm

Web-opas, ”Mikä on XML?”. Viitattu 14.8.2019. http://www.webopas.net/xml.html

Vitikka, J. 2000. Visual Basic –sovelluskehitin. Seminaari. Helsingin yliopisto. Viitattu 28.7.2019. https://docplayer.fi/18218712-Visual-basic-sovelluskehitin-juha-vitikka.html

XML-tietojen tuominen. Viitattu 14.8.2019. https://support.office.com/fi-fi/article/xml-tietojen-tuominen-6eca3906-d6c9-4f0d-b911-c736da817fa4