Arto Mertala

PLC-OHJELMAN LAADUNVARMISTUS PLC CHECKER-OHJELMAN AVULLA

Sähkötekniikan koulutusohjelma
2017

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

PLC-OHJELMAN LAADUNVARMISTUS PLC CHECKER-OHJELMAN AVULLA

Arto Mertala
Satakunnan ammattikorkeakoulu
Sähkötekniikan koulutusohjelma
Lokakuu 2017
Sivumäärä: 35
Liitteitä: 1

Asiasanat: ohjelmoitava logiikka, PLC, ohjelmien laatu, laadunvarmistus

---

Tämän opinnäytetyön tavoitteena on tutkia tietokonepohjaisen ohjelmoitavien logiikkaohjelmien analysointityökalun PLC Checkerin käytettävyyttä PLC-ohjelmien laadunvarmistamiseen. Työhön sisältyi PLC Checker-ohjelmaan perehtyminen ja valitun PLC-ohjelmiston analysoiminen PLC Checker-ohjelmalla.

Työn perusteella voidaan todeta, että PLC Checker on loistava työkalu PLC-ohjelmien laadunvarmistamisessa.

PLC PROGRAM QUALITY CONTROL WITH PLC CHECKER

Mertala Arto
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in electrical engineering
October 2017
Number of pages: 35
Appendices: 1

Keywords: Programmable Logic Controller, PLC, programs quality, quality control

---

The target of this thesis has been to investigate the use of the computer program PLC Checker programmable logical control analyzing tool for checking the quality control of PLC-programs. The work consisted of familiarizing to the PLC-Checker and analyzing a selected PLC-program with the PLC Checker.

According to the work done it can be stated that the PLC Checker is a great tool for quality control of PLC-programs.

SISÄLLYS

1 JOHDANTO

Opinnäytetyö tehtiin CS Control Software Oy:lle. CS Control Software Oy on vuonna 2006 Espooseen perustettu ja monipuoliseksi ja maailman laajuiseksi kasvanut automatisointiin erikoistunut yritys. Yritys tarjoaa korkeatasoista tietotaitoa useille aloille, kuten sähköntuotantoteollisuudelle infrastruktuurien ohjausjärjestelmille, merenkulun ohjausjärjestelmille ja kaivosteollisuudelle. Palvelu tarjonta kattaa kaikki tasot valvontatuesta täydelliseen käyttöönottoon ja käyttöönottotukeen. (CS Control Softwaren www-sivut)

Ohjelmoitavien logiikoiden ohjelmien monimutkaistuessa ja kasvaessa, niiden manuaalinen tarkistus on työlästä ja kallista ja on jopa mahdotonta suuren koodin määrän johdosta. Tämän opinnäytetyön tavoitteena on tutkia tietokonepohjaisen ohjelmoitavien logiikkaohjelmien analysointityökalun PLC Checkerin käytettävyyttä ohjelmien laadunvarmistamiseen. PLC Checker on CS Control Software Oy:n yhteistyökumppanin Itris Automationin kehittämä ohjelmisto.

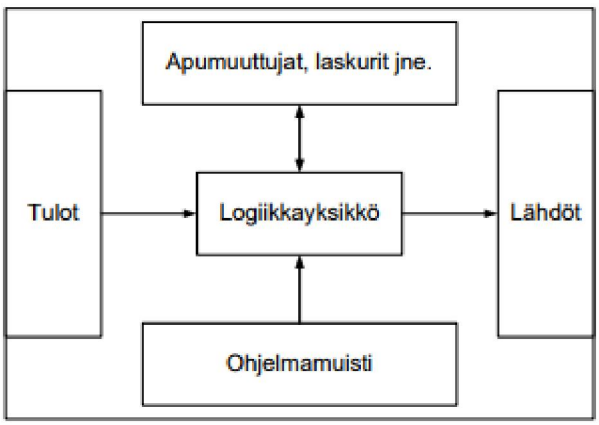
Työhön sisältyi PLC Checker-ohjelmaan ja sen säännöksiin perehtyminen, jonka jälkeen suoritettiin valitun asiakkaan PLC-ohjelmiston analysointi.

2 OHJELMOITAVA LOGIIKKA (PLC)

Ohjelmoitava logiikka (Programmable Logic Controller, PLC) on mikroprosessoripohjainen laite ja sitä käytetään automaatioprosessien ohjaukseen. Ohjelmoitavan logiikan isänä pidetään Dick Morleytä. Morley kehitti ensimmäisen logiikan vuonna 1968 General Motorssin tarpeisiin. Laitteella korvattiin releteknikalla ohjattavien koneiden ohjauksia. Laite sai nimekseen MoDiCon (Modular Digital Controller). MoDiCon on tuotenimenä säilynyt näihin päiviin asti ja se on Shcneider Electricin omistuksessa. Dick Morley kuoli 17 lokakuuta 2017. (Laaksonen henkilökohtainen tiedonanto, 29.10.2017)

PLC:n rooli on ajan myötä lisääntynyt automaatiossa ja ne ovat saaneet uusia ominaisuuksia ja käyttökohteita. PLC-järjestelmät ovat erilaisia kooltaan, tiedonsiirroltaan, anturoinniltaan ja ohjauskomponenteiltaan. PLC-järjestelmien koot vaihtelevat muutaman digitaalisen I/O tulon logiikoista aina usean logiikan muodostamaan kokonaisuuteen, joita käytetään esimerkiksi kokonaisen laitoksen ohjaukseen. (Asmala 2005, 5-7)

PLC:ssä ei ole kiinteästi tehtyä johdotusta joka määrittelisi logiikan toiminnan. Ohjelmamuistiin kirjoitetaan ne loogiset toiminnat, jotka halutaan järjestelmässä tapahtuvan. Nämä toiminnat määrittelevät miten logiikkayksikön (CPU) pitää toimia. CPU suorittaa yhden käskyn kerrallaan, mutta se suorittaa koko ohjelman monta kertaa sekunnissa, jolloin se näyttää ulospäin, että asiat tapahtuvat saman aikaisesti. (Ohjelmoitava logiikka 1991, 19-23)



Kuva 2.1. Ohjelmoitavan logiikan perusrakenne. (Asmala 2005, 9)

3 PLC-OHJELMOINTI

PLC:n ohjelmoinnissa käytettävät erilliset ohjelmointilaitteet ovat poistuneet ja nykyisin ohjelmointi tapahtuu yleensä PC-ohjelmalla. PLC-ohjelmointimenetelmät eivät ole yhtä kehittyneitä kuin tavalliset tietokoneohjelmointimenetelmät. Yksi syy tähän on useiden valmistajien erilaiset ohjelmointikielet ja -työkalut. Ratkaisumallit ovat usein riippuvaisia tekijästä ja käytettävistä automaatiotuotteista. Näitä ongelmia on pyritty ratkaisemaan standardoinnin avulla. Yksi keskeisimmistä standardeista on ohjelmointikieliä käsittelevä IEC 61131-3. (Asmala 2005, 5-10)

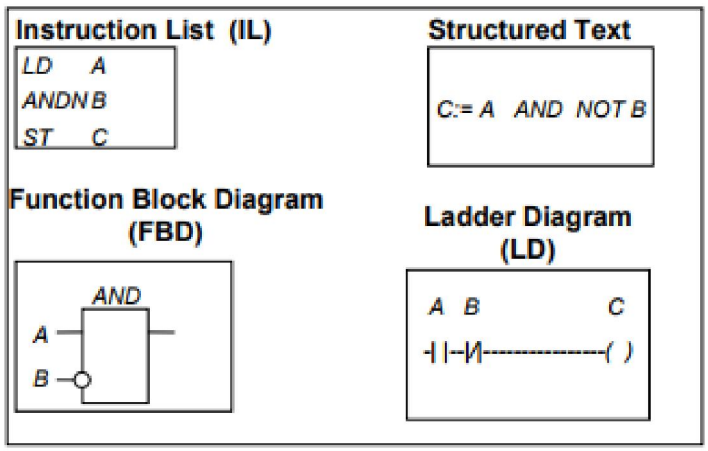
3.1 IEC 61131-3 standardi

IEC 61131-3 standardin ensimmäinen versio on julkaistu vuonna 1993 ja se on päivitetty vuosina 2003 ja 2013. Se käsittelee ohjelmoitavien logiikoiden ohjelmointikieliä ja se on yleisesti hyväksytty laitevalmistajien keskuudessa. (Asmala 2005, 10)

Standardi voidaan jakaa kahteen pääosaan, yleiset elementit (Common Elements) ja ohjelmointikielet (Programming Languages). (PLCopen 2017, 1)

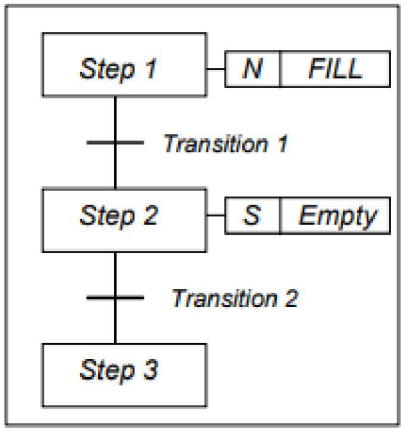
Ensimmäisessä osassa käsitellään ohjelmointikielien yleisiä elementtejä, joiden avulla logiikkaohjelmat muodostetaan. Yleisiä elementtejä ovat esimerkiksi data tyypit, muuttujat, kokoonpanot, resurssit, tehtävät ja rakenneyksiköt (Program Organisation Units, POUs). (PLCopen 2017, 1-3)

Standardin mukaisia ohjelmointikieliä ovat käskylista (Instruction List, IL), rakenteinen teksti (Structured Text, ST), tikapuukaavio (Ladder Diagram, LD) ja toimilohkokaavio (Function Block Diagram, FBD). (PLCopen 2017, 1-3)



Kuva 3.1. Sama yksinkertainen ohjelmaosa kaikilla neljällä ohjelmointikielellä.

Standardi määrittelee myös sekvenssikaavion (Sequential Function Chart, SFC). Se kuvaa graafisesti ohjelman sekvenssin mukaista käyttäytymistä. SFC koostuu vaiheista, jotka yhdistävät toimintalohkot ja siirtymät. Jokainen elementti voidaan ohjelmoida SFC:n sisään millä tahansa IEC 61131-3 mukaisella ohjelmointikielellä. SFC:n vaiheet voivat olla vaihtoehtoisia tai rinnakkaisia toisilleen. (PLCopen 2017, 2)



Kuva 3.2. Sekvenssikaavio.

IEC 61131 standardin muut osat ovat:

1. Yleisinformaatio (General information)
2. Laitevaatimukset ja testit (Equipment require and test)
3. Ohjelmointi kielet (Programming languages)
4. Käyttöohjeistus (User guidelines)
5. Tiedonsiirtoliikennöinti (Communications)
6. Toiminnallinen turvallisuus (Functional safety)
7. Sumea säätö (Fuzzy control programming)

8. Ohjelmointikielten käyttö ja soveltaminen (Guidelines for the application and implementation of programming languages)

3.2 Muita ohjelmointiin liittyviä standardeja ja oppaita

Ohessa kerrotaan lyhyesti muutamista standardeista ja oppaista, jotka liittyvät ohjelmoinnin laatuun ja tähän opinnäytetyön aiheeseen.

IEC 25010 standardi (entinen IEC 9126). Standardin tavoitteena on määritellä yhdenmukaiset mittaustavat ohjelmistojen laatuvaatimuksille. (Itris Automation, 2017a, 7)

GAMP - opas (Good Automated Manufacturing Practice), joka on lääketeollisuuden opas automaatiojärjestelmien valmistuskäytännöistä. Uusin opas (GAMP5) julkaistiin vuonna 2008. (Itris Automation, 2017a, 7)

PLCopen Coding Guidelines Version 1.0 on opas, joka sisältää PLCopen-työryhmän kokoaman PLC-ohjelmoinnin hyvien käytäntötapojen mukaisia ohjeita. Opas käsittelee muun muassa nimeämiseen, kommentointiin, koodauskäytäntöihin ja ohjelmointikieleen liittyviä sääntöjä. (PLCopen, 2016)

3.3 PLC-ohjelmien laadunvalvonta

Mitä myöhemmin ohjelman virheet löytyvät sitä kalliinmaksi sen korjaaminen tulee. Ohjelmien virheet, kuten kommenttien puuttumiset, elementtien huonot nimeämiset, muuttujien kirjoittaminen useampiin tehtäviin, virheelliset kopiointi/liitä toiminnot, kuolleet koodit, käyttämättömät elementit huonontavat merkittävästi ohjelman ylläpitävyyttä ja luettavuutta. (Itris Automation, 2017a, 5-9)

Standardoinnin myötä PLC-ohjelmointialustat ja ohjelmointitavat ovat yhdenmukaistuneet, mutta samanaikaisesti PLC-ohjelmat ovat muuttuneet yhä monimutkaisimmiksi ja kriittisimmiksi, ja ohjelmien laadunvalvonta on tullut vaikeammaksi. Manu-

aalinen tarkastuksen merkitys ja täsmällisyys riippuvat suuresti vastuuhenkilöstä. Kattava manuaalinen tarkistus monimutkaisissa ja suurissa ohjelmissa on työlästä ja kallista, jopa mahdotonta koodin suuren määrän vuoksi. (Itris Automation, 2015)

Tähän kalliiseen ja aikaa vievään manuaaliseen ohjelmien laaduntarkkailuun on ranskalainen ohjelmistotalo Itris Automation kehittänyt automaattisen tietokonepohjaisen PLC-ohjelmien analysointityökalun PLC Checkkerin. Tämän avulla käyttäjät voivat tarkistaa nopeasti ja helposti PLC-ohjelmien koodaussääntöjen mukaisen laadun. (Itris Automation, 2015)

4 PLC-OHJELMAN ANALYSOINTITYÖKALU (PLC CHECKER)

4.1 Mikä on PLC Checker

PLC Checker kuuluu ohjelmatuoteperheeseen, johon sisältyy myös PLC DocGen, PLC-ohjelmien dokumentointigeneraattori ja versiohallinta työkalu, sekä PLC Converter, jonka avulla voidaan kääntää PLC-ohjelmia toisiin ohjelma-alustoihin. (Itris Automation:n www-sivut 2017)

PLC Checker on staattinen pilvipohjainen analysointityökalu, jonka avulla voidaan tarkistaa PLC-ohjelmien laatu ja säästää aikaa. Ohjelma tarkistaa automaattisesti PLC-koodin ennalta määrättyjen sääntöjen pohjalta. (Itris Automation:n www-sivut 2017)

PLC Checker mahdollistaa ohjelmointivirheiden havaitsemisen ja standardien noudattamisen ohjelman tuottaman yksityiskohtaisten raportoinnin avulla. Näin mahdolliset virheet saadaan korjattua mahdollisimman aikaisessa vaiheessa. Raporttien avulla voidaan myös osoittaa sidosryhmille ohjelman laatu. (Itris Automation, 2015)

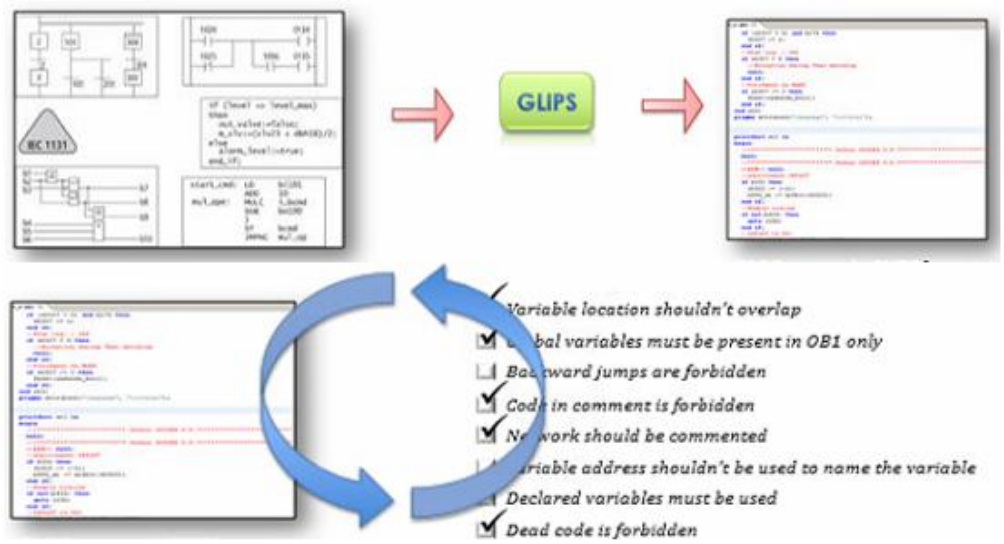
PLC Checker mahdollistaa toimitettavan PLC-ohjelman päivittäisen tarkkailun. Useamman ohjelmoijan tai alihankintana hankittujen PLC-ohjelmien tarkkailu onnistuu helposti PLC Checkerin tuottaman visuaalisten raporttien avulla. (Itris Automation:n www-sivut 2017)

PLC Checkerin avulla PLC-ohjelmien ohjelmoinnin siirtäminen kokeneimmilta ohjelmoijilta kokemattomammille ja aloitteleville ohjelmoijille, sekä käyttö-/ylläpitokustannuksien optimointi ja ohjelman kääntäminen toiselle ohjelmointialustalle tulee olemaan helpompaa ohjelman kehitys ja testaus ja ylläpito vaiheessa paremman koodin luettavuuden ja johdonmukaisuuden johdosta. (Itris Automation:n www-sivut 2017)

Ohjelmistojen turvallisuusstandardit suosittelevat staattisten analysointityökalujen, kuten PLC Checkerin käyttämistä varmistaakseen ohjelman vakauden ja turvallisuuden. (Itris Automation:n www-sivut 2017)

4.2 Kuinka PLC Checker toimii

PLC Checker kääntää IEC 61131-3 standardin mukaiset ohjelmointikielet Itris Automationin kehittämään ylemmän tason ohjelmointikielelle niin sanotulle Glips-kielelle. Tämän jälkeen ohjelma tarkistaa ennalta määriteltyjen sääntöjen pohjalta PLC-ohjelman ja tuottaa siitä monenlaisia hyödyllisiä raportteja. (Itris Automation, 2015)



Kuva 4.1: IEC 61131 ohjelmointi kääntäminen GLIPS kielelle

4.3 PLC Checkkerin tuetut PLC-ohjelmointialustat

PLC Checkerillä pystyy analysoimaan useiden eri brändin PLC ohjelma-alustoja. Tuettavien alustojen määrä kasvaa jatkuvasti. (Itris Automation:n www-sivut 2017)

Tuettuja ohjelma-alusta malleja ovat tällä hetkellä:

- Siemens Step 5, Step7 ja Tia Portal
- Rockwell Automation RSLogix5 (Allen Bradley PLC5), RSLogix 500 (Allen-Brandley SLC500, Micrologix) ja RS Logix 5000 (ControlLogix, CompactLogix, …)
- PLCopen XML format
- 3S - Codesys versio 2.3 ja versio 3.2
- Beckhoff TwinCAT 2 ja TwinCAT 3
- Schneider Electric Unity Pro, PL7 Pro, April PB80 ja PB400, Orphee (April 2000,3000,5000, 7000), VPSOFT/EDIDOS (April SMC) ja XTEL (Telemecanique – PL7-3 Series 7)
- Alstom C1 ja Ap 80
- Omron Sysmac Studio 1.0.3
- Mitsubishi Electric GX Works2
- Phoenix Contact PC Worx ja MULTIPROG 5.50
- Yokogawa Stardom
- ICS Triplex ISaGRAF (4.12)

4.4 PLC Checkerin säännöstö

PLC Checkkerin säännöstöllä määritellään mitä ohjelmalla analysoidaan. Sillä voidaan etsiä virheitä tai informaatiota ohjelmasta, kuten ohjelman monimutkaisuuden mittausta, ulostulojen ja lähtöjen määrää. PLC Checkerissä on nyt kolme yleistä säännöstöä. Ohjelmaan voidaan räätälöidä yrityksille, heidän omien ohjelmointitapojen mukainen säännöstö.

Ensimmäinen säännöstö on PLC Checkerin oletusarvoinen säännöstö, jota käsitellään kohdassa 4.5

Toinen säännöstö perustuu GAMP5 oppaan mukaisiin ohjelmointikäytäntöihin.

Uusin säännöstö perustuu PLCopen järjestön kokoamaan koodaus oppaaseen (PLCopen Coding Guidelines), joka sisältää 64 hyvän ohjelmointikäytännön mukaista sääntöä. Tämän avulla käyttäjä voi suoraan tarkistaa PLC Checker-ohjelmalla, että PLC-ohjelma täyttää PLCopenin säännöt. (Itris Automation, 2017c)

4.5 Oletusarvoinen säännöstö.

PLC Checkerin oletusarvoinen säännöstö on luotu ohjelmointi standardien ja yleisesti havaittujen ja kokemusten perusteella saatujen tietojen pohjalta. Sääntöjä voi muokata, lisätä tai poistaa asiakkaan tarpeiden mukaan.

Säännöt on luokiteltu kategorioihin: nimeäminen (Naming), kommentointi (Comments), kirjoitus (Writing), rakenne (Structure), informaatio (Information) ja kieli (Language).

Ohjelman säännöt ovat myös luokiteltu virheen vakavuusasteen mukaan: Kohtalokas (Fatal), virhe (Error), varoitus (Warning) ja informaatio (Info). Virheen vakavuusaste on myös mahdollista muuttaa ohjelmassa.

Alla listaus oletusarvoisesta säännöstöstä. Jokainen sääntö on vielä jaettu alasääntöihin. Kaikki säännöt ja alasäännöt löytyvät liitteestä 1.

4.5.1 Nimeäminen (Naming)

Tämän sääntökategorian tarkoituksena on varmistaa, että elementit kuten ohjelma, moduulit, muuttujat, toimilohkot (FB), organisaatioyksiköt (OB), SR:t, rutiinit ja lisäosat noudattavat nimeämissääntöjä, jotka takaavat ohjelman luettavuuden, ylläpitävyyden. (Itris Automation, 2015b)

N1 Kaikki ohjelman elementit pitäisi olla nimetty.

- Kun kaikki ohjelman elementit on nimetty, ylläpito on helpompaa ja ohjelmasisältö on helpompi ymmärtää. (Itris Automation, 2015b)

N2 Elementin nimen pituus pitäisi olla annetussa rajoissa.

- Elementtien nimet tulee sisältää vähintään 3 merkkiä ja enintään 15 merkkiä. (Itris Automation, 2015b)

N3 Topologista fyysistä osoitetta ei pitäisi käyttää nimeämisen osana.

- Nimeäminen ei saa viitata niiden fyysiseen osoitteeseen (esimerkiksi “MW2”, “FB4”…). Viittaus koodin fyysiseen osoitteeseen huonontaa luettavuutta ja voi aiheuttaa vaarallisen sekaannuksen. (Itris Automation, 2015b)

N4 Testien aikaisia apumuuttujien nimeämiset ei pitäisi olla osa lopullista ohjelmaa.

- Vaikka testien aikaisien apumuuttujien nimeämiset ovat hyödyllisiä kehityksen aikana, eivät ne saa olla osa lopullista ohjelmaa. Esimerkiksi nimet, kuten testaus (test), poistettava (to be removed) ovat kiellettyjä, sillä nämä antavat kuvan, että ohjelma ei ole vielä valmis. (Itris Automation, 2015b)

N5 Erikois tai graafiset merkit ovat kiellettyjä nimeämisessä.

- Erikoismerkkien (ä, ö, å., à, è…) käyttö voi aiheuttaa ongelmia siirrettävyydestä seuraaviin ohjelmaversioihin ja/tai muihin alustoihin. Ainoastaan alfanumeeriset merkit A-Z, a-z 0-9 ja alaviiva ovat sallittuja (Itris Automation, 2015b)

N6 Ohjelmoinnin avainsanoja ei pitäisi käyttää nimeämisessä.

- Ohjelmointikielen avainsanoja (esimerkiksi “if”, “set”, “start”) ei pidä käyttää nimeämisessä. Avainsanojen käyttö nimeämisessä huonontaa ohjelman luettavuutta ja siirrettävyyttä. (Itris Automation, 2015b)

4.5.2 Kommentointi (Comments)

Hyvien nimeämiskäytäntöjen lisäksi on tärkeää noudattaa sääntöjä kommentoidessa ohjelmia. Mitä tarkemmat kommentit ovat, sitä ymmärrettävämpi ohjelmointi koodi on. (Itris Automation, 2015b)

C1 Kaikki ohjelman elementit pitäisi olla kommentoitu.

- Kaikki ohjelman elementit pitää kommentoida. Jos kaikki elementit kommentoidaan hyvin, niin ohjelman lukeminen ja ymmärtäminen on paljon helpompaa ja näin ohjelman ylläpito on halvempaa. (Itris Automation, 2015b)

C2 Kommentin pituus pitäisi ylittää annetun merkkien määrä.

- Ohjelman elementtien kommenttien on sisällettävä vähimmäismäärä merkkejä (oletusarvo: muuttujat 7, koodit 15), jotta vältettäisi huonoja kommentteja. Vähimmäispituus varmistaa, että kommentit ovat ymmärrettäviä. (Itris Automation, 2015b)

C3 Jokainen virtapiiri (network) pitäisi olla kommentoitu.

- Virtapiirit täytyy kommentoida. Tämä sääntö on tarkoin määrätty IEC 61131 standardissa. Hyvien kommenttien lisäksi on tärkeä varmistaa, että kommentit jakautuvat oikeaan paikkaan ohjelmassa. (Itris Automation, 2015b)

C4 Kommentit ei pitäisi sisältää kommentin aloitus merkkejä.

- Kommentin aloitus merkkien läsnäolo johtuu usein unohdetusta ohjelman loppumerkinnästä, joka voi johtaa siihen, että osa ohjelmasta jää kommentin sisään ja siten ohjelman osa ei toimi. (Itris Automation, 2015b)

4.5.3 Kirjoitus (Writing)

E1 Kaikki muuttujat pitäisi olla kirjoitettu ennen niiden lukemista.

- Lukuun ottamatta fyysisiä ja viestintätuloja ja järjestelmän muuttujia, kaikki muuttujat tulisi kirjoittaa ennen niiden lukemista PLC-syklin aikana. Muuttuja joka luetaan ennen kirjoittamista, voi lisätä ohjelman viivettä. (Itris Automation, 2015b)

- Joissain tapauksissa on perusteltua jättää tämä sääntö huomioimatta
    - tietokannassa alustetut muuttujat
    - muuttujat, jotka on päivitetty tietoliikenneyhteydellä

- tilamuuttujat, jotka tallentavat aiemman PLC-syklin arvon (Itris Automation, 2015b)

E2 Toimilohkojen (FB) parametreja pitäisi käyttää oikein.

- Tämä sääntö on jaettu kolmeen alasääntöön
  - määritellyt tulo parametrit (inputs) tulee lukea mutta ei kirjoittaa
  - määritellyt tulo/lähtö parametrit (inputs/outputs) tulee lukea ja kirjoittaa
  - määritellyt lähtö parametrit (outputs) pitäisi kirjoittaa mutta ei lukea
- Virhe viestit:
  - E2a – tulo parametria ei ole koskaan luettu
  - E2b – tulo parametri on kirjoitettu
  - E2c – tulo/lähtö parametria ei ole luettu
  - E2d – tulo/lähtö parametria ei ole kirjoitettu
  - E2e – lähtö parametria ei ole kirjoitettu
  - E2f – lähtö parametri on luettu
- Käyttämättömät parametrit ovat osoitus keskeneräisestä koodista tai koodista joka sisältää tarpeettomia elementtejä. Joissakin PLC-laitteissa väärinkäytetyt parametrit saattavat aiheuttaa ei toivottuja tai jopa vaarallisia vaikutuksia, kuten datan korruptiota. (Itris Automation, 2015b)

4.5.4 Rakenne (Structure)

S1 Palaaminen jo suoritettuun ohjelmariviin on kielletty.

- Palaaminen jo suoritettuun ohjelmariviin voi laukaista sisäisen silmukan, joka voi johtaa PLC:n pysähtymiseen. Yleensä palaaminen tekee koodista vaikeammin ymmärrettävän ja siksi niiden ylläpito on monimutkaisempaa. (Itris Automation, 2015b)

S2 Muuttujaa pitäisi käyttää vain yhdessä virtapiirissä (network).

- Yksinkertainen ja selkeä koodirakenne helpottaa koodin ymmärtämistä ja sen ylläpitoa. Lisäksi useasti käytetty muuttuja voi johtaa koodin virheelliseen käyttäytymiseen. (Itris Automation, 2015b)

S3 Muuttuja pitäisi olla kirjoitettu vain yhdessä toiminnassa (FC).

- Muuttujan kirjoittaminen useissa toiminnassa johtaa ei sovittuun käyttäytymiseen. (Itris Automation, 2015b)

S4 Fyysinen ulostulo pitäisi kirjoittaa vain kerran PLC-syklin aikana.

- Moninkertainen kirjoitus voi aiheuttaa luotettavuusongelmia. Myös koodi, jossa lähdöt kirjoitetaan useammin kuin kerran, on vaikeampi ymmärtää ja ylläpitää. On huomioitava, että kirjoittaminen silmukkaan pidetään moninkertaisena kirjoituksena, ja siksi se on tämän säännön rikkomista. (Itris Automation, 2015b)

S6 FB tapahtumia pitää käyttää vain kerran samoilla parametreilla.

- Toimintalohko omistaa sisäiset muuttujansa, jotka tallentavat FB:n tilan kahden peräkkäisen syklin välillä. Jokainen FB käsittely pitäisi muuttaa näitä muuttujia osoittamaan tilan kehitystä. Kun se suoritetaan useammin kuin kerran samoilla parametreilla, näyttää siltä, että sykli etenee kaksi kertaa nopeammin kuin muissa tapauksissa. Usein kun FB:n tapahtumia käytetään useammin kuin kerran samoilla parametreilla, niin se on seurausta kopioimisesta/liittämisestä, jota ei ole päivitetty. Tämä voi johtaa vikoihin, joita on vaikea löytää. (Itris Automation, 2015b)

S7 Määriteltyjä muuttujia pitäisi käyttää.

- Määriteltyjä muuttujia pitäisi käyttää, lukuun ottamatta varalla olevia muuttujia. Varalla olevia muuttujia ei saa käyttää. Määrittelemättömät muuttujat (paitsi varalla olevat) ja käyttämättömät FB:t ovat usein merkki unohtuneesta käsittelystä tai epätäydellisestä tietokannan puhdistamisesta. Kun varalla olevaa muuttujaa on käytetty, on se merkki unohtuneesta uudelleennimeämisestä. (Itris Automation, 2015b)

S8 Muuttujien sijainti ei saa olla päällekkäinen.

- Päällekkäiset muuttujat on tarkistettava. Paikalliset muuttujat, jotka ovat päällekkäisiä ovat samanarvoisia toistensa välillä. Tämä on joskus vaikeasti luettava ohjelmakoodi. Se on myös joskus virhe paikan määrittämisessä. (Itris Automation, 2015b)

S9 Monimutkaisuuden mittaus.

- Koodin monimutkaisuus on arvioitava. Silmukat, hypyt ja koodin ylätason limitys ovat kriteereitä koodin monimutkaisuudesta. Osa ohjelmaa on monimutkaisempaa kuin toiset osat. Tämä on erityisen yleistä, kun koodin jatkuvuus katkeaa. Tämä sääntö etsii syitä jatkuvuuden katkeamiselle kuten silmukoita, palaamisia jo suoritettuihin ohjelmariveihin ja ylätason käskyjen monimutkaisuuksia. (Itris Automation, 2015b)

S10 SCADA rajoitukset.

- Scada-ohjelmistot eivät tue rakenteellisia tietotyyppejä. Tallaisia rakenteita pitää välttää. Kun käytetään SCADA:a, joka ei tue rakenteellisia tietotyyppejä, on tärkeää olla tekemättä PLC-tietokantaa, joka perustuu näihin rakenteisiin. Tätä sääntöä käytetään tunnistamaan tällaiset rakenteet. (Itris Automation, 2015b)

S12 Ehto lause ilman oletusarvo muuttujaa.

- Joissakin standardeissa voidaan pyytää aina lisäämään oletusarvo muuttuja ehtolauseeseen (IF, CASE, …). Tämän säännön tavoitteena on varmistaa, että jokaiselle IF-käskylle on olemassa ELSE-käsky ja että jokaiselle CASE-käskylle on olemassa OTHERS-käsky. Näin ohjelma varmistaa, että kaikki tapaukset on otettu huomioon ohjelmassa, vaikka se olisikin “ELSE NULL” tapaus. (Itris Automation, 2015b)

S13 SFC diagrammit ilman alustusaskelta.

- SFC diagrammit ilman alustusaskelta olisi yksilöitävä selvästi. Tällaisia diagrammeja käynnistetään yleensä ohjelman muissa osissa. On vaikeampaa ymmärtää diagrammin käyttäytymistä, mikä vähentää sovelluksen ylläpitämistä. (Itris Automation, 2015b)

S15 Koodissa ei saa käyttää välittömiä arvoja.

- Välittömiä arvoja tai vakioita ei tulisi käyttää koodissa. Käyttämällä koodeissa välittömiä arvoja tai vakioita saadaan siitä vähemmän tietoja kuin selkeästi hyvin nimetyn vakion käyttämisestä. Kuvassa 4.2 on esimerkki tästä säännöstä. (Itris Automation, 2015b)

Kuva 4.2: Esimerkki S15 säännön käyttämisestä.

4.5.5 Informaatio (Information)

I1 Kommenttien suhdeluku.

- On suositeltavaa, että ohjelma sisältää vähimmäismäärän kommentteja. PLC Checker suosittelee 30% suhdetta (kommentoidut käskyt/käskyjen kokonaismäärä). Tällainen kommenttien vähimmäismäärä varmistaa, että ohjelma ottaa huomioon koodin ylläpitävyyden ja että se on helposti ymmärrettävissä, myös tulevissa kehitysvaiheissa. (Itris Automation, 2015b)

I2 Koodi kommenttien sisällä.

- On tärkeä tarkistaa koodin esiintyminen kommenttina. Koodi kommenttien sisällä on tyypillinen testaus- ja korjausstrategia, mutta lopullisessa ohjelmassa ei pitäisi olla koodia kommentin sisällä, koska se vaikeuttaa ohjelman luettavuutta. (Itris Automation, 2015b)

I3 Kuollut koodi.

- Kuollut koodi vaikuttaa ohjelman luotettavuuteen ja ylläpidettävyyteen ja voi olla toiminnallisen ongelman syy. Huomioitava, että kun PLC Checker havaitsee epäsuosittuja toimintoja ja yksittäisiä koodin osia, virheiden täydellistä havaitsemista ei voida taata. (Itris Automation, 2015b)

I4 Lista lukituista elementeistä.

- PLC Checker ei tarkista lukittuja elementtejä, vaan ne on tarkistettava manuaalisesti. Tämä sääntö on sitä varten että, sovelluksen käyttäjä löytää lukitut koodit ja saa tietoa niiden sijainnista. (Itris Automation, 2015b)

I5 Kompleksisuusarvo.

- Kompleksisuusarvo antaa tietoa monimutkaisuudesta. On tärkeä pitää se kohtuullisen alhaisena. Kompleksisuusarvo laskee eri polkujen määrän, jotka voidaan suorittaa virtapiirissä. Mitä suurempi arvo on, sitä vaikeampi on vahvistaa virtapiirin oikea käyttäytyminen. Tällaisissa tapauksissa virtapiiri on jaettava aliohjelmiin niin, että jokainen niistä on kompleksisuusarvon kynnysarvon alapuolella. (Itris Automation, 2015b)

I6 Kompleksisuudenrakenne.

- Kompleksisuudenrakenne antaa tietoa rakenteettomista käskyistä. On tärkeä pitää se kohtuullisen alhaisena. Kompleksisuudenrakenne laskee eri rakenteettomien käskyjen määrän virtapiirissä. Mitä suurempi luku on, sitä vaikeampi on vahvistaa virtapiirin oikea käyttäytyminen. Suurin osa rakenteettomista käskyistä voidaan korvata jäsennellyillä käskyillä. (Itris Automation, 2015b)

I7 Koodirivin funktion kompleksisuus.

- Koodirivin funktion kompleksisuus laskee käskyjen määrän rakenneyksiköissä POU:ssa. On tärkeää pitää se hallinnassa. Hyvin suuri prosessi tai toimintalohkot voivat olla vaikeita testata, lukea ja ylläpitää. Voi olla parempi jakaa erittäin suuri prosessi tai toimintalohko kohtuullisemman kokoiseksi. (Itris Automation, 2015b)

I8 SFC toimenpidelaskuri

- Tämä toiminta laskee tietyn SFC:n toimenpiteiden määrän. Tätä lukua voidaan käyttää kompleksisen SFC:n havaitsemiseen ja se antaa yleiskatsauksen SFC:n suhteellisesta koosta. (Itris Automation, 2015b)

I9 SFC ohjelmahaarojenlaskuri.

- Tämä sääntö laskee kaikkien ohjelmahaarojen määrän tietyssä SFC:ssä. Tämä määrä antaa tietoa SFC:n monimutkaisuudesta. (Itris Automation, 2015b)

I10 Kopio/Liitä suhdeluku.

- Tämä suhdeluku havaitsee koodin, joka on tarpeeton. Kopioidun koodin uudelleen käyttäminen ei ole hyvä käytäntö. Kunnossapito ja kehitys tulee olemaan kalliimpaa, koska muutokset ja korjaukset tulee tehdä jokaiseen kopioituun koodiin. Lisäksi manuaalinen kopiointi on virheellinen tapa, koska suorittamisen jälkeen koodi on muutettava. Kun koodia ei muuteta oikein, se tuo esille virheitä joita on vaikea löytää. (Itris Automation, 2015b)

4.5.6 Kieli (Language)

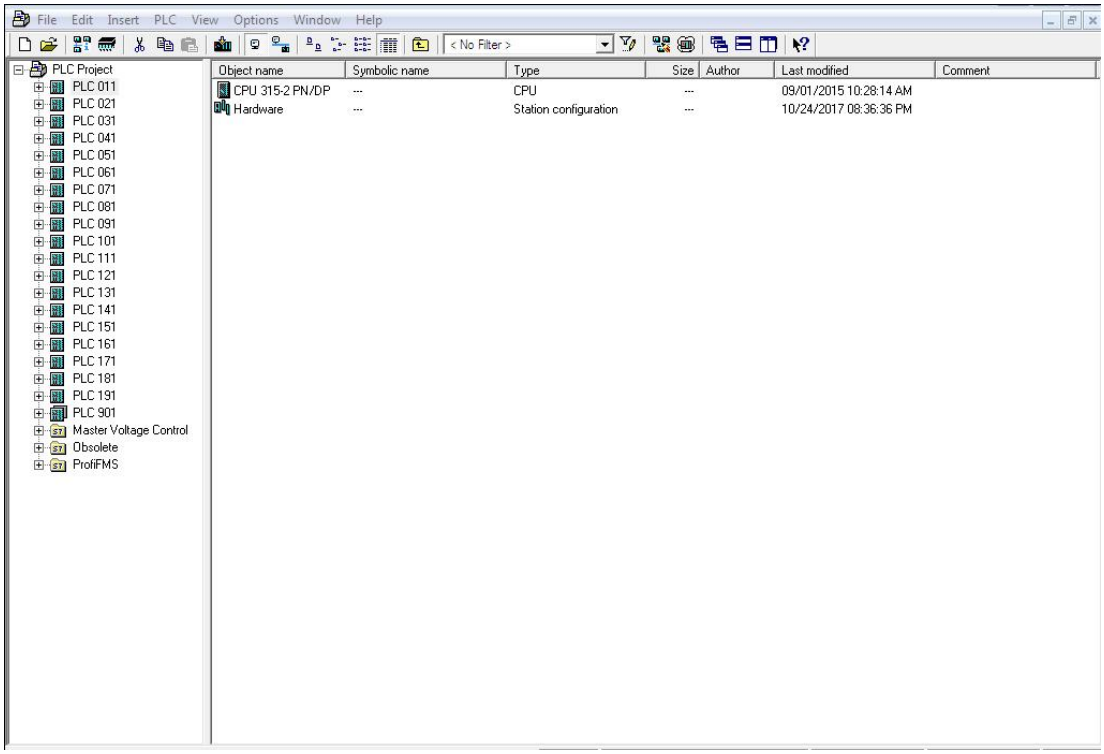
L1 IL (Instruction List) kieli on kielletty.

- IL kieli on matalantasonkieli ja sitä on vaikea lukea ja ymmärtää. Siksi on suositeltavaa välttää sitä niin paljon kuin se on mahdollista.
- Ohjelmointitavasta riippuen voi olla vaikea välttää IL kieltä. Esimerkiksi Siemens Step7 PLC:n avulla kaikki tikapuukaaviot (LD) ja toimintalohkokaaviot (FBD) tallennetaan IL kieleen. Muiden PLC-järjestelmien, kuten Schneider ja Rockwellin, kanssa on paljon helpompi välttää IL kielen käyttöä. (Itris Automation, 2015b)

5 PLC-OHJELMAN ANALYSOINTI PLC CHECKER-OHJELMALLA

5.1 Analysoitava PLC-ohjelma

Analysoitavaksi ohjelmaksi valittiin useamman logiikan muodostaman kokonaisen laitoksen PLC-ohjelmisto, joka sisältää 20 Siemens 300 tai 400 sarjan logiikkaa. Ohjelmat on ohjelmoitu Step7 ohjelma-alustalla. Ohjelma on identtinen PLC 010 ja – PLC 191, joten opinnäytetyössä käsitellään lähinnä PLC 901 ja PLC 021 logiikoiden ohjelmistoja ja tuloksia. PLC-ohjelmisto on salassa pidettävää tietoa ja tästä syystä ohjelmasta kerrotaan hyvin vähän, eikä se ole tämän opinnäytetyön kannata oleellista-kaan.



Kuva 5.1. PLC ohjelma Step7 ohjelma-alustalla

PLC 901 on laitoksen yleis/tuki PLC, joka kerää yleistietoa laitoksen tilatiedoista ja valvoo tukiprosesseja ja kommunikoi ulospäin esimerkiksi laitoksen ylläpitäjille.

PLC 011 – PLC 191 ovat ohjattavien laitteiden automaatio logiikoita, jotka kommunikoivat PLC 901 kanssa laitoksen tilasta ja hoitavat itsenäisesti ohjattavien laitteiden toimintaa ja turvallisuutta.

5.2 Ohjelman valmisteluvaihe

Ennen varsinaisen PLC-ohjelman analysointia täytyy luoda ensin joitakin lähdekoodeja. Analysoitaessa Step7 ohjelma-alustalla ohjelmoitua ohjelmaa täytyy:

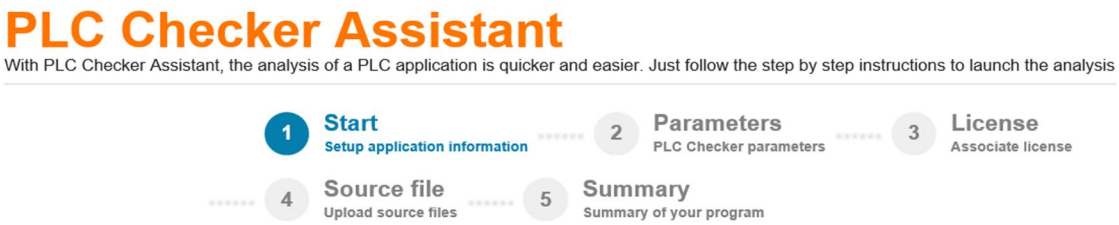
- Luoda ja viedä ohjelman symbolit .ASC tiedostoon käyttäen saksankielisiä lyhenteitä.
- Luoda ja viedä kontakti koodit (contact code, CONT) .AWL tiedostoon. (AWL (saksa) = STL (englanti) = käskylistamuoto)
- Luoda ja viedä kaikki sekvenssikaaviot (GRAPH, SFC) .GR7 tiedostoihin.
- Luoda SCL lähdekoodi CFC-koodeista.
- Viedä rakenteinen koodi (SCL) .SCL tiedostoihin

Seikkaperäiset ohjeet oheisten tiedostojen luomiseen löytyvät oppaasta: Export instruction for PLC Checker Step7. (Itris Automation, 2017b, 4-13)

5.3 Ohjelman analysointi.

PLC Checker projektin luominen ja PLC ohjelman analysoiminen on hyvin helppoa ja nopeaa. Prosessissa on seuraavat vaiheet:

1. Asetetaan ohjelman informaatio.
2. Valitaan ohjelma-alusta ja säännöstö.
3. Valitaan lisenssi.
4. Ladataan kohdassa 5.2 luodut tiedostot.
5. Yhteenveto projektista.



Kuva 5.2. PLC Checker analysointi projektin luominen.

Tässä opinnäytetyössä PLC-Ohjelman analysointi suoritetaan oletusarvoisia sääntöjä käyttäen. Jokaisen logiikan ohjelmisto analysoidaan erikseen, ja projektikansio pidetään saman näköisenä kuin se on Step7 ohjelmassa, jotta analysointiprojekti pysyy selkeänä.

Application name	Program key	Creation date	Last exec date	PLC name	Visibility	Used tool(s)
PLC 001	FFKANQ	2017-01-10 14:31:30	2017-09-19 13:37:28	Siemens - Simatic Step7 (S7-300, S7-400, C7)	private	PLC Checker
PLC 011	LIKZYV	2017-01-11 10:11:37	2017-09-23 13:14:50	Siemens - Simatic Step7 (S7-300, S7-400, C7)	private	PLC Checker
PLC 021	RYIDPN	2017-01-11 10:20:33	2017-09-23 13:07:43	Siemens - Simatic Step7 (S7-300, S7-400, C7)	private	PLC Checker
PLC 031	KOXRCM	2017-01-11 10:32:32	2017-01-23 11:02:29	Siemens - Simatic Step7 (S7-300, S7-400, C7)	private	PLC Checker
PLC 041	WCOJXY	2017-01-11 10:39:53	2017-01-23 12:02:23	Siemens - Simatic Step7 (S7-300, S7-400, C7)	private	PLC Checker
PLC 051	QRPGRH	2017-01-11 10:48:42	2017-01-24 10:27:04	Siemens - Simatic Step7 (S7-300, S7-400, C7)	private	PLC Checker
PLC 061	AWKOLR	2017-01-11 10:58:28	2017-01-11 11:02:08	Siemens - Simatic Step7 (S7-300, S7-400, C7)	private	PLC Checker
PLC 071	QDUCXS	2017-01-11 11:12:19	2017-01-11 11:16:16	Siemens - Simatic Step7 (S7-300, S7-400, C7)	private	PLC Checker
PLC 081	ULYTJR	2017-01-11 11:21:03	2017-01-11 11:25:03	Siemens - Simatic Step7 (S7-300, S7-400, C7)	private	PLC Checker
PLC 091	QJLAZI	2017-01-11 11:30:52	2017-01-11 11:35:02	Siemens - Simatic Step7 (S7-300, S7-400, C7)	private	PLC Checker

Showing 1 to 10 of 20 entries

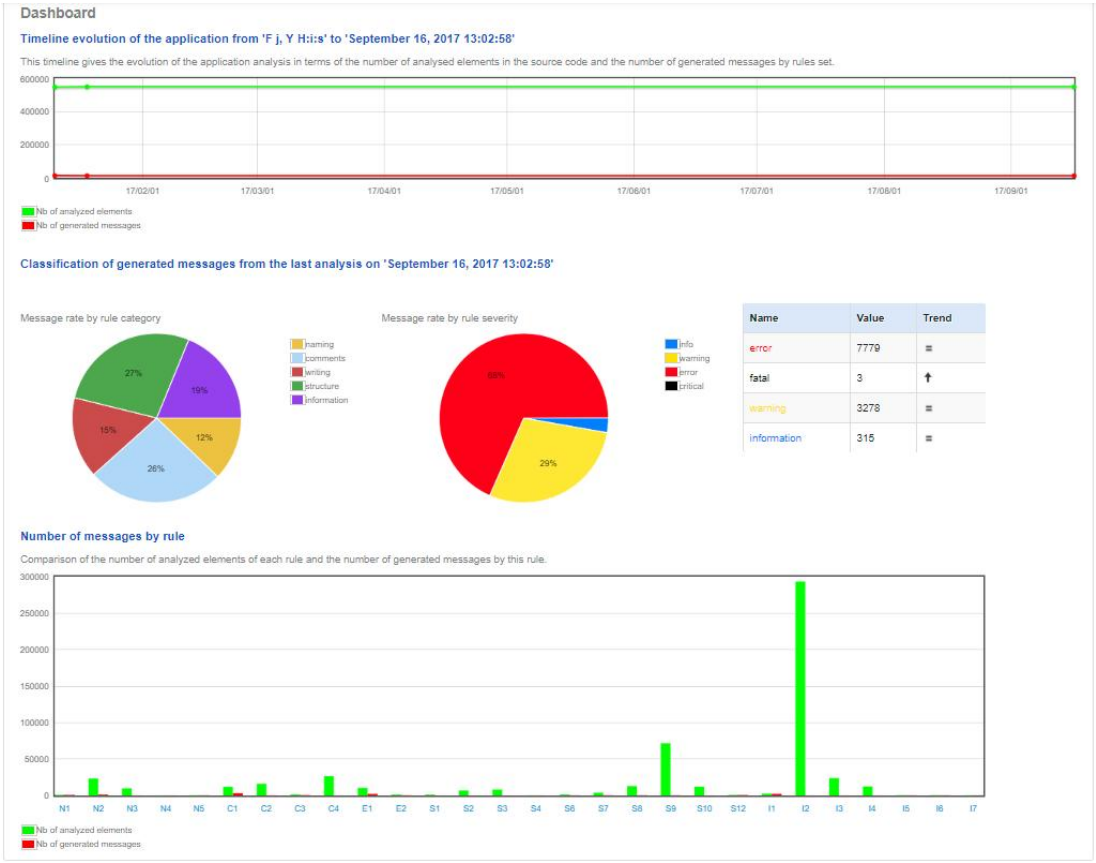
Previous 1 2 Next

Kuva 5.3. PLC Checker projekti.

5.3.1 Analysointi raportit

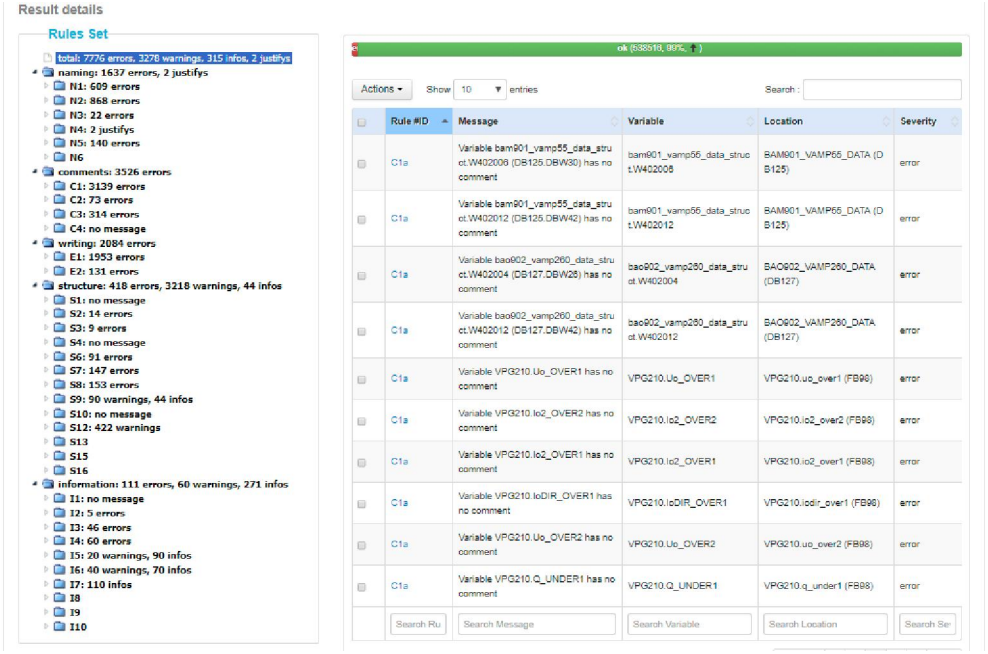
Suuren ja monipuolisenkin logiikkaohjelmiston analysointi tulokset saadaan PLC Checker-ohjelmalla muutamassa minuutissa. Analysoinnin tuloksena saadaan kolme erilaista raporttia välittömästi.

Ensimmäisessä graafisessa yleiskatsausraportissa kuvassa 5.4 nähdään PLC Checkerillä analysoitujen elementtien määrän ja sen antamien virheviestien määrän suhdetta ajan funktiona. Raportissa kuvataan myös kahden ympyrädiagrammin avulla tulostettujen virheiden suhdelukua eri kategorioissa sekä suhdelukua eri vakavuusasteissa. Taulukossa kuvataan virheviestien määrää eri vakavuusasteissa sekä niiden kehityssuuntaa. Alin kuvaaja näyttää analysoitujen elementtien sekä luotujen virheviestien määrää jokaisessa eri säännössä. Tämä raportin avulla on helppo ja vaivaton tapa seurata kehitteillä olevan ohjelman kehitystä ja sen laatua.



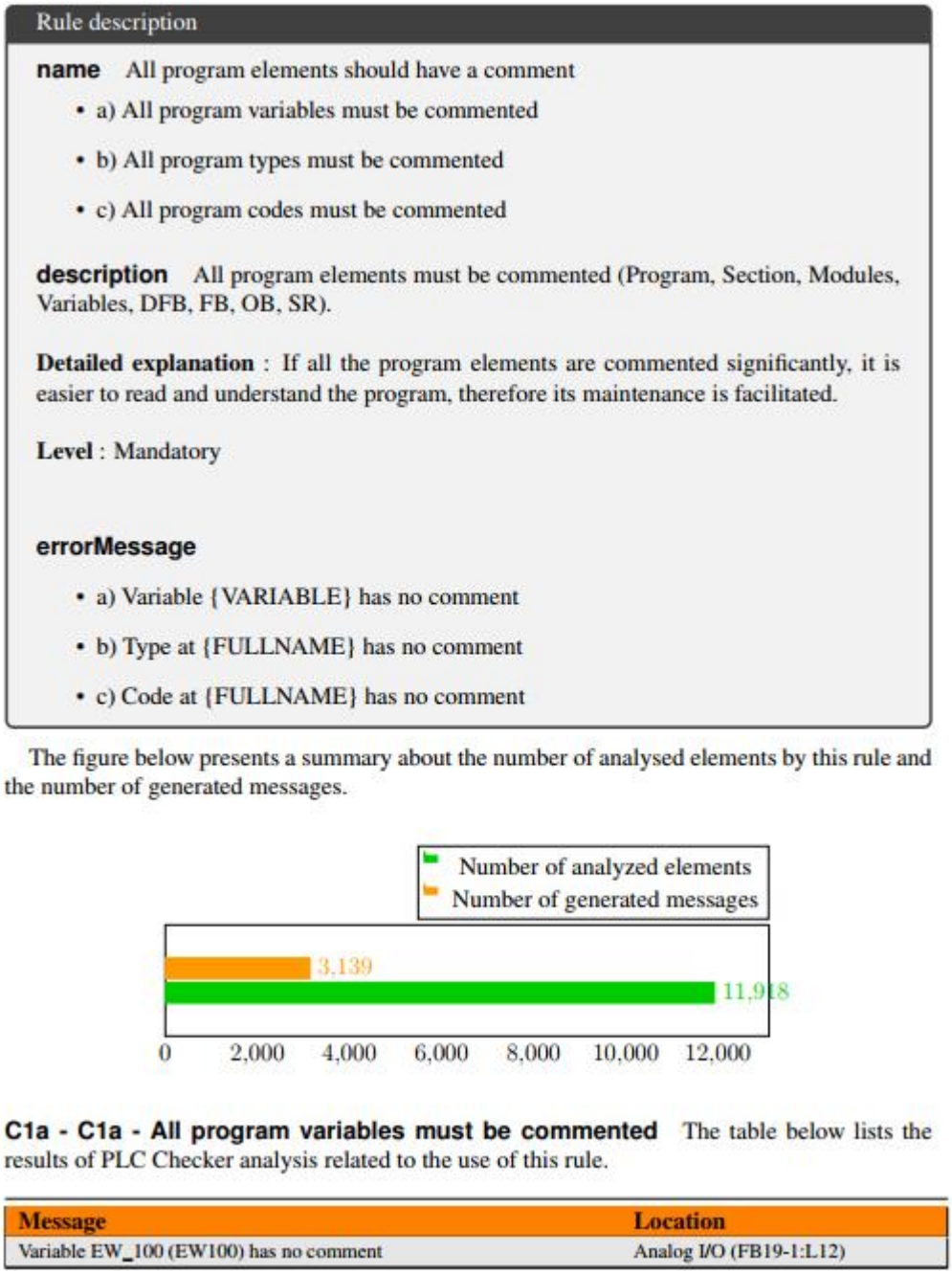
Kuva 5.4. PLC 901 yleiskatsaus.

Yksityiskohtaisessa raportissa (Result details) kuva 5.5 kuvataan hyvin yksityiskohtaisesti ohjelman löytämät virheet, jossa ilmenee rikotun säännön ID, virheviesti, muuttuja, sijainti ohjelmassa sekä virheen vakavuusaste. Tämän raportin avulla on helppo löytää ohjelman säännöstön mukaiset virheet ja korjata ne.



Kuva 5.5. PLC 901 yksityiskohtainen virheraportti.

Kolmas raportti kuva 5.6 on tulostettava PDF raportti, jossa selitetään hyvin seikkaperäisesti muun muassa säännöt, analysoitujen elementtien määrät ja luodut virheviestit. Tämän raportin avulla on helppo osittaa ohjelman kulkua ja laatua projektin sidosryhmille.



Rule description

name All program elements should have a comment

- a) All program variables must be commented
- b) All program types must be commented
- c) All program codes must be commented

description All program elements must be commented (Program, Section, Modules, Variables, DFB, FB, OB, SR).

Detailed explanation : If all the program elements are commented significantly, it is easier to read and understand the program, therefore its maintenance is facilitated.

Level : Mandatory

errorMessage

- a) Variable {VARIABLE} has no comment
- b) Type at {FULLNAME} has no comment
- c) Code at {FULLNAME} has no comment

The figure below presents a summary about the number of analysed elements by this rule and the number of generated messages.

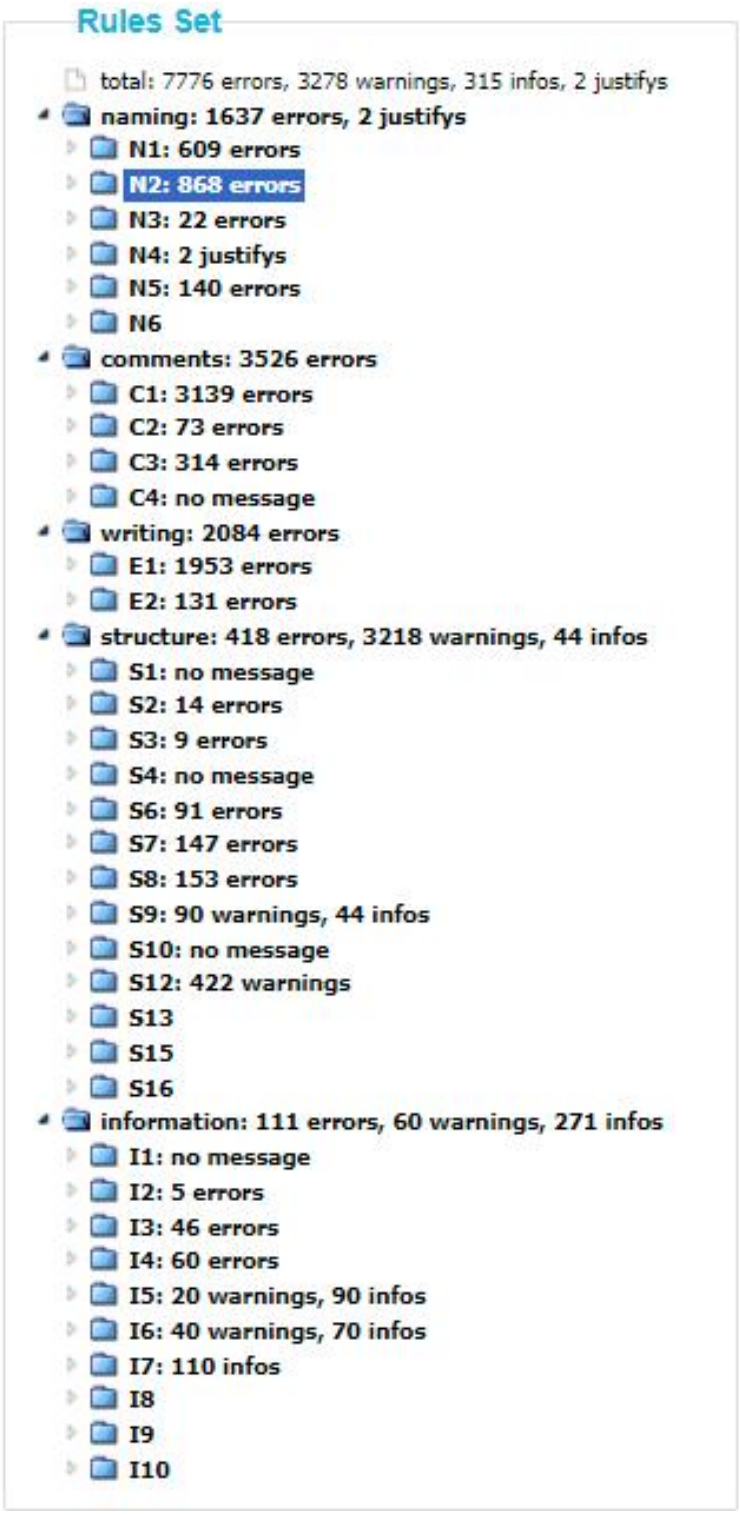
C1a - C1a - All program variables must be commented The table below lists the results of PLC Checker analysis related to the use of this rule.

Message	Location
Variable EW_100 (EW100) has no comment	Analog I/O (FB19-1:L12)

Kuva 5.6. Ote PLC 901 PDF raportista.

5.3.2 PLC 901 analysointitulokset

PLC 901 osalta PLC Checker analysoi yhteensä 549887 elementtiä, joista se loi 11366 virheviestiä. Säännöstön mukaisia elementtejä on noin 98%.



Kuva 5.7. PLC 901 virheviestien määrä sääntöryhmittäin.

5.3.3 PLC 021 analysointitulokset

PLC 021 osalta PLC Checker analysoi yhteensä 398299 elementtiä, joista se loi 8408 virheviestiä. Säännöstön mukaisia elementtejä on myös noin 98%.

Kuva 5.8. PLC 021 virheviestien määrä sääntöryhmittäin.

5.3.4 Tulosten käsittely

Virheviestien suuren määrän vuoksi en ala käsitellä niitä yksityiskohtaisesti. Yksi merkittävä tekijä virhemääriin voi olla se, että ohjelman kehitystyö on aloitettu jo 90 luvulla ja näin ollen näitä oletusarvoisia sääntöjä ei ole ollut alusta asti käytössä/tiedossa. Mutta todettiin, että käytettävä säännöstö on hyvien käytäntötapojen mukainen.

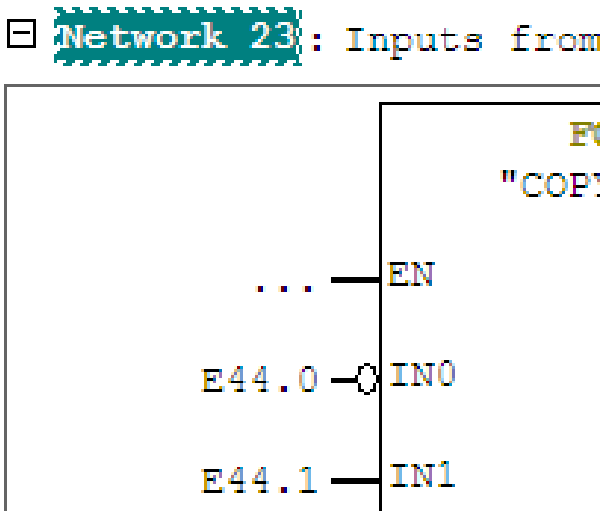
Käydään läpi jokaisesta kategoriasta pari virhettä. Oletusarvoinen säännöstössä kohdassa 4.5 on käyty yksityiskohtaisesti läpi virheviestien merkitykset.

Tässä kohdassa kuva 5.9 on rikottu nimeämissääntöä, N1, kaikki elementit pitäisi olla nimetty, jotta ylläpito olisi helpompaa ja ja ohjelmasisältö on helpompi ymmärtää.

Raportista nähdään hyvin virheen sijainti ohjelmasta esimerkiksi Muuttuja E_44_0 sijaitsee FB20:ssä virtapiirissä (network) 23 rivillä 1

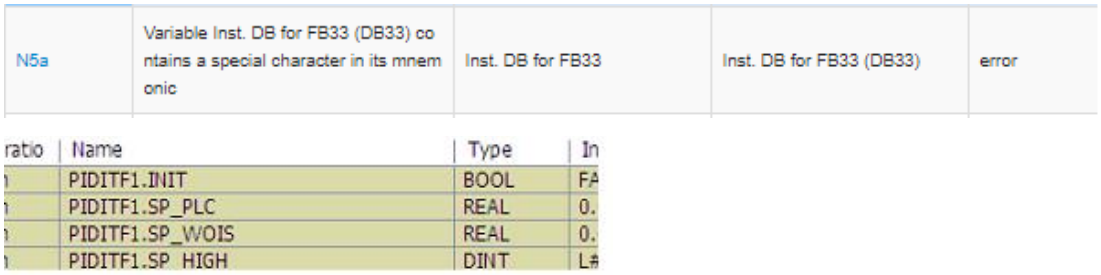
	Rule #ID	Message	Variable	Location	Severity
	N1a	The E_43_7 (E43.7) variable has no mnemonic	E_43_7	DInput control (FB20-22:L46)	error
	N1a	The E_44_0 (E44.0) variable has no mnemonic	E_44_0	DInput control (FB20-23:L1)	error
	N1a	The E_44_1 (E44.1) variable has no mnemonic	E_44_1	DInput control (FB20-23:L4)	error
		The E_44_2 (E44.2) variable has no			

Kuva 5.9. PLC Checkerin antama virhe viesti



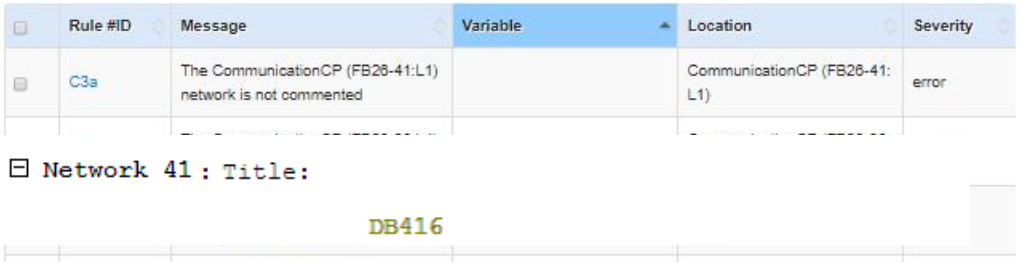
Kuva 5.10 Nimeämisvirheen sijainti PLC ohjelmassa

Tässä kohdassa kuvassa 5.11 on rikottu sääntöä N5. Erikoismerkin tai graafistemmekin käyttö muuttujan nimeämisessä on kielletty. Vain merkit A-Z, a-z, 0-9 ja alaviiva on sallittu. Tässä on käytetty nimeämisessä pistettä, joka ei ole tässä säännössä sallittu. Tätä virhettä on rikottu lähes 100 prosenttisesti, joten voidaan olettaa, että se on ollut käytäntönä ohjelmaa tehdessä. Halutut merkit voidaan lisätä sallittuihin merkkeihin.



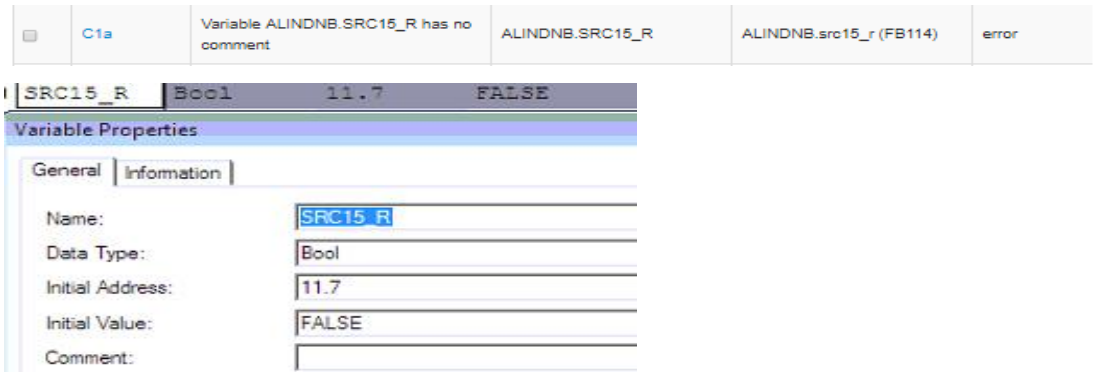
Kuva 5.11: Erikoismerkin käyttövirhe ja sijainti PLC ohjelmassa.

Tässä kohdassa kuvassa 5.12 on rikottu sääntöä C3. Jokainen virtapiiri pitäisi olla kommentoitu. Tämä sääntö on tarkoin määrätty IEC 61131 standardissa.



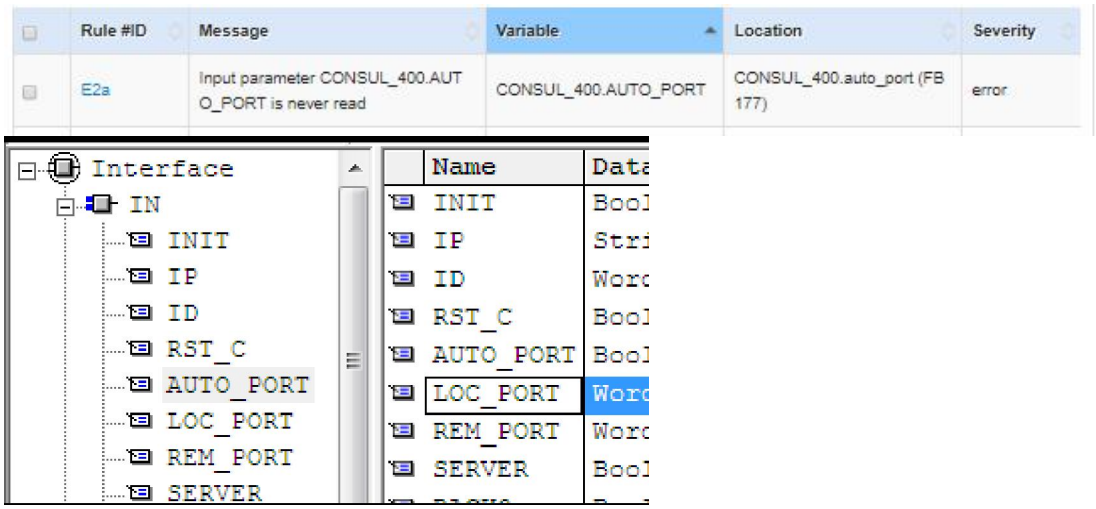
Kuva 5.12. Kommentointi virhe ja sijainti PLC ohjelmassa

Tässä kohdassa kuvassa 5.13 on rikottu sääntöä C1. Muuttujaa SRC15_R ei ole kommentoitu. Kaikki elementit pitäisi kommentoida, jotta ohjelman lukeminen, ymmärtäminen ja ohjelman ylläpitäminen olisi helpompaa

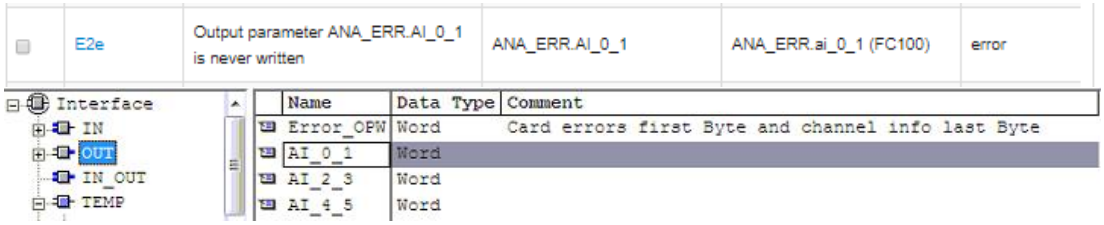


Kuva 5.13: Muuttujan kommentointi virhe ja sijainti PLC ohjelmassa

Tässä kohdassa kuvassa 5.14 on rikottu sääntöä E2a ja kuvassa 5.15 on rikottu sääntöä E2e. Parametreja pitäisi käyttää oikein ja tässä tapauksessa määriteltyä tulo parametria ei ole koskaan luettu ja määriteltyä lähtö parametria ei ole kirjoitettu ohjelmassa. Käyttämättömät parametrit ovat osoitus keskeneräisestä koodista tai koodista, joka sisältää tarpeettomia elementtejä.

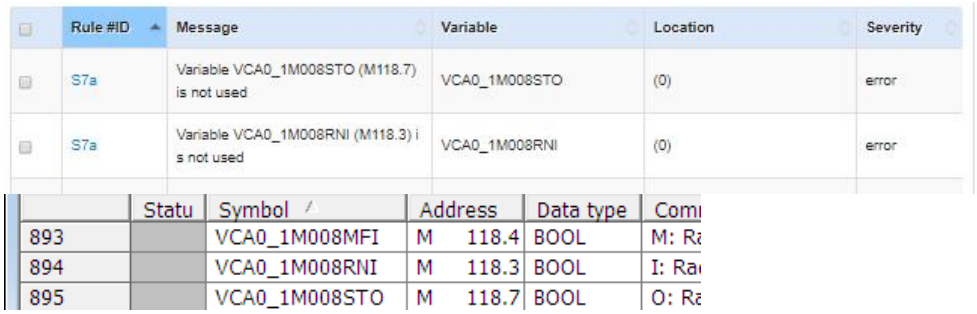


Kuva 5.14. Tulo parametri, jota ei ole luettu ohjelmassa.



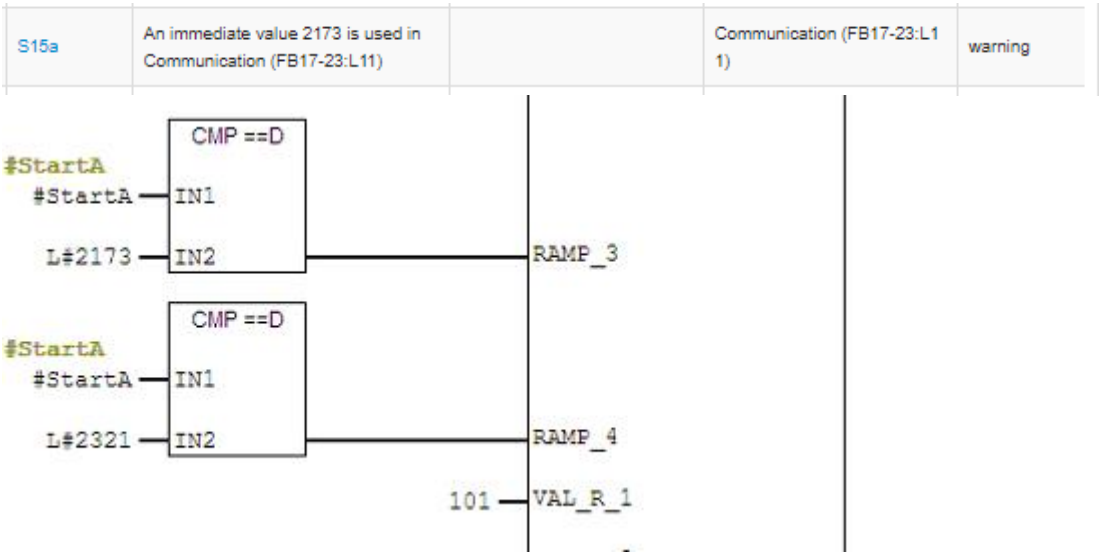
Kuva 5.15: Lähtö parametri, jota ei ole kirjoitettu ohjelmassa.

Tässä kohdassa kuvassa 5.16 on rikottu rakenteellista sääntöä S7. Määriteltyjä muuttujia pitäisi käyttää, lukuun ottamatta varamuuttujia. Määrittelemättömät muuttujat ja käyttämättömät tyypit ovat usein merkki unohtuneesta käsittelystä tai epätäydellisestä tietokannan puhdistamisesta.



Kuva 5.16. Rakenteellinen virheviesti ja käyttämättömän määritellyn muuttujan sijainti symboli taulukossa

Tässä kohdassa kuvassa 5.17 on rikottu sääntöä S15. Välittömiä arvoja tai vakioita ei tulisi käyttää koodissa. Käyttämällä hyvin nimettyjä vakioita, koodi on ymmärrettävämpi ja selkeämpi.



Kuva 5.17: Välittömiä arvoja on käytetty koodissa.

Informaatiokategoriassa tulostettavat viestit eivät ole aina virheitä vaan ne voivat olla hyödyllistä tietoa ohjelman rakenteesta. Informaatio I6 (kuva 5.18) kuvataan elementin kompleksisuudenrakennetta. Mitä suurempi luku on, sitä vaikeampi on vahvistaa rutiinin oikea käyttäytyminen. Kompleksisuuden rakenteen arvon ollessa suurempi kuin 1 niin vakavuusaste muuttuu infosta varoitukseksi.

Rule #ID	Message	Variable	Location	Severity
I6a	Element fc.RAD_GAIN has an essential Complexity ev(G)=1		RAD_GAIN (FC178:L3)	info
I6a	Element fc.SIGN_TO_UNSIGN has an essential Complexity ev(G)=1		SIGN_TO_UNSIGN (FC171:L6)	info

Kuva 5.18. Informaatio kategorian viestejä

Informaatio I9 laskee ohjelmahaarojen määrää tietyssä SFC:ssä. Tämän avulla saadaan tietoa SFC:n monimutkaisuudesta. Tähän kategoriaan voidaan luoda erilaisia laskureita, jotka antavat haluttuja tietoja ohjelmasta.

I8a	SFC fb.Engine Sequence.engine sequence_InitialStep has 18 steps.		Engine Sequence (FB30)	info

Kuva 5.19: SFC: n ohjelmahaaralaskuri.

6 POHDINTAA

Tämän opinnäytetyön pohjalta voin todeta, että PLC Checker on loistava työkalu PLC-ohjelmien määriteltyjen ohjelmointikäytäntöjen mukaisen laadunvarmistamiseen. Työkalun selkeiden nopeasti saatavien raporttien avulla virheiden havaitseminen ja niiden korjaaminen on helppoa. Raporttien avulla on helppo seurata ohjelman kehitystä koko sen kehityskaaren ajan.

PLC-ohjelma on ollut toiminnassa jo pitkään eikä sitä tämän analysoinnin jälkeen ole lähdetty kehittämään tai korjaamaan, joten PLC Checkerin konkreettista vaikutusta tämän ohjelman, luettavuuteen, luotettavuuteen, ylläpitävyyteen tai siirrettävyyteen ei voida tämän analysoinnin avulla todeta.

Voisin kuvitella, että ohjelmointi virheiden havaitseminen varhaisessa ohjelman kehitysvaiheessa, ennen ohjelman varsinaista testaamista ja käyttöönottoa nopeuttaa merkittävästi työnvalmistumista ja parantaa ohjelman luotettavuutta, luettavuutta ja siirrettävyyttä ohjelman ylläpidon ja mahdollisen myöhemmän ohjelmakehityksen aikana. Ja uskoisin näillä olevan merkittäviä taloudellisia hyötyjä.

Ohjelman valmiit säännökset ovat sellaisenaan jo hyviä hyvän ohjelmointitavan mittaamiseen. Mutta kaikki säännöt eivät välttämättä ole tarpeellisia kaikissa tapauksissa ja varmasti isommilla yrityksillä on myös omia käytäntöjä, jotka on mahdollista integroida PLC Checker-ohjelmaan. Selkeän säännöstön avulla on helppo ohjeistaa ohjelmoijia yrityksen ohjelmointitavoista ja yhtenäistää useamman eri ohjelmoijan ohjelmointikäytäntöjä.

Korvaamalla kalliin ja työlään manuaalisen laadun varmistamisen PLC Checker-ohjelmalla voidaan säästää merkittäviä summia koko ohjelmiston elinkaaren ajan.

PLC Checker olisi hyvä ottaa käyttöön heti jo ohjelman suunnitteluvaiheessa, jolloin yrityksen säännöstö olisi välittömästi käytössä ohjelmoinnin alettua. Vasta testien, käyttöönoton tai ylläpidon aikana käyttöönotettu PLC Checkkerin hyöty vähenee merkittävästi.

CS Control Software Oy tulee jatkossa myymään ja markkinoimaan tätä ohjelmaa ja palvelua ympäri maailmaa nykyisille ja uusille asiakkaille.

LÄHTEET

Laaksonen J. 2017. Toimitusjohtaja, CS Control Software Oy, Espoo. Henkilökohtainentiedonanto 29.10.2017.

Suomen sähköurakoitsijaliitto ry. 1991. Ohjelmoitava logiikka. Forssa: Forssan Kirjapaino Oy.

Asmala, H., Koskinen, K., Koskela, M., Mätäsniemi, T., Soini, A., Strömman, M., Tommila, T., Valkonen, J. 2005 Automaatiosovellusten ohjelmistokehitys. Suomen automaation tuki Oy.

SFS-käsikirja 631-2, Automaatio. Osa 2: Ohjelmointi ja dokumentointi, 2014, Helsinki: Suomen Standardisointiliitto SFS Ry.

IEC 61131-3: a standard programming resource. Verkkodukumentti PLCopen. www.plcopen .org/pages/tc1_standards/download/intro_iec.pdf. Luettu 21.10.2017

PLCopen Coding Guidelines Version 1.0. Verkkodukumentti PLCopen, 2016, http://www.plcopen.org/pages/pc2_training/. Luettu 21.10.2017

Presentation of PLC Checker Version 2.2, 2017a. Verkkodukumentti Itris Automation. https://myitris.automationsquare.com/products/plc-checker/. Luettu 21.10.2017

PLC Checker verifies and improves quality of PLC program. Verkkolehden artikkeli, Itris Automation, 2015, http://www.machinebuilding.net/ap/a1880.htm. Luettu 26.10.2017

PLC Checker now support PLCopen coding guidelines. Verkkolehden artikkeli, Itris Automation, 2017c. http://www.machinebuilding.net/p/p10461.htm. Luettu 26.10.2017

PLC programs development guidelines, 2015b. Itris Automation. https://myitris.automationsquare.com/products/plc-checker/. Luettu 20.6.2017

Itris Automation:n www-sivut. 2017. Luettu 20.9.2017. www.itris-automation.com

Export instruction for PLC Checker Step7, 2017b. Verkkodukumentti Itris Automation. https://myitris.automationsquare.com/products/plc-checker/. Luettu 24.10.2017

CS Control Software Oy: n www-sivut 2017. Luettu 26.10. www.controlsoftware.eu

LIITE 1

PLC CHECKER OLETUSARVOINEN SÄÄNNÖSTÖ

- S13 - SFC codes should contain one and only one initial step
  - S13a - SFC codes (except macro-steps) should not contain no initial step
  - S13b - SFC codes (except macro-steps) should not contain more than one initial step
  - S13c - Macro-steps should not contain initial step
- S15 - Immediate values should not be used in the code
  - S15a - Immediate values should not be used in the code
- S16 - The float or reals should not be compared using equality or difference operators
  - S16a - The float or reals should not be compared using equality or difference operators
- Information
  - I1 - Code should contain at least x% of commented instructions
    - I1a - Ladder code should contain at least x% of commented instructions (so at max 100-x% of instructions not commented). x=30.
    - I1a - Ladder code should contain at least x% of commented instructions (so at max 100-x% of instructions not commented). x=30.
    - I1b - Literal code should contain at least x% of commented instructions (so at max 100-x% of instructions not commented). x=30.
  - I2 - Code in comment
    - I2a - The comments must not contain code (PL7-Pro)
    - I2b - The comments must not contain code (Unity Pro)
    - I2c1 - The comments must not contain code (STEP7, STEP5, TIAPORTAL, PLCOPEN, CODESYS)
    - I2c2 - The comments must not contain code (STEP7, STEP5, TIAPORTAL)
    - I2c3 - The comments must not contain code (STEP7, STEP5, TIAPORTAL)
    - I2c4 - The comments must not contain code (STEP7, STEP5, TIAPORTAL)
    - I2c5 - The comments must not contain code (STEP7, STEP5, TIAPORTAL)
    - I2c6 - The comments must not contain code (STEP7, STEP5, TIAPORTAL)
    - I2c7 - The comments must not contain code (STEP7, STEP5, TIAPORTAL)
    - I2c8 - The comments must not contain code (STEP7, STEP5, TIAPORTAL)
    - I2c9 - The comments must not contain code (STEP7, STEP5, TIAPORTAL)
    - I2c10 - The comments must not contain code (STEP7, STEP5, TIAPORTAL)
    - I2c11 - The comments must not contain code (STEP7, STEP5, TIAPORTAL)
    - I2d - The comments must not contain code (RSLogix 5000)
  - I3 - The program should not contain dead code
    - I3a - The program must not contain dead code
    - I3b - The program must not contain never called code
    - I3c - No Always False Instructions (AFI) for Rockwell
  - I4 - List of locked elements
    - I4a - List of locked elements
    - I4b - List of locked elements
  - I5 - Cyclomatic number v(G)
    - I5a - Cyclomatic number v(G) (< =15)
    - I5b - Cyclomatic number v(G) (> 15)
  - I6 - Essential Complexity ev(G)
    - I6a - Essential Complexity ev(G) (= 1)
    - I6b - Essential Complexity ev(G) (> 1)
  - I7 - Source Line of Code count
    - I7a - Source Line of Code count
  - I8 - SFC steps count
    - I8a - SFC steps count
  - I9 - SFC branches count
    - I9a - SFC branches count
  - I10 - Copy/Paste detection ratio
    - I10a - Copy/Paste detection ratio
- Options
- Language
  - L1 - The Instruction List Language is not allowed.
    - L1a - The Instruction List Language is not allowed.