Tampereen ammattikorkeakoulu

TwinCAT 3:n versionhallinta

TortoiseSVN:llä

Arttu Söderblom

OPINNÄYTETYÖ
Maaliskuu 2020

Sähkö- ja automaatiotekniikka
Automaatiotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Sähkö- ja automaatiotekniikka
Automaatiotekniikka

SÖDERBLOM, ARTTU:
TwinCAT 3:n versionhallinta TortoiseSVN:llä

Opinnäytetyö 61 sivua, joista liitteitä 0 sivua
Maaliskuu 2020

Glaston Oy:lle automaation kehittäminen ja sen elinkaaren hallinta lasinkäsittelykoneissa on vaatimus toimiville koneille. Nykyaikana lasinkäsittelykoneissa on todella korkea automaatioaste. Tämä tarkoittaa sitä, että oikein toimiva PLC-ohjelma on elinehto asiakasyrityksille, jotka tuottavat käsiteltyä lasia. Jotta pystytään hallitsemaan jokaisen PLC-ohjelman elinkaarta, on erittäin tärkeää käyttää versionhallintaa. Tämän avulla yritys kykenee hallitsemaan PLC-ohjelmaa askel askeleelta projekteissaan.

Glastonin automaatio-osastolla on suunnitelmana vaihtaa TwinCAT 2 -ohjelmasta TwinCAT 3 -ohjelmaan. Tämä loi tarpeen kehittää versionhallintaa, sillä kyseisillä ohjelmilla on merkittäviä eroja siinä, kuinka ne luovat tiedostohakemistoja projektin alle. Syynä tälle on se, että TwinCAT 3 on täysin integroitu Visual Studio -alustaan. Tämän opinnäytetyön lähtökohtana on kehittää versionhallintaa vanhalla, jo käytössä olevalla TortoiseSVN-versionhallintaohjelmalla. Tavoitteena on tuoda versiotietoa TortoiseSVN:stä TwinCAT 3 -projekteihin.

Ongelma ratkaistiin kehittämällä kaksi skriptiä C#-ohjelmointikielellä. Ensimmäinen näistä ohjelmista on tarkoitettu suoritettavaksi automaattisesti, kun käyttäjä tekee tallennuksen TortoiseSVN:llä palvelimelle. Toisen ohjelman suoritus taas tapahtuu, kun käyttäjä suorittaa TwinCAT 3 -ohjelmalla PLC-ohjelman käännöksen. Molempien on tarkoitus tuoda version tietoja arkistosta TwinCAT 3:n vastaavaan projektiin.

Ohjelmien suurin hyöty tulee mahdollisuudesta automatisoida työvaiheita PLC-ohjelman kehityksen aikana. Tällöin on mahdollisuus työskennellä nopeammin ja välttää inhimillisiä virheitä. Glastonin automaatio-osastolla on tarkoituksena ottaa nämä ohjelmat yleiseen käyttöön samalla kun siirrytään käyttämään TwinCAT 3 -ohjelmaa.

Asiasanat: versionhallinta, TwinCAT 3, TortoiseSVN

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Electrical and Automation Engineering
Automation Engineering

SÖDERBLOM, ARTTU:
TwinCAT 3 Version Control with TortoiseSVN

Bachelor's thesis 61 pages, appendices 0 pages
March 2020

For Glaston Oy, development of automation and its lifecycle management is a prerequisite for the glass treatment machines they produce. Nowadays glass treatment machines have a high degree of automation. This means that properly working PLC programs are vital for companies which produce treated glass. In order to be able to control lifecycle of every PLC program in every product, it is important to use version control. In that way, the company can control PLC programs step by step in their projects.

The automation department in Glaston has a plan to change TwinCAT 2 to TwinCAT 3. This created the need to develop version control, because these programs have significant differences in their ways to generate file directories under the project. The reason for that is that TwinCAT 3 has been fully integrated into the Visual Studio platform. In this thesis the basis is to stay in the old version control system, TortoiseSVN, which is already in use. The objective is to bring version control information from TortoiseSVN to the projects in TwinCAT 3.

The solution to the problem was to develop two scripts with the C# programming language. The first program is meant to be executed automatically when the user commits to the TortoiseSVN server. The second program is executed when the user does any PLC program conversion in TwinCAT 3. Both have the general purpose of bringing version information from the repository to the corresponding working copies of TwinCAT 3 projects.

The greatest benefit from these programs is the capability to automate working stages in developing PLC programs. In that way, the user is able to work faster and to avoid human errors. The automation department at Glaston aspires to place these programs in general use while simultaneously transferring to using TwinCAT 3.

Key words: version control, TwinCAT 3, TortoiseSVN

SISÄLLYS

1 JOHDANTO 6
2 OPINNÄYTETYÖN TILAAJA JA TARVE 7
2.1 Tilaaja 7
2.2 Tuotteet 7
2.3 Suunnittelu tilaajalla 8
2.3.1 Automaatiosuunnittelu 8
2.3.2 Automaatio-osaston tarve versionhallinnan kehittämiselle ... 9
3 VERSIONHALLINTA JA VERSIONHALLINTAJÄRJESTELMÄT 10
3.1 Versionhallinta yleisesti 10
3.1.1 Tiedonsäilytys 11
3.1.2 Työkopio 12
3.2 Versionhallintajärjestelmät 12
3.2.1 Keskitetty versionhallintajärjestelmä 12
3.2.2 Hajautettu versionhallintajärjestelmä 14
3.3 TortoiseSVN 15
3.3.1 Järjestelmä-arkkitehtuuri 15
3.3.2 Versionhallinta Subversionilla 22
4 BECKHOFF TWINCAT 30
4.1 Beckhoff yleisesti 30
4.2 TwinCAT 30
4.3 TwinCAT 2:n erot TwinCAT 3:en 31
4.4 TwinCAT 3 projekti 32
5 Suoritettavat ohjelmat 36
5.1 Suoritettavien ohjelmien tarkoitus 36
5.2 Skriptien käyttöönottaminen ja käyttäminen 36
5.2.1 Skriptien käyttöönotto 36
5.2.2 Skriptien käyttäminen 43
5.3 Versionhallinnan skripti TortoiseSVN:ään 53
5.4 PLC käännöksen skripti 57
6 POHDINTA 59
LÄHTEET 61

ERITYISSANASTO tai LYHENTEET JA TERMIT (valitse jompikumpi)

ADS	Automation Device Specification
AnkhSVN	Subversionin integrointi ohjelma Visual Studioon.
Branch	Versionhallinnan kehityspolku
C#	C sharp
Commit	Versionhallinnan tallennus
Fork	Kehityspolun haaroitus
Hook	Kytkin ulkopuolisten ohjelmien suorittamiseen
PLC	Programmable logic controller
Repository	Versionhallinnan tietovarasto tai säiliö
Skripti	Suoritettava ohjelma
SVN	Subversion. Keskitetty versionhallintajärjestelmä
TwinCAT	Total Windows Control and Automation Technology
XAE	eXtended Automation Engineering
VCS	Version Control System. Versionhallintajärjestelmä
VisualSVN	Subversionin integrointi ohjelma Visual Studioon
Whitelist	Lista sallituista asioista
Working directory	Työhakemisto

1 JOHDANTO

Automaatio-ohjelmiston kehittämiseen käytettävä aika ei ole sen koko elinikään nähden välttämättä kovinkaan suuri, vaan usein koneita varten kehitettävät automaatio-ohjelmat palvelevat käyttötarkoituksissaan jopa kymmeniä vuosia asiakkaalla. Tähän elinkaareen sisältyy ylläpitoa, päivityksiä sekä vikatilanteiden ratkaisua, jolloin on ensiarvoisen tärkeää olla mahdollisuus saada nopeasti tieto siitä, mikä ohjelmiston versio asiakkaan koneella toimii. Esimerkiksi opinnäytetyön tilaaja Glaston toimittaa lasinkäsittely koneita, joilla tuotannon keskeytykset ovat erittäin haitallisia asiakkaan toiminnan kannalta ja on Glastonin edunmukaista saada mahdolliset ongelmatilanteet ratkaistua nopeasti.

Ohjelmien kehitys ei useissa tapauksissa ole yhden henkilön työ, vaan tähän kehitykseen osallistuvat useat henkilöt. Tällöin kehitettävän ohjelman täytyy olla kaikkien kehittäjien saatavilla ja siitä pitää kyetä seuraamaan, mitä muutoksia ja milloin toiset ovat kehityksen aikana tehneet ohjelmistoon. Tätä tarvetta varten on luotu erilaisia versionhallintajärjestelmiä, jotka mahdollistavat useamman toimijan yhteisen kehittämistyön.

Usein ei ole tuotteen toimittajan edunmukaista antaa asiakkaallensa tuotteen lähdekoodia. Tällöin halutaan pitää tämä itsellään ja antaa asiakkaalle toimitettavan koneen yhteydessä ainoastaan toimintaan vaadittavat osuudet. Tällöin kuitenkin täytyy olla jokin keino, millä selvittää asiakkaalla olevan ohjelmiston versio.

TwinCAT 3:n tapauksessa asiakkaan koneella olevien tarpeellisten tiedostojen joukkoon halutaan myös Glastonin oman versionhallinnan mukaiset versiotiedot. TortoiseSVN tai TwinCAT 3 -ohjelmat eivät tuo näitä tietoja, joten se olisi käytännössä käyttäjän tehtävä lisätä nämä tiedot.

Opinnäytetyön tarkoitus on ratkaista tämä ongelma siten, että automaatio-ohjelmiston kehittäjä onnistuu mahdollisimman pienellä vaivannäöllä seuraamaan itse sekä saattamaan asiakaskoneelle tiedon siitä, mikä ohjelmoitavan logiikan ohjelma on työnalla. Tämän ongelman ratkaisuksi kehitettiin kaksi skriptiä, jotka toimivat sekä TwinCAT 3:n että TortoiseSVN versionhallintajärjestelmän kanssa.

2 OPINNÄYTETYÖN TILAAJA JA TARVE

2.1 Tilaaja

Glaston-konserni on Suomessa perustettu yritys, joka on kehittää, valmistaa sekä myy lasinkäsittelyyn tarkoitettuja koneita ja palveluita. Kyseisillä koneilla käsitellään rakennus-, ajoneuvo-, aurinkoenergia- sekä kaluste- ja laiteteollisuuden käyttöön tulevia laseja. Esimerkkikohteena, jonka lasienkäsittelyyn on käytetty Glastonin koneita, on maailman korkein rakennus, 828 metriä korkea Arabiemiirikunnissa sijaitseva Burj Khalifa. Glaston tarjoaa siis laajan ja kehittyneen lasinjalostuskonevalikoiman. Sitä täydennettiin vuonna 2019 tehdyllä yritysostolla, jossa Bystronic glass yhdistyi Glastoniin. Itse Glastonilla on pääkonttori Helsingissä sekä tuotantolaitoksia Tampereella sekä Kiinan Tianjinissa (glaston.net 2019).

2.2 Tuotteet

Merkittävin tuoteryhmä Glastonilla on tasolasinkarkaisukoneet, mutta valikoimaa löytyy myös laminointiin, taivutukseen ja ajoneuvolaseihin liittyen. Glastonin suosituimpiin tuotteisiin kuuluu FC-sarjan karkaisulinja, joka on kuvassa 1 (glaston.net 2019).

KUVA 1. FC-sarjan tasolasinkarkaisulinja (glaston.net 2019)

Kuvan 1 linjasto koostuu tasolasinkarkaisu-uunista ja jälkijäähdyttimestä sekä kuljettimista. Tähän on asiakkaan tarpeiden mukaan myös mahdollista yhdistää erilaisia mittauslaitteita sekä muista tuotesarjoista taivuttimia ja laminointilaitteita.

2.3 Suunnittelu tilaajalla

Glastonin kaikki suunnittelu tapahtuu Tampereella. Suunnittelu voidaan jakaa karkeasti mekaniikka-, digitalisaatio-, sähkö- ja automaatiosuunnitteluun. Kyseisistä osastoista löytyy tämän lisäksi vielä alaosastoja, joihin palataan myöhemmin automaatiosuunnittelun osalta. Koneiden pääasiallinen tuotanto ja tuotekehitys tehdään Tampereella. Tämä helpottaa suunnittelua, koska on helppo sopia tapaaminen tai käydä kysymässä ja tarkastamassa mahdollisesti toiselta osastolta omaan työhön liittyviä asioita.

2.3.1 Automaatiosuunnittelu

Automaatiosuunnittelu on Glastonilla jaettu kahteen eri ryhmään, jotka toimivat tiiviisti toistensa kanssa yhteistyössä. Koneiden ohjauksesta vastaa PLC-osasto ja käyttöliittymien suunnittelusta ohjelmisto-osasto.

PLC-osasto voidaan jakaa vielä kahteen osaan. Toinen käyttää suunnittelussaan tällä hetkellä Beckhoff TwinCAT 2 ja toinen Siemens TIA portalia. Käytännössä TwinCAT:in avulla toteutetaan päätuoteryhmän eli tasolasinkarkaisulinjastojen ohjaaminen. Siemensiä käytetään taas esimerkiksi laminointi ja muottiuunien suunnitteluun. Glastonin kaikki logiikoiden ohjelmointi tapahtuu lähes poikkeuksetta jäsennellyllä tekstillä. Koneissa on korkea automaatioaste eli käytännössä kaikki ohjaus tapahtuu automaation avulla. Konetta käyttävät erityisen koulutuksen saaneet operaattorit asiakkaalla. Koneissa käytettävä lähdekoodi on erittäin laaja ja koneissa voi olla jopa 10 000 sisään- ja ulostuloja. Eli itse automaatioon käytettävä ohjelma on todella suuri.

Toinen automaatiosuunnitteluun liittyvä osasto on ohjelmistosuunnittelu. Tämä on keskittynyt toteuttamaan koneiden käyttöliittymät. Näiden suunnittelussa ja toteutuksessa käytetään C#-ohjelmointikieltä. Koneissa käytettävät käyttöliittymät päätuotesarjoissa tehdään alusta loppuun asti tällä, jolloin käyttöliittymä saadaan räätälöityä täysin asiakkaan tarpeiden mukaan, korostaen asiakkaan toivomia ominaisuuksia käytön kannalta.

2.3.2 Automaatio-osaston tarve versionhallinnan kehittämiselle

Automaatio-osasto on päätuotesarjoissaan päättänyt siirtyä käyttämään TwinCAT 3:a vanhan TwinCAT 2:n sijasta. TwinCAT 3 tuo lukuisia etuja mukanansa, joita käsitellään tarkemmin opinnäytetyön aikana. Tämä kuitenkin synnytti tarpeen versionhallinnan kehittämiselle, sillä uuden ohjelman myötä kehitys siirtyy myös automaatio-osaston käyttämään TortoiseSVN versionhallintaohjelmaan aidosti. Osaston tarpeena on saada TortoiseSVN:stä saatava versionumero tuotua TwinCAT 3:n projektin tietoihin, jotta PLC-ohjelman kehitystä olisi helpompi seurata. TwinCAT 3 ei taas luo tätä versionumeroa missään vaiheessa, jolloin se vaatii skriptin eli automaattisesti suoritettavan ohjelman, joka toimii käyttäjän PC:n, TwinCAT 3:n sekä TortoiseSVN:n välillä. Tätä haastetta kuitenkin käsitellään tarkemmin opinnäytetyön edetessä.

3 VERSIONHALLINTA JA VERSIONHALLINTAJÄRJESTELMÄT

3.1 Versionhallinta yleisesti

Versionhallinnalla tarkoitetaan järjestelmää, jonka avulla voidaan hallita tiedostoja tallentaen niiden historia myös järjestelmään, monissa tapauksissa palvelimelle. Tämän avulla on mahdollista tarkastella tiedostoja myöhemmin halutun version kohdalta. Vanhoista tiedostoista löytyy siis käytännössä kopiot aina siltä hetkeltä, kun kyseiset tiedostot ovat siirretty versionhallintaan (Ernst, M. 2012). Vanhojen versioiden tarkastusmahdollisuus on suuri etu, kun mietitään esimerkiksi ohjelmistokehittäjää, joka muokkaa vanhaa ohjelmaa ja se ei toimisikaan muokkauksen jälkeen. Tällöin ohjelman palauttaminen jälleen toimivaksi voisi olla suuren työn takana. Versionhallintajärjestelmän kanssa on taas mahdollista palata aikaisempaan toimivaan versioon helposti, tai sellaiseen kohtaan, missä epäilee virheen tapahtuneen. Tällöin säästyy aikaa huomattavasti.

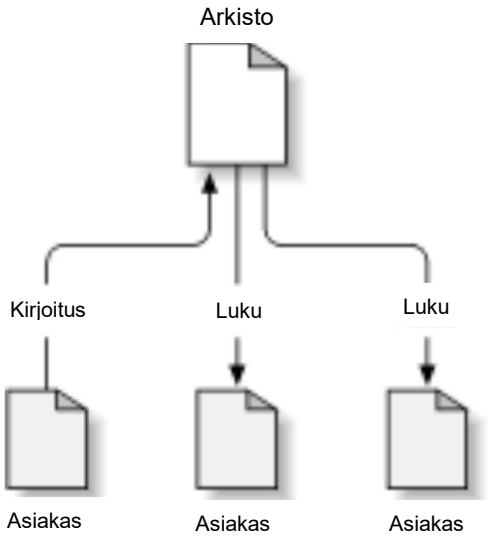
Vanhojen tietojen hallinnan lisäksi tärkein syy versiohallintajärjestelmien olemassa ololle ja käytölle on se, että se mahdollistaa useiden henkilöiden samanaikaisen työskentelyn kyseisen tiedoston tai tiedostojen parissa. Käyttäjä voi päättää, minkä version kyseisen tiedoston historiasta ottaa muokattavaksensa, sekä sen, milloin hän julkaisee sen muille. Versionhallintajärjestelmien avulla on mahdollista lukita tiettyjä tiedostoja muokkauksensa ajaksi, jotta voidaan välttyä konflikteilta, mikäli toinen käyttäjä muokkaisi yhtäaikaisesti samaa kohtaa vahingossa. Mikäli lukitusta ei ole tehty ja edellä mainittu konflikti sattuisi, versionhallinta järjestelmät usein pyytävät käyttäjän päättämään, miten muutoksien kanssa toimitaan. Tämä ei siis johtaisi tiedoston vioittumiseen. Versionhallintajärjestelmän avulla on myös mahdollista tarkkailla tiedostoa tai tiedostoja koskevia muutoksia, esimerkiksi kuka muutoksen on tehnyt, milloin muutos on tehty ja mitä on muutettu (Ernst, M. 2012).

Edellä mainittujen lisäksi versionhallintajärjestelmät mahdollistavat työskentelyn useammalta koneelta, sillä kaikki versiot löytyvät pääasiassa käytettävältä palvelimelta. Se on hyödyllistä myös yksin projekteja tehdessä (Ernst, M. 2012).

Versionhallintajärjestelmät voidaan jakaa kahteen osaan. Keskitettyihin versionhallintajärjestelmiin (Centralized Version Control Systems) ja hajautettuihin versionhallintajärjestelmiin (Distributed Version Control Systems) (Ernst, M. 2012). Opinnäytetyössä olennaisempi on keskitetty, sillä Glastonilla on käytössään TortoiseSVN, joka kuuluu edellä mainittuun keskitettyihin järjestelmiin. Tästä huolimatta käsitellään myös hajautettua versionhallintajärjestelmää, jotta voidaan vertailla näiden kahden eroja. Näiden erojen avulla pyritään selvittämään, olisiko mahdollisesti hajautettu versionhallintajärjestelmä parempi Glaston tarpeisiin.

3.1.1 Tiedonsäilytys

Yksi versionhallinnan ydinasioita on se, mihin tietoa tallennetaan. Käsitellään tässä tapauksessa, kuinka keskitetyssä järjestelmässä tiedontallennus tapahtuu. Versionhallintajärjestelmässä on käytössä tätä varten keskitetty järjestelmän tiedontallennus (Repository). Useimmiten tieto tallennetaan tänne puuta muistuttavaan tiedostohierarkiaan (Filesystem tree) tiedostojen polkujen perusteella. Muutkin käyttäjät voivat lukea ja muokata täällä olevia tiedostoja halutessaan. Kuvassa 2 esitetään, kuinka tätä tiedonsäilytystä voidaan käyttää asiakas/palvelin (client/server) tyyppisessä järjestelmässä (Collins-Sussman, Filzpatrick & Pilato 2011, 1)



KUVA 2. Asiakas/palvelin järjestelmä (Collins-Sussman, Filzpatrick & Pilato 2011 1)

Kuten kuvasta 2 havaitaan, järjestelmä muistuttaa itsessään hyvin paljon tavallista palvelimen toimintaa. Erona kuitenkin tavalliseen palvelimeen on se, että tässä tapauksessa versionhallintajärjestelmän tallennuspaikka muistaa kaikki aikaisemmatkin versiot kyseisestä tiedostosta. Tällöin käyttäjällä on mahdollisuus päästä käsiksi. Tämän lisäksi on myös mahdollisuus tarkastella kuka ja milloin on tehty muutoksia tiedostoon tai jopa, kuka on tehnyt muutoksen tiettyyn kohtaan tiedostoa (Collins-Sussman, Filzpatrick & Pilato 2011, 1).

3.1.2 Työkopio

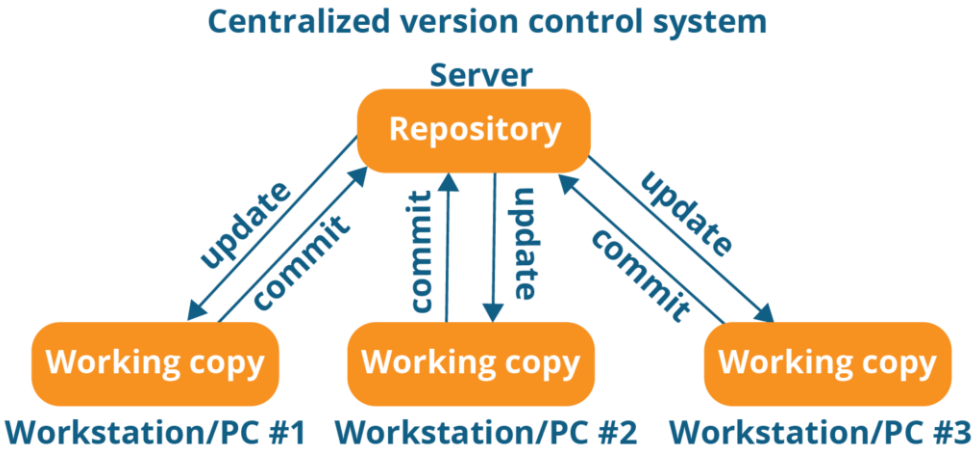
Tiedostoa on mahdollista muokata tai lukea vain työkopiona (Working Copy). Tämä tarkoittaa sitä, että haluttu versio kyseisestä tiedostosta siirtyy suoraan käyttäjän henkilökohtaiselle koneelle palvelimelta, jolloin hän voi tarkastella sitä tai muokata ilman, että muiden käyttäjien tekemät muutokset vaikuttavat siihen. Kyseiset tiedostot voivat olla mitä tahansa tyyppiä, sillä versionhallintajärjestelmässä ei tehdä muutoksia tiedostojen sisältöön, vaan ainoastaan tallennetaan ne. Versionhallintajärjestelmä tunnistaa kuitenkin, jos tiedostoon tehdään muutoksia. Tämän avulla se osaa myös siirtää muokatun työkopion versionhallintaan muiden tiedostojen joukkoon palvelimelle, kun käyttäjä haluaa näin. Tällöin ei siirretä turhaan ajan tasalla olevia tiedostoja, joka isojen tiedostomäärien kanssa vaatisi aikaa ja voisi näin ollen vähentää versionhallinnan käyttöä (Collins-Sussman, Filzpatrick & Pilato 2011, 2).

3.2 Versionhallintajärjestelmät

3.2.1 Keskitetty versionhallintajärjestelmä

Järjestelmä toimii tyypillisesti asiakas/palvelin periaatteella. Perusajatuksena on, että järjestelmällä on yksi keskus, johon kopioidaan tehdyt muutokset. Tallennuspaikkana toimii käytännössä lähes poikkeuksetta palvelin. Käyttäjä siirtää muu-

toksensa palvelimelle, jonka jälkeen nämä muutokset ovat kaikkien muiden nähtävissä järjestelmässä, kun he päivittävät siellä olevan tietokannan ajan-tasalle. Kuvassa 3 on nähtävillä järjestelmäarkkitehtuuri. Käytetyimpiä tämän tyyppisiä versionhallintajärjestelmiä ovat Subversion ja Perforce (medium.com 2020)



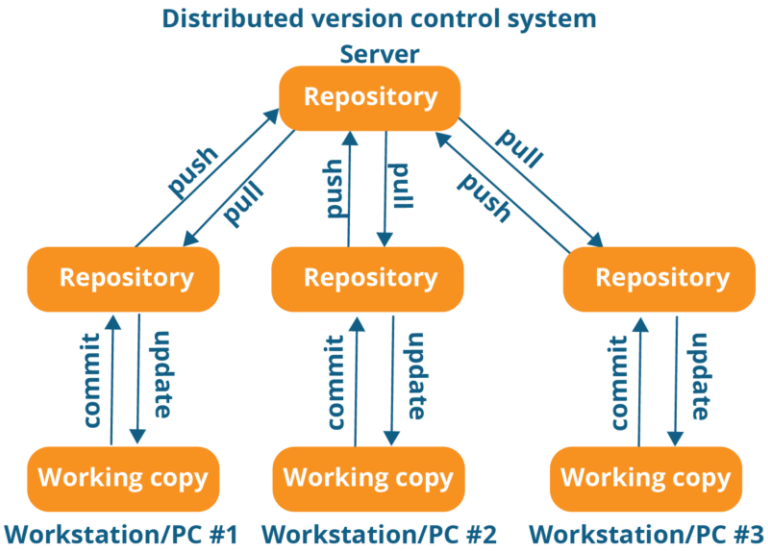
KUVA 3. Keskitetty versionhallintajärjestelmä (medium.com 2020)

Keskitetyssä järjestelmässä etuna on laajempi valikoima eri käyttöliittymäsovelluksia. Myös oppaita ja dokumentaatiota löytyy suhteellisen hyvin, ainakin perinteisiin käyttötarkoituksiin. Keskitetty järjestelmä on myös helppo ymmärtää toimintaperiaatteeltaan, jolloin sen kanssa toimiminen on suhteellisen helppoa. Näiden lisäksi tärkeänä etuna voidaan pitää sen hyvää toimintaa binääristen tiedostojen kanssa työskennellessä (medium.com 2020).

Keskitetyn järjestelmän heikkous on sama kuin sen vahvuus, eli yksinkertaisuus. Koska tallennus toteutetaan vain yhteen paikkaan, niin palvelimen vikaantuessa kaikki kehitystyö keskeytyy ja pahimmassa tapauksessa kaikki tiedot katoavat. Järjestelmä vaatii myös verkkoyhteyden, tiedon tallentamiseksi versionhallintajärjestelmään. Tukea sekä päivityksiä voi olla suhteellisen harvoin tarjolla, sillä kyseessä on kuitenkin suhteellisen vanha versionhallintatekniikka (bitbucket.org 2020).

3.2.2 Hajautettu versionhallintajärjestelmä

Hajautetussa järjestelmässä kaikilla käyttäjillä on kopio palvelimen sisällöstä myös oman koneensa kovalevyllä. Nykyään tallennustila on niin halpaa ja tiedostot pakataan tehokkaasti, että se ei ole ongelma. Kun koko palvelimen sisältö on käyttäjän koneella, on mahdollisuus tehdä muutoksia tiedostoihin ilman verkkoyhteyttä. Verkkoyhteyden kautta nämä versiot on mahdollista siirtää palvelimelle, jossa ne ovat kaikkien saatavilla. Kuvassa 4 on hajautetun versionhallintajärjestelmän toiminta (medium.com 2020).



KUVA 4. Hajautetun versionhallintajärjestelmän arkkitehtuuri (medium.com 2020)

Suosituimpia hajautettuja versionhallintajärjestelmiä ovat Git ja Mercurial. Näiden etuna on hajautetun arkkitehtuurin ansiosta hyvä vikasieto. Mikäli palvelimella tulisi ongelma, löytyvät versiot myös käyttäjän koneelta. Tällöin myös varmuuskopioita kyseisistä tiedostoista olisi kaikilla, jotka käyttävät kyseessä olevaa versionhallintajärjestelmää. Järjestelmä on myös optimoitu siten, että tiedostojen siirto on nopeata palvelimelle (bitbucket.org 2020).

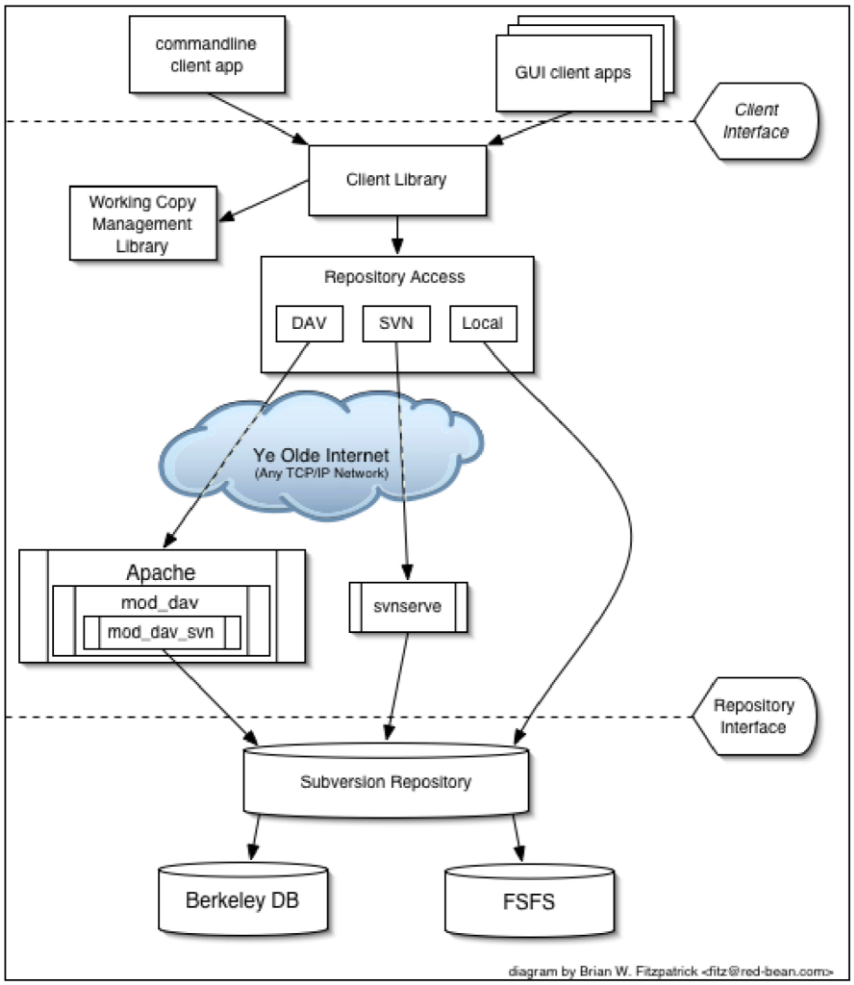
Huonoina puolina voidaan pitää sen heikkoa käytettävyyttä binääristen tiedostojen kanssa. Myös jos tiedostojen historia ja säilö ovat erittäin suuria, järjestelmän vuorovaikutus voi hidastua. Näiden lisäksi hajautetun versionhallintajärjestelmän käyttö on hankalampaa, johtuen epäintuitiivisistä komennoista ja hankalamasta järjestelmäarkkitehtuurista (bitbucket.org 2020).

3.3 TortoiseSVN

Apachen Software Foundationin kehittämä Subversion on avoimen lähdekoodin versionhallintajärjestelmä. Kyseessä on keskitetty versionhallintajärjestelmä, jonka tavoitteena on ollut saada luotua luotettava ja yleismaailmallisesti soveltuva järjestelmä. Se korvaa vanhan CVS:n (Concurrent Versions System), joka on tarkoitettu lähdekoodien versionhallintaan. CVS:ä jokaiselle tiedostolle tuli oma versionumero, mutta SVN:ssä koko versiopuulle päivittyy kyseinen versionumero (subversion.apache.org 2020).

3.3.1 Järjestelmä-arkkitehtuuri

Subversion on yksi järjestelmä, joka koostuu useista eri osista. Käytännössä järjestelmä voidaan jakaa kuvan 5 mukaisesti asiakas- ja tietovarastorajapintoihin.



KUVA 5. Subversionin arkkitehtuuri (Brian W. Fitzpatrick 2020)

Kuva 5 mukaisessa järjestelmäarkkitehtuurissa asiakasrajapinnan ulkopuolella toimivat ne osat järjestelmästä, jotka ovat käyttäjän käytössä. Nämä voidaan jakaa kahteen osaan. Järjestelmää voidaan käyttää Windowsin komentorivin tai graafisen käyttöliittymän avulla. Graafinen käyttöliittymä on hyvin intuitiivinen ja se on integroitu tietokoneen resurssienhallintaan. Sen avulla voi tehdä käytännössä kaiken mitä tarvitsee versionhallintaa käyttääkseen, esimerkiksi tiedostojen siirron tiedostovarastoon, päivityksen tai tarkastella tallennettujen tiedostojen rakennetta. Komentorivi soveltuu taas paremmin esimerkiksi tapauksiin, joissa halutaan hallita ulkopuolisilla ohjelmilla tai skripteillä versionhallinnan ominaisuuksia, kuten tämän opinnäytetyön tapauksessa. Tätä kautta toimivat myös erilaiset ohjelmistojen integraatiot, esimerkiksi AnkhSVN ja VisualSVN, jotka on integroi Subversionin suoraan Visual Studioon (Collins-Sussman, Filzpatrick & Pilato 2011, 232).

Asiakasrajapinnasta eteenpäin siirryttäessä on seuraavaksi asiakaspuolenkirjastot. Niiden tarkoitus on hallita työkopioita, jossa hakemistot sisältävät tiedostoja ja alihakemistoja. Ne toimivat paikallisena palvelimen arkiston esityksenä, jota voidaan muokata ja tiedostot siirretään tätä kautta seuraavalle tiedostovaraston tasolle (Collins-Sussman, Filzpatrick & Pilato 2011, 232).

Asiakaskirjaston yhteydessä toimiin työkopiokirjasto, jossa hallitaan taas käytännössä kaikkia tietoa työkopioista. Työkopiointikirjasto sisältää kaiken työkopioiden hallintaan vaadittavan tiedon, jota varten on alihakemisto .svn. Alihakemistossa sijaitsee jokaisen työkopion ja useiden muiden tiedostojen ja hakemistojen tilatiedot käyttäjää varten (Collins-Sussman, Filzpatrick & Pilato 2011, 232).

Laajinta vastuuta järjestelmässä hoitaa asiakaskirjasto, joka suorittaa toiminnot tietovarastoon pääsytasoon. Asiakaskirjasto tarjoaa myös ylemmän-tason ohjelmointirajapinnan sovelluksille, jotka suorittavat yleisiä versionhallintaan liittyviä toimintoja (Collins-Sussman, Filzpatrick & Pilato 2011, 232).

Käyttäjän toimintaan liittyvä alue päättyy tietovaraston pääsytasoon. Sen tarkoitus on valvoa tietoa asiakaskirjastojen ja tietovarastojen välillä. Tämä sisältää liitännäisyyksiä eri protokollille moduuleiden avulla. Subversion-palvelinmoduuli on luotu sitä varten, että mikäli valmiiksi tarjolla olevat protokollat eivät riitä käyttäjän

tarpeisiin, on mahdollista kehittää omia kirjastoja näitä varten. Nämä kirjastot voivat sisältää uusia protokollia tai näihin voidaan jopa kehittää omia protokollia tarpeidensa mukaan (Collins-Sussman, Filzpatrick & Pilato 2011, 236).

Viimeisenä tietovarannon käyttöliittymässä on taso, joka säilyttää kaikkea versionhallinnassa olevaa tietoa. Tätä kutsutaan Subversion tietovarastoksi. Periaatteessa tämä on vain hakemisto, jonka alikirjastoissa on konfigurointiin liittyviä tiedostoja. Osan näistä tiedostojen sisällöistä ihminen ymmärtää ja osan ei. Osiossa tulee myös määritellä, minkä tyyppistä tiedontallennusta käytetään (Collins-Sussman, Filzpatrick & Pilato 2011, 140).

Vaihtoehtoja tiedontallennukselle on kaksi, Berkley DB tietokanta tai FSFS tiedostojärjestelmä. Kumpikaan näistä sen virallisempi vaihtoehto kuin toinen, vaikka Subversion 1.2 versiosta eteenpäin oletuksena on käytössä FSFS. Molemmat ovat riittävän luotettavia versionhallintaan, mutta FSFS on joustavampi eri käyttötilanteissa. Joustavuus vähentää riskiä käyttää sitä väärin, jolloin se voidaan laskea sen eduksi. Berkley DB taas sisältää enemmän osia, jolloin se on myös tämän suhteen haastavampi. Tämän seurauksena nykyään on yleisempää käyttää Berkely DB:tä. Valintaan ei tarvitse kuitenkaan sitoutua, sillä Subversion tukee muutoksia tiedostovaraston muutoksissa. Näistä on esitetty vertaileva kuvaus svn-bookissa, joka näkyy taulukossa 1 (Collins-Sussman, Filzpatrick & Pilato 2011, 144).

TAULUKKO 1. Tiedontallennus menetelmät (Collins-Sussman, Filzpatrick & Pilato 2011, 144)

	Ominaisuus	**Berkeley DB**	**FSFS**
Toimintavarmuus	Tiedon eheys	Normaalisti käytettäessä äärimmäisen luotettava	Toimintavirheet tuhoavat sisältöä
	Alttius häiriöille	Herkkä virheille. Kaatuessaan ja käyttöoikeus ongelmien yhteydessä tietokanta voi estyä. Voi vaatia lokikirjojen palautuksen	Melko tunnoton
Helppokäyttöisyys	Lukutila	Ei	Kyllä
	Itsenäinen varasto	Ei	Kyllä
	Verkko-järjestelmä	Yleisesti ei	Kyllä
	Lupien hallinta	Altis käyttäjän oletusoikeuksien muutoksissa tapahtuville virheille, suositellaan ainoastaan yhdelle käyttäjällä pääsyä näihin	Toimii oletusoikeuksien muutoksissa laajasti
Skaalattavuus	Levyn käyttö	Käyttää paljon, varsinkin jos lokitietoja ei poisteta	Käyttää vähemmän.
	Tiedosto-varaston laajuus	Tietokanta, jolloin voidaan luoda ongelmitta useita	Ei sovi laajoille alkiokannoille
	Paljon tiedostoja	Hitaampi	Nopeampi
Suorituskyky	Version tarkastus	Ei merkittävää eroa	Ei merkittävää eroa
	Laajat tallennukset	Hitaampi. Pidempään käytettynä kuitenkin parempi tallennus	Nopeampi. Mutta viimeistelyviiveet voivat aiheuttaa asiakkaalle aikakatkaisuja

Berkeley DB

Subversionia kehittäessä Berkeley DB valittiin useiden syiden perusteella tietokanta vaihtoehdoksi. Näihin lukeutuivat avoimen lähdekoodin lisenssit, tiedonsiirron tukeminen, luotettavuus, suorituskyky, ohjelmointirajapinnan yksinkertaisuus, vuorovaikutuksen suojaus sekä tuki osoittimille (Collins-Sussman, Filzpatrick & Pilato 2011, 145).

Yksi Berkeley DB:n suurimmista eduista on sen tiedonsiirron tuki. Useat prosessit voivat päästä tietovarastoihin, jolloin voisi olla pelkona toisen tiedoston tahaton korvaaminen. Berkeley DB:ssä tiedonsiirtojärjestelmä on eristettynä, jolloin toimintoja suorittaessa Subversionin tiedostovarasto näkee suoraan staattisesta tietokannasta, jota vain Berkeley käsittelee. Tällä vältetään muiden prosessien aiheuttamat häiriöt tietokannassa, jolloin ei myöskään tule päällekkäisiä toimintoja. Myös mikäli aiheutuisi jokin konflikti, koko toiminto peruuntuisi siten, että oltaisiin taas alkuperäisessä tilanteessa (Collins-Sussman, Filzpatrick & Pilato 2011, 145).

Toinen hyödyllinen ominaisuus Berkley DB:ssä on kyky ottaa varmuuskopioita tietokannasta ilman tarvetta kytkeä sitä pois käytöstä. Tämä tarkoittaa, että siitä voidaan tarvittaessa ottaa täysin toimivia varmuuskopioita ilman keskeytystä (Collins-Sussman, Filzpatrick & Pilato 2011, 145).

Oikein käytettäessä Berkeley DB on erittäin luotettava tietokanta. Subversion käyttää Berkeley DB:n kirjausympäristöä, joka tarkoittaa sitä, että tietokanta kirjoittaa ensin levyllä tapahtuneista muutoksista lokitiedostoja kuvauksen kanssa ja mitä muutoksia se aikoo tehdä. Tämän jälkeen se suorittaa tehtävät muutokset itse. Tällä varmistetaan se, että jos jokin menee vikaan, tietokanta voi palata takaisin viimeiseen tarkastuspisteeseen, jotta voidaan olla varmoja, etteivät lokitiedostot ole vikaantuneet. Tämän jälkeen se voi yrittää tiedonsiirtoa uudestaan, kunnes saadaan varmuus siitä, että tieto on saatu tallennettua tarvittavassa muodossa (Collins-Sussman, Filzpatrick & Pilato 2011, 145).

Berkeley DB:n heikkouksina voidaan pitää sen tiettyjä rajoituksia. Berkeley DB:tä ei voi siirtää toiselle alustalle. Esimerkiksi ei ole mahdollista siirtää Unix-järjestelmällä luotua tietovarastoa Windows-järjestelmälle. Vaikka itsessään Berkeley

DB:n tietokanta on arkkitehtuuriltaan itsenäinen, muu osa järjestelmästä ei ole. Tällöin se on edellä esimerkeinä mainitusta käyttöjärjestelmistä riippuvainen. Se ei myöskään toimi Windows 95/98 -järjestelmillä, mutta tämä ei ole kovinkaan todennäköinen ongelman aiheuttaja enää nykypäivinä, koska se toimii näiden jälkeen tulleilla Windows-käyttöjärjestelmillä (Collins-Sussman, Filzpatrick & Pilato 2011, 145).

Berkely DB tukee verkossa jakoa, mutta vaatii tiettyjä erityisiä ominaisuuksia. Tässä on ongelma taas se, ettei suurin osa verkossa toimivista tiedostojärjestelmistä tue näitä vaadittuja ominaisuuksia. Berkeley DB:n tietovarasto ei myöskään salli verkossa jakoa useille asiakkaalle yhtä aikaa, mikä on kuitenkin usein juuri se asia, mitä verkossa jakamiselta haluttaisiin. Mikäli tämän tyyppisiä tapauksia yritettäisiin toteuttaa Berkeley DB:n avulla, voisi olla välittömästi havaittavissa selittämättömiä vikatiloja tai vasta myöhemmin voitaisiin huomata tietokannan vikaantuminen (Collins-Sussman, Filzpatrick & Pilato 2011, 145).

Koska Berkeley DB on suoraan yhteydessä kirjastojen kautta Subversioniin, on se herkempi häiriöille, kuin tyypillisimmät tietokannat. Esimerkiksi useimmissa SQL-järjestelmissä on oma palvelinprosessi, joka hoitaa pääsyn kaikkiin taulukoihin. Mikäli tietokantaan pääsy jotenkin estyy, tietokannan taustalla toimiva palvelinohjelmisto havaitsee yhteyden katkenneen ja siivoaa mahdolliset epäselvyydet. Ja koska tietokannan taustalla pyörivällä palvelinohjelmistolla on ainoastaan pääsy taulukoihin, ei ole riskiä konflikteista lupien kanssa. Berkeley DB ei taas tue tämän kaltaista järjestelyä. Subversion sekä ohjelmat, jotka käyttävät Subversionin kirjastoja omaavat pääsyn suoraan taulukoihin. Mikäli jokin näistä ohjelmista kaatuu, voi tietokanta jäädä hetkellisesti tilaan, jolloin sinne ei ole pääsyä. Tämän ratkaisemiseksi Subversionin ylläpitäjän tulee pyytää Berkeley DB:tä palauttamaan viimeisin tallennuspiste, joka ei ole kovinkaan käytännöllinen tapa hoitaa asioita automatisoitujen toimintojen sijasta (Collins-Sussman, Filzpatrick & Pilato 2011, 145).

Vaikka Berkeley DB onkin suhteellisen nopea ja skaalautuva tietokanta, se soveltuu parhaiten yhdelle käyttäjälle yhdellä palvelimella, esimerkiksi tässä tapauksessa Apachen httpd tai svnserver (Collins-Sussman, Filzpatrick & Pilato 2011, 146).

FSFS

Toinen vaihtoehto arkistoinnille on FSFS, joka ei käytä tietokantaa. Tämä tallentaa versioihin liittyvät muutokset yhdeksi tiedostoksi, josta löytyvät kaikki arkiston versiot yhdestä alihakemistosta, tiedostot numeroituna. Tiedonsiirrot tapahtuvat eri alihakemistoista yksittäisille tiedostoille. Kun tiedonsiirtotiedosto on uudelleen nimetty ja siirretty versiohakemistoon, voidaan olla varmoja tämän toimivuudesta. Ja koska versiotiedosto on muuttumaton, voi arkisto tehdä varmuuskopioita muun järjestelmän toimiessa samaan tapaan kuin Berkeley DB:ssä (Collins-Sussman, Filzpatrick & Pilato 2011, 146).

FSFS versiotiedostot kuvailevat version hakemistorakennetta, tiedoston sisältöä sekä tiedostojen muutoksia muihin versiopuihin verrattaessa. Toisin kuin Berkeley DB:n tietokanta, tämä tallennustapa mahdollistaa siirtymisen erikäyttöjärjestelmien välillä, eikä ole sidottu tietokoneen keskusyksikön toimintaan. Koska FSFS ei käytä jaettua muistia, arkistoihin on turvallista antaa pääsy myös verkon yli tapahtuvilla tiedostojärjestelmillä ja antaa niiden toimia siellä lukutilassa. Kun ei ole myöskään tietokantaa käytössä, arkisto vie vähemmän tallennustilaa (Collins-Sussman, Filzpatrick & Pilato 2011, 146).

Kun paljon tiedostoja sisältäviä hakemistoja tallennetaan kerralla, FSFS kykenee lisäämään näitä nopeasti. Toisaalta FSFS on taas tallennuksien viimeistellyssä hitaampi kuin Berkeley DB, mikä voi ääritapauksissa aiheuttaa aikakatkaisun viiveellisen vastauksen myötä (Collins-Sussman, Filzpatrick & Pilato 2011, 146).

Kaikkein tärkeimmäksi eduksi FSFS:lle voidaan määritellä sen toiminta vikatilanteissa. Tilanteet, joissa Berkeley DB:n kaatuessa tarvittaisiin ylläpitäjää hoitamaan palautus, ei FSFS:ssä aiheudu häiriöitä. Pahimmissa tapauksissa FSFS:ssä osa tiedonsiirrosta jäisi jälkeen (Collins-Sussman, Filzpatrick & Pilato 2011, 146).

FSFS toimii siis paremmin useamman henkilön työskennellessä samalla palvelimella, jolloin tämä on yrityskäytössä parempi. Sen joustavuuden, verkon kautta käytettävyyden ja ylläpidon tarpeen vähäisyyden myötä sitä voi käyttää myös huomattavasti vaivattomammin.

3.3.2 Versionhallinta Subversionilla

Aikaisemmin opinnäytetyössä käsiteltiin versionhallinnan perusteita. Tämän osion tarkoitus on perehtyä tähän nimenomaan Subversionin näkökulmasta. Vaikka toiminta muistuttaakin normaalia keskitettyä versionhallintaa, se sisältää kuitenkin yksityiskohtia ja poikkeavuuksia niistä.

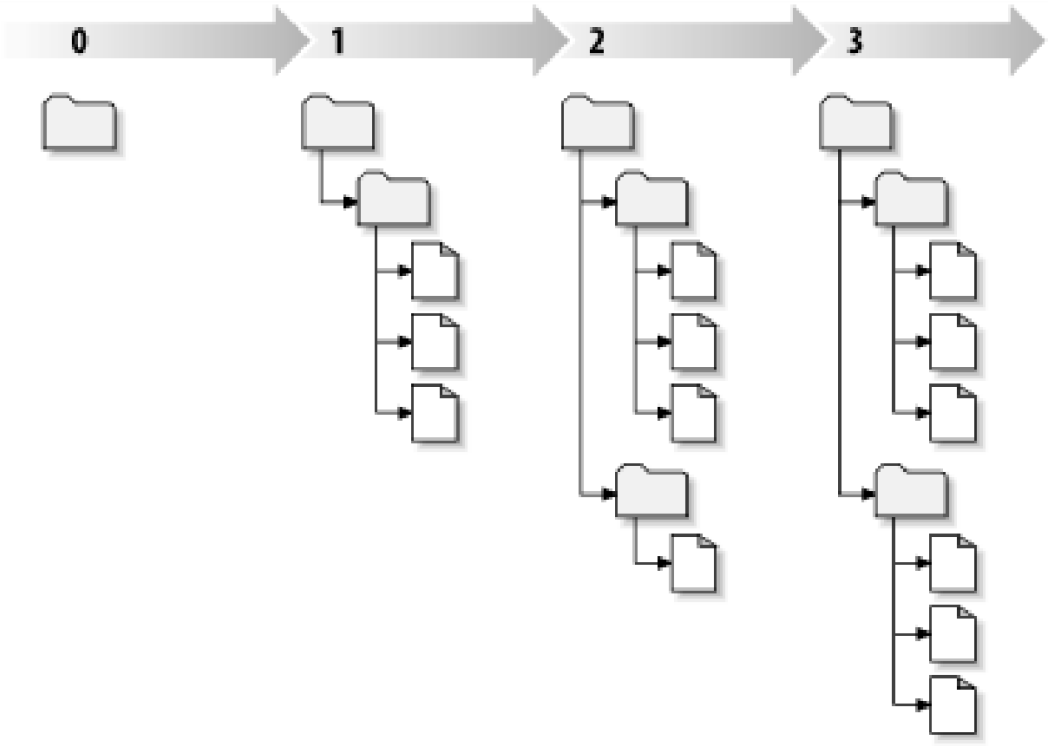
Arkistot

Subversionin arkistointi ei poikkea juurikaan muista keskitetyistä versionhallintajärjestelmistä. Tämä tarkoittaa sitä, että toisin kuin työkopiot, Subversionin arkistot ovat eristettyjä kokonaisuuksia, joita käsittelevät lähes poikkeuksetta Subversionin omat kirjastot ja työkalut. Suurin osa itse Subversionin käytöstä liittyy asiakasrajapintaan ja työkopioihin, joten on syytä käsitellä aihetta niiden hallinnan näkökulmasta (Collins-Sussman, Filzpatrick & Pilato 2011, 7).

Versiointi

Asiakas voi tallentaa kerralla niin paljon tiedostoja ja hakemistoja kuin haluaa. Tämä tapahtuu kuitenkin kokonaisena siirtona, joka tarkoittaa sitä, että käytännössä on vain kaksi tilaa tallennuksessa. Joko kaikki muutokset hyväksytään arkistossa tai mitään muutosta ei hyväksytä. Subversionissa onkin pyritty säilyttämään tämä kaikissa toiminnoissa, esimerkiksi ohjelman tai järjestelmän kaatuessa sekä verkko- ja käyttäjäongelmissa (Collins-Sussman, Filzpatrick & Pilato 2011, 7).

Aina kun arkisto hyväksyy tallennuksen, se luo uuden tason tiedostopuuhun. Jokainen versio tiedostopuusta saa uuden yksilöllisen versionumeron. Tämä poikkeaa useista versionhallinta ohjelmista siten, että versionumero koskee koko tiedostopuuta, eikä yksittäistä tiedostoa. Numerointi lähtee nollasta ja kasvaa aina hyväksytyn tallennuksen tapahduttua yhdellä numerolla. Tätä toimintaa kuvataan kuvassa 6 (Collins-Sussman, Filzpatrick & Pilato 2011, 8).



KUVA 6. Puun kasvaminen (Collins-Sussman, Filzpatrick & Pilato 2011, 8)

Kuvasta 6 on havaittavissa, että sama arkisto voi sisältää useita hakemistoja. Näiden hakemistojen sijainti vastaa siis koneenpaikallista hakemistojen arkkitehtuuria. Mikäli hakemistoja lisätään, se luo niitä lisää, kuten tiedostoja.

Arkiston osoitteen määrittely

Subversionin asiakasohjelmat käyttävät URL (Uniform Resource Locator) merkkijonoja yksilöitäessä versioituja tiedostoja ja hakemistoja Subversionin arkistossa. Pääasiassa näissä merkkijonoissa käytetään tavanomaista syntaksia, jolloin on mahdollista määrittää palvelimen nimi sekä porttinumero osana URL-osoitetta (Collins-Sussman, Filzpatrick & Pilato 2011, 8).

Tyypillisesti URL-osoitteet käyttävät http:tä (Hypertext Transfer Protocol) eli hypertekstin siirtoprotokollaa. Subversion mahdollistaa kuitenkin muidenkin protokollien käytön. Tämä on tarpeellista, sillä asiakkaalla on mahdollisuuksia kommunikoida eri tavoilla palvelimen kanssa ja sen avulla voidaan valita käytettävä mekanismi. Taulukossa 2 esitetään, mitä kaikkia URL-osoitteen pohjalla toimivia tiedonsiirtomalleja Subversion tukee ja mitä käytetään missäkin tapauksissa (Collins-Sussman, Filzpatrick & Pilato 2011, 8).

TAULUKKO 2. Tiedonsiirto (Collins-Sussman, Filzpatrick & Pilato 2011, 8)

Malli	**Yhteystekniikka**
file:///	Suora yhteys arkistoon paikallisella-asemalla
http://	Yhteys WebDAV-protokollalla Subversionin tuntemalle Apachen-palvelimelle
https://	Sama kuin http://, mutta käyttää myös SSL (Security Sockets Layer) kotelointia salatakseen tiedon
svn://	Yhteys itseluodulla protokollalla svnserver palvelimelle
svn+ssh://	Sama kuin svn://, mutta tieto kulkeutuu SSH (Secure Shell)-linjaa

Subversionilla URL:iä käyttäessä on joitakin tiettyjä asioita kuitenkin tiedettävä. Esimerkiksi, mikäli kyseessä on file:/// yhteystekniikka, tulee palvelin nimetä joko localhostiksi tai jättää kokonaan nimeämättä (Collins-Sussman, Filzpatrick & Pilato 2011, 9).

Kun käytetään file:/// mallia Windows-käyttöjärjestelmällä, täytyy käyttää epävirallista syntaksia, esimerkiksi X-levyllä file:///X:/”polku” tai file:///X|/”polku”. Tämän avulla voidaan päästään samalla koneella sijaitseviin kansioihin, jotka sijaitsevat kuitenkin eri levyllä, kuin asiakkaan käyttämä levy on (Collins-Sussman, Filzpatrick & Pilato 2011, 9).

Huomionarvoista on, että URL käyttää eteenpäin olevia kenoviivoja, kun taas Windows käyttää poluissaan taaksepäin olevia kenoviivoja. Mikäli käytät syntaksia file:///X|/ komentorivillä, tulee URL-osoite laittaa lainausmerkkien sisään, jottei pystysuuntaista viivaa tulkita väärin. Subversion myös kääntää osoitteen koodiksi, samaan tapaan kuin verkkoselaimet. Osoitteen, joka sisältää esimerkiksi välilyöntejä ja normaalin ASCII (American Standard Code for Information Interchange) ulkopuolisia merkkejä, Subversion tulkitsee eri tavalla verrattuna alkuperäiseen muotoon:

URL-aluksi: http://host/path with space/project/españa
Tulkittu: http://host/path%20with%20space/project/espa %C3%B1a

Kuten tästä esimerkistä havaitaan, muuttuu osoite merkittävästi, jos SVN sitä tulkitsee. Tässäkin tapauksessa komentoriviltä käytettäessä tulisi osoite laittaa lainausmerkkeihin, jotta se tulkittaisiin yhdeksi muuttujaksi ohjelmalle (Collins-Sussman, Filzpatrick & Pilato 2011, 9).

Työkopiot

Subversionissa työkopiot on tavallinen hakemistopuu, joka sisältää tiedostot. Näitä tiedostoja voi muokata ja käyttää myös lähdekooditiedostoina normaaliin tapaan ohjelmistoilla. Työkopiot ovat käytännössä siis henkilökohtainen työalue käyttäjälle, jota voi muuttaa ainoastaan omalla toiminnalla. Subversion ei siis tee mitään muutoksia ilman, että käyttäjä haluaa sen tekevän niitä. Jos käyttäjä tekee tiedostoihin muutoksia, Subversion mahdollistaa niiden julkaisun toiselle käyttäjällä. Kun taas toinen käyttäjä tekee muutoksia tiedostoihin, Subversion mahdollistaa näiden tietojen liittämisen omiin työkopioihin hakemalla ne arkistosta (Collins-Sussman, Filzpatrick & Pilato 2011, 9).

Työkopiot sisältävät joitakin ylimääräisiä Subversionin luomia ja hallitsemia tiedostoja, joita se tarvitsee suorittaakseen käskyt. Nämä sijaitsevat .svn nimisessä kansiossa työkopion alla, jota kutsutaan myös ylläpidon kansiossa. Tällä olevat tiedostot auttavat Subversionia tunnistamaan työkopioissa tehdyt muutokset sekä mitkä tiedostot eivät ole ajan tasalla muiden käyttäjien arkistoitujen muutoksien kanssa. Tämä sijaitsee työkopion hakemiston juuressa ja sisältää käytännössä metatietoa (Collins-Sussman, Filzpatrick & Pilato 2011, 10).

Jokaisesta työhakemistosta olevasta tiedostosta Subversion arkistoi muiden toiminnan kannalta kaksi tärkeää tietoa. Ensimmäinen sisältää tiedon siitä, mihin versioon työtiedosto perustuu. Tätä kutsutaan tiedostojen työversioksi (working revision). Toinen sisältää taas aikaleiman siitä, koska paikallinen työkopio on päivitetty arkistosta. Näiden tietojen avulla Subversion kykenee päättelemään, missä tilassa työtiedosto on. Näitä tiloja on neljä kappaletta, joiden avulla käyttäjä sekä Subversion voi päätellä tarpeelliset toimenpiteet, joita ovat näiden tapauksen käsittelyä varten käytännössä tallennus (svn commit) ja päivitys (svn update) (Collins-Sussman, Filzpatrick & Pilato 2011, 10).

Ei muokattu ja ajankohtainen:

Tiedostoa ei ole muokattu työhakemistossa ja siihen ei ole tehty muutoksia viimeisen arkistosta tehdyn päivityksen jälkeen. Tällöin ei voida tehdä tallennusta tai päivitystä tiedostolle, koska ei ole mitään tehtävää (Collins-Sussman, Filzpatrick & Pilato 2011, 10).

Paikallisesti muokattu ja ajankohtainen:

Tiedostoa on muokattu työhakemistossa ja sille ei ole muiden käyttäjien toimesta tehty muutoksia arkistoon. Tällöin paikalliset muutokset ei ole tallennettuna arkistoon, jolloin voidaan tehdä tallennus mutta ei päivitystä (Collins-Sussman, Filzpatrick & Pilato 2011, 10).

Ei muokattu ja vanhentunut:

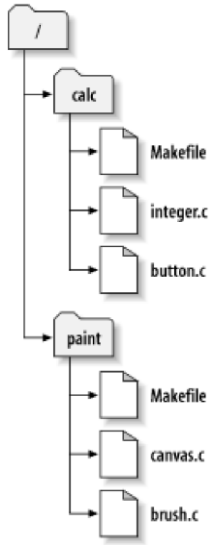
Tiedostoa ei ole muokattu työhakemistossa, mutta siihen on tehty muutoksia muiden käyttäjän toimesta arkistoon. Tällöin työhakemisto tulisi päivittää, jotta viimeisimmät arkistoon tehdyt muutokset tulisivat voimaan myös työhakemistossa. Tallennus komento ei tässä tapauksessa tee mitään (Collins-Sussman, Filzpatrick & Pilato 2011, 10).

Paikallisesti muokattu ja vanhentunut:

Tiedostoa on muokattu sekä työhakemistossa ja siihen on muiden käyttäjien toimesta tehty muutoksia myös arkistoon. Tällöin käyttäjän yrittäessä tallentaa sitä ennen päivitystä, Subversion antaa virhe ilmoituksen vanhentumisesta (out-of-date). Tiedosto tulee päivittää siis ensin, jolloin päivityskomento yrittää yhdistää arkistoon tehdyt muutokset paikallisiin muutoksiin. Mikäli Subversion ei onnistu tässä itsenäisesti, täytyy käyttäjän ratkaista konfliktit sisällön kanssa (Collins-Sussman, Filzpatrick & Pilato 2011, 10).

Työkopioiden vuorovaikutuksen perusteet

Useissa tapauksissa Subversionin arkisto sisältää useiden projektien tiedostoja. Nämä halutaan usein asettaa siten, että jokainen projekti on oma alihakemistonsa tiedostopuussa. Tässä tapauksessa käyttäjän työkopio voi sisältää siis vain tietyn haaran arkiston puusta. Esimerkiksi jos arkisto sisältäisi kaksi ohjelmistoprojektia, paint ja calc, molemmat projektit olisivat omassa päätason alihakemistossa kuvan 7 mukaisesti (Collins-Sussman, Filzpatrick & Pilato 2011, 10).



KUVA 7. Arkiston tiedostojärjestelmä (Collins-Sussman, Filzpatrick & Pilato 2011, 11)

Jotta saadaan työkopio, täytyy se hakea arkistosta (check out). Esimerkiksi saadaksesi arkistosta projektin calc, tulee se hakea arkistosta työkopioksi komentoriviä käyttäessä esimerkkitapauksessa ensin syöttämällä komento:

svn checkout http://svn.example.com/repos/calc

Tämän jälkeen komentoriville ilmestyisi komennon perään:

```
A         calc/Makefile
A         calc/integer.c
A         calc/button.c
Checked out revision 56.
ls        -A calc
Makefile  button.c integer.c .svn/
```

A-kirjain merkitsee marginaalissa Subversionin lisäävän kohteita työkopioihin. Tämän toiminnon jälkeen käyttäjän koneella olisi henkilökohtainen kopio arkiston /calc-hakemistosta sekä yksi tiedonkirjasta varten oleva .svn hakemisto, josta aikaisemmin mainittiin. Kuten havaitaan, kyseessä on 56. versio arkistosta. Mikäli nyt tekisi muutoksia esimerkiksi tiedostoon button.c, .svn hakemisto havaitsi sen, jonka jälkeen voisit tallentaa muokatun tiedoston. (Collins-Sussman, Filzpatrick & Pilato 2011, 11.) Komentorivillä tämä muokkaus tehdään kirjoittamalla:

svn commit button.c -m “Fixed a typo in button.c.”

Jonka jälkeen komentoriville tulee seuraava rivistö automaattisesti:

Sending button.c
Transmitting file data .
Commited revision 57.

Nyt tehdyt muutokset button.c-tiedostoon on tallennettu arkistoon, joka sisältää myös kuvailun. Jos toinen käyttäjä olisi hakenut kyseisen arkiston työkopioksensa samaan aikaan, niin tällä hetkellä hänen työkopionsa ei olisi ajan tasalla button.c tiedoston osalta. (Collins-Sussman, Filzpatrick & Pilato 2011, 11.) Tällöin toinen käyttäjä voi tehdä Subversionilla päivityksen työkopioihin kirjoittamalla:

svn update

Tämän jälkeen Subversion aloittaisi päivittämisen ja komentoriville tulisi lukemaan:

U button.c
Update to revision 57.

Nyt molemmilla käyttäjillä olisi ajankohtaiset arkistoversiot työkopioina. Huomion arvoista on se, ettei käyttäjän tarvitse päivittäessään yksilöidä päivitettävää tiedostoa, vaan .svn-hakemiston avulla pystytään suorittamaan päivitys kaikille tiedostoille, jotka eivät ole ajan tasaisia (Collins-Sussman, Filzpatrick & Pilato 2011, 12).

Ulkopuolisten toimintojen käynnistäminen

Subversion tarjoaa mahdollisuuden suorittaa ulkopuolisia toimintoja erilaisten kytkimien (hooks) avulla. Esimerkiksi kun halutaan suorittaa jokin ohjelma ennen tallennuksen tekemistä. Tällöin tulee käyttää ennen tallennusta kytkintä (pre commit hook). Ohjelman suoritus aloitetaan välittömästi, kun käyttäjä painaa arkistoon tallennuspainiketta ja vasta ohjelman suoriuduttua kokonaan tapahtuu tallennus. Taulukossa 3 esitellään Subversionin tarjoamat kytkimet (Collins-Sussman, Filzpatrick & Pilato 2011, 148).

TAULUKKO 3. Kytkimien toiminta (Collins-Sussman, Filzpatrick & Pilato 2011, 405 - 414).

Kytkimen nimi	Toiminnan kuvaus
start-commit	Ajetaan ennen tallennuksen tiedonsiirto tapahtuman luomista.
pre-commit	Ajetaan ennen kuin tallennus siirtyy uudeksi versioksi.
post-commit	Ajetaan, kun uusi versio on tallennettu.
pre-revprop-change	Ajetaan ennen kuin version tila muuttuu käyttäjällä.
post-revprop-change	Ajetaan version tilan muututtua käyttäjällä.
pre-lock	Ajetaan, kun joku yrittää lukita polkua.
post-lock	Ajetaan sen jälkeen kun polku on lukittu.
pre-unlock	Ajetaan, kun joku yrittää avata polun lukitusta.
post-unlock	Ajetaan polun lukituksen avaamisen jälkeen.

Kuten taulukosta 3 nähdään, on kytkimiä useisiin eri tarkoituksiin. Kytkimiä käyttäessä tulee määrittää Subversionissa polku kyseiseen suoritettavaan tiedostoon ja tämän lisäksi missä arkiston hakemistoissa kyseinen kytkin vaikuttaa. Ei ole välttämättä ideaalia asettaa kytkintä siten, että se vaikuttaa koko arkistoon, sillä se voi hidastaa suuria tiedostomääriä siirrettäessä muutenkin hidasta toimintaa (Collins-Sussman, Filzpatrick & Pilato 2011, 148).

4 BECKHOFF TWINCAT

4.1 Beckhoff yleisesti

Keskittynyt toimittamaan avoimia automaatiojärjestelmiä. Nämä erottuvat monesta muusta kilpailijasta siten, että ne pohjautuvat PC-pohjaiseen ohjaustekniikkaan. Tuotteet kattavat kenttäväyläkomponentit, liikkeenohjaustuotteet, teollisuus-PC:t, ohjauspaneelit sekä automaatiosovelluksien ohjelmistot (beckhoff.com. 2020).

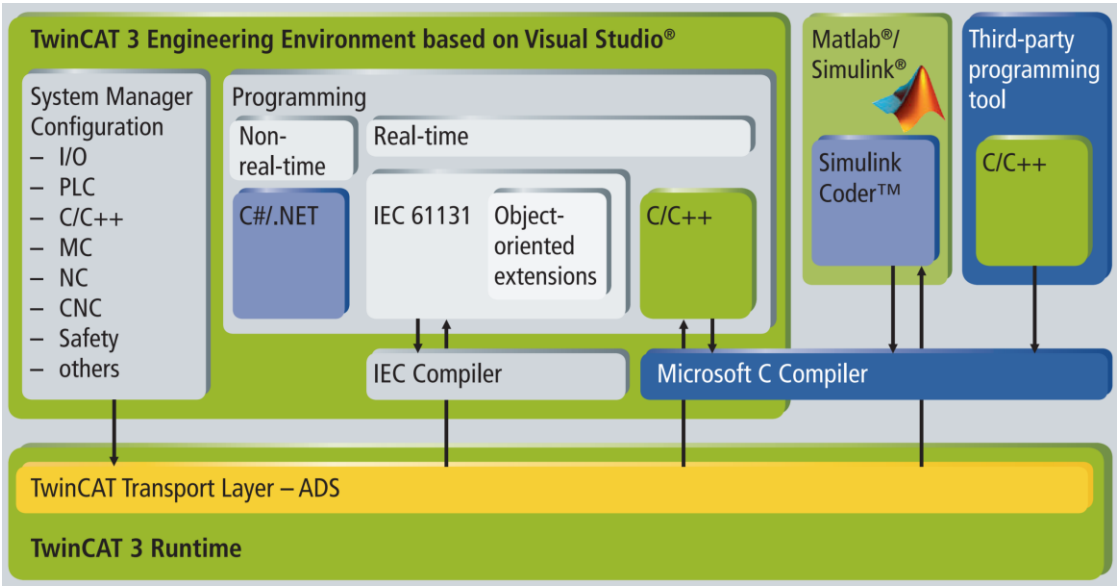
Beckhoff-yhtiön Suomen pääkonttori sijaitsee Hyvinkäällä, lisäksi toimisteitä on Tampereella, Seinäjoella ja Oulussa. Kaikista näistä konttoreista löytyy tekninen tuki, koulutus, tuotekehitys, myynti, sovellukset sekä huolto (beckhoff.com. 2020). Tämän opinnäytetyön toteutuksen yhteydessä on konsultoitu Tampereen toimipisteen teknistä tukea joidenkin yksittäisten asioiden selvittämisessä.

4.2 TwinCAT

TwinCAT on Beckhoffin automaatiosovellusohjelmisto. Kun tätä käytetään yhdessä Beckhoffin teollisuus-PC:n ja kenttäväyläkomponenttien kanssa, saadaan muodostettua avoin ohjausjärjestelmä. Tämä järjestelmä on liitettävissä muihin järjestelmiin, kun käytetään standardirajapintoja. TwinCAT-ohjelmisto integroi PLC-, NC- ja CNC-toiminnat reaaliaikaisesti toisiinsa. Ohjaimien ohjelmoinnissa on käytetty IEC 61131-3 -standardia. Standardi käsittelee ohjelmistoarkkitehtuuria sekä käytettäviä ohjelmointikieliä. Käytettävät ohjelmointikielet voidaan jakaa graafisiin ja tekstipohjaisiin ohjelmointikieliin. Näitä standardin mukaan ovat tikapuukaavio (Ladder diagram), toimilohkokaavio (Function Block diagram), sekvenssitoimilohkotaulukko (Sequential function chart), jäsennelty teksti (Structured text) sekä käskylista (Instruction list) (IEC 61131-3. 2013. 9).

4.3 TwinCAT 2:n erot TwinCAT 3:en

TwinCAT 3:n suurin ero TwinCAT 2:en on sen integrointi suoraan Microsoft Visual Studioon, joka on käytännössä maailman suosituin ja parhaiten tuettu ohjelmistojen kehitysympäristö. Tämä mahdollistaa sen, että kaikki halutut toiminnot kuten konfigurointi ja ohjelmointi, onnistuvat käyttämällä yhtä ohjelmaa, kun taas TwinCAT 2 vaati erillisen konfiguraatio-ohjelman. Lisäksi tähän on tuotu mahdollisuus käyttää C- sekä C++-ohjelmointikieliä, kun halutaan toteuttaa reaaliaikaisia sovelluksia. TwinCAT 3 mahdollistaa myös Matlabin ja Simulinkin yhdistämisen siihen, jolloin voidaan myös hyödyntää näitä esimerkiksi laskennassa. Kuvassa 8 on nähtävissä ohjelmistoympäristö kaaviona (beckhoff.com. 2020).



KUVA 8. TwinCAT 3 laajennettu automaatio suunnittelu (beckhoff.com. 2020)

TwinCAT 3 tukee myös kahden ytimen käyttöä ja 64-bittisiä järjestelmiä, joka on huomattavasti enemmän nykyaikaa tehokkaamman muistin käytön ja suojauksen kannalta. Lisäksi tämä 64-bittisen käyttöjärjestelmän hyöty tulee siinä, ettei ohjelmaa testattaessa tarvitse käyttää virtuaalikonetta, vaan TwinCAT 3 toimii nykyaikaisissa 64-bittisissä Windows-käyttöjärjestelmissä. Tämä nopeuttaa esimerkiksi suurien lähdekoodien lataamista ohjelmaan sekä vähentää ohjelmien kaatumista, jota tapahtuu virtuaalikonetta käyttäessä (beckhoff.com. 2020).

TwinCAT 3:n erona TwinCAT 2 -ohjelmaan on myös tapa, miten projektista luodaan tiedostot. TwinCAT 2 luo ainoastaan muutaman tiedoston, jotka sisältävät kaikki ohjelmallisesti luodut toiminnot sisällään binäärisesti. TwinCAT 3 luo taas kaikista toiminnoista ja ominaisuuksista erillisen tiedoston aina, joka on tekstimuodossa luettavissa. Esimerkiksi jos on luotu jokin yksittäinen toiminto jäsennellyllä teksti ohjelmoinnilla, tästä muodostuu projektin alakansioon uusi tiedosto, jonka voi avata millä tahansa tekstinkäsittelyohjelmalla ja ohjelman osuus on luettavissa siitä. Käytännössä on siis mahdollista muokata ohjelmaa esimerkiksi Notepadin avulla, mikäli näin haluaa tehdä. TwinCAT 2 projektia on taas mahdoton avata muilla editoreilla, johon on syynä binäärimuotoinen esitystapa (beckhoff.com. 2020).

4.4 TwinCAT 3 projekti

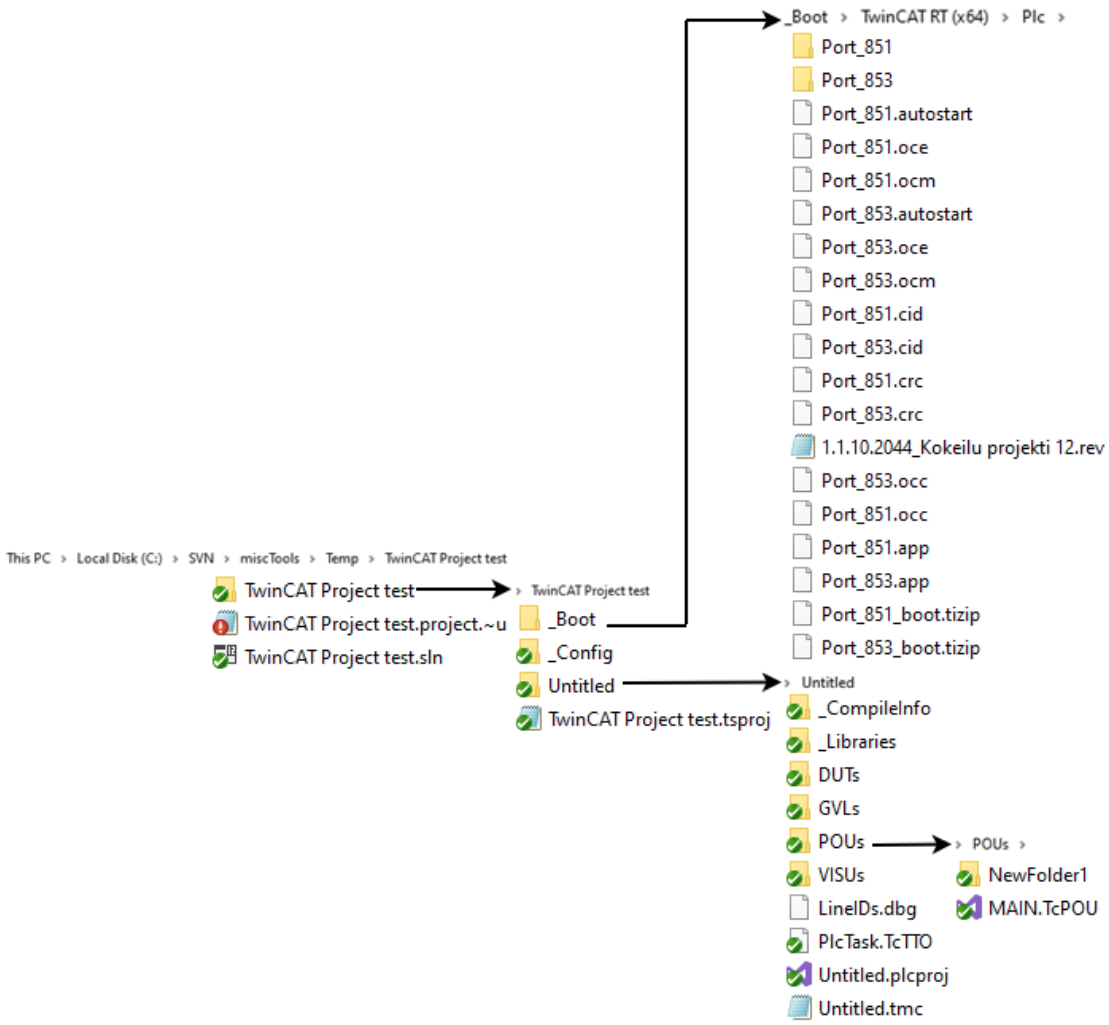
TwinCAT 3:lla on mahdollista käyttää erilaisia liitäntäsovelluksia, jonka avulla voidaan käyttää versionhallintaohjelmia TwinCAT 3:n kautta. Esimerkkinä näistä Visual Studion ja TortoiseSVN yhdistämiseen käytettävä VisualSVN. Tämän avulla saadaan integroitua suoraan TwinCAT 3:en TortoiseSVN:än ominaisuuksia, kuten esimerkiksi arkistoon tallennuksen tai päivityksen. Tämä on varteenotettava vaihtoehto, kun ajatellaan projektin tekemistä Visual Studio pohjaisella ohjelmalla. Tämän avulla vältytään esimerkiksi ylimääräisiltä ikkunan vaihdoilta tilanteissa, kun halutaan tallentaa jotakin palvelimelle. VisualSVN ei kuitenkaan ole mainittujen ohjelmien joukossa, joita Beckhoff sivuillaan (infosys.beckhoff.com. 2020) ilmoittaa tukevansa, toisin kuin esimerkiksi AnkhSVN. AnkhSVN kokeiltiin opinnäytetyön aikana. Tässä kuitenkin ongelmaksi ilmeni, ettei se kykene jostakin syystä käyttämään Subversionin kytkintä ennen tallennusta. Syynä tälle voi olla se, että TwinCAT 3:ssa eivät toimi kaikki Visual Studio .NET -projektien ominaisuudet.

TwinCAT 3:n (TC) tiedostot ovat hajautettuna useaan eri tiedostoon, jotta se tukee ryhmätyöskentelyä sekä versionhallintaa (VH) aidosti. Taulukossa 4 esitetään TwinCAT 3 –projektin sisältämät tiedostot (infosys.beckhoff.com. 2020).

TAULUKKO 4. TwinCAT 3 projektin tiedostot (infosys.beckhoff.com. 2020)

Tiedosto-pääte	**VH**	**Yhdistäminen**	**Kuvaus**
*.tsproj	Kyllä	TC projektivertailijalla	TC projektitiedosto
*.plcproj	Kyllä	TC projektivertailijalla	TC PLC projektitiedosto
*.tmc	Kyllä	Ei sallittu PLC projekteille	TC moduuliluokka (kuvailutiedosto TcCom-moduulille)
*.tpy	Ei	-	Tiedosto muiden ohjelmistojen yhteensovittamiseen
*.xti	Kyllä	TC projektivertailijalla	Kun tuki tiedostoja on useita, osa TC –projektitiedostoista tallennetaan tällä päätteellä
*.TcTTO	Kyllä	TC projektivertailijalla	PLC:n tehtävien päämäärä
*.TcPOU	Kyllä	TC projektivertailijalla	PLC ohjelman organisointiyksikkö
*.TcDUT	Kyllä	TC projektivertailijalla	PLC: tietotyyppi
*.TcGVL	Kyllä	TC projektivertailijalla	PLC:n yleismaailmallinen muuttujalista
*.TcVis	Kyllä	Ei tukea tällä hetkellä	PLC:n visuaalisuus
*.TcVMO	Kyllä	Ei tukea tällä hetkellä	PLC:n visuaalisuuden hallinta
*.TcGTLO	Kyllä	-	PLC:n yleismaailmallinen tekstilista
*.sln	Ei	-	VS sovellustiedosto. Se sisältää kaiken muun tiedon lisäksi tunnisteen siitä, mitä VS versiota käytetään.
*.suo	Ei	-	Käyttäjän valikkotiedosto VS projektille. Se sisältää tietoa valitusta alustasta, pysäytyspisteistä ja muusta vastaavasta. Käyttäjäkohtainen. Muodostuu kun projekti avataan ensimmäistä kertaa.

Taulukko 4:ssä näkyvä .plcprj-tiedosto on opinnäytetyön kannalta yksi tärkeimmistä tiedostoista. Se sisältää kaiken projektin tarvitseman tiedon XML-muodossa versionumerosta kuvaukseen. Kuvassa 9 on näkyvillä, kuinka TwinCAT 3:n hakemisto rakentuu käyttäjän koneelle.



KUVA 9. TwinCAT 3 projektin hakemistoarkkitehtuuri TSVN alaisena

Kuvassa 9 sekä taulukossa 4 näkyvä .sln-tiedosto eli sovellustiedosto (Solution) on se, missä TwinCAT 3:n kaikki ohjelman kehitystyö tehdään. Tämä käyttää kaikkia loppuja tiedostoja tukenaan toimiakseen ja on siis se ympäristö, missä PLC-ohjelmointi tehdään.

TwinCAT Project test -kansiossa taas näkyy kansio nimeltä Untitled. Sen nimen voi käyttäjä määritellä itse, mutta tässä tapauksessa siinä näkyy vain oletusnimi. Tämän alle voi tehdä erillisen projektikohtaisen hakemiston, jolloin yhden TwinCAT 3 –projektin sisällä voi olla useampia projekteja.

POUs kansio projektikansion alapuolella sisältää taas kaikki PLC-ohjelmat, esimerkiksi kuvassa 9 näkyy MAIN.TcPOU, joka on PLC-ohjelman pääohjelma. Näiden tiedostojen sisältö on mahdollista lukea myös tekstieditorilla, jos ohjelma on tehty käyttämällä tekstipohjaisia ohjelmointimenetelmiä.

Kuvassa 9 näkyvä _Boot-kansio sisältää käytännössä kaiken, jota tarvitaan ohjelmoitavan logiikan käyttöön. Kuvan 9 tapauksessa sinne on suoritettavalla ohjelmalla lisätty .plcproj-tiedostosta versionumero sekä projektin kuvaus. Tästä löytyy lisää tietoa myöhemmistä kappaleista.

Kuten aikaisemmin on sanottu, TwinCAT 3 tuo projektin aidosti versionhallintaa. Tämä merkitsee sitä, että kun esimerkiksi muokataan PLC-ohjelmaa, on helppo tarkastella jälkikäteen, mitä osuutta on muokattu laajasta kokonaisuudesta, kun muutokset koskevat vain tiettyjä tiedostoja ja vain nämä tietyt tiedostot tallentuvat arkistoinnissa. Aikaisemmin TwinCAT 2:lla tehtiin muutos esimerkiksi tähän PLC-ohjelmaan, tallennettiin koko projektin PLC-ohjelma, sillä yksi tiedosto sisälsi tämän kokonaan. TwinCAT 3:n tapa helpottaa usean ihmisen työskentelyä saman projektin parissa, sillä konfliktien riskit pienenevät versionhallinnan palvelimelle tallennuksen yhteydessä, kun ihmiset eivät työstä samaa tiedostoa.

5 Suoritettavat ohjelmat

5.1 Suoritettavien ohjelmien tarkoitus

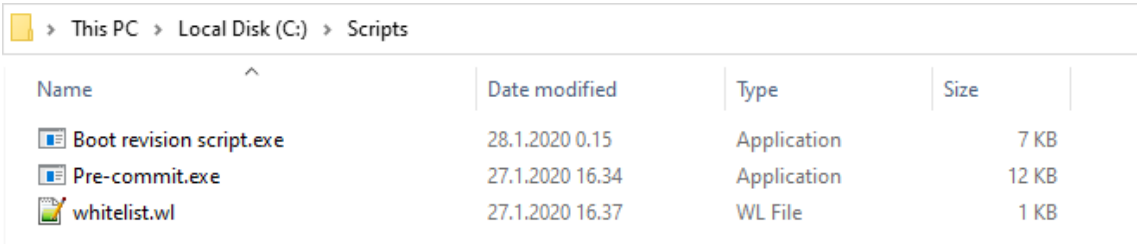
Suoritettavien ohjelmien eli skriptien tavoitteena on automatisoida työvaiheita, kehitettäessä ohjelmoitavien logiikoiden ohjelmia TwinCAT 3 -ympäristössä kun halutaan hallita ohjelman kehitysvaiheita sekä versioita versionhallinnassa. Käytännössä yrityksen tarpeena oli saada TwinCAT:in puolelle näkyviin automaation kehitysvaiheessa kyseessä olevan projektin versionumero, joka vastaa TortoiseSVN:än palvelimella olevaa. Tämän lisäksi oli tarve saada ohjelman käännön yhteydessä tuotettua versionumeron sisältävä tiedosto, jotta voitiin saada selville esimerkiksi asiakkaalla olevan ohjelman versionumero ilman sitä, että asiakkaalla olisi automaatioon käytettävä lähdekoodi itsellään hallussa.

Edellä mainittujen ongelmien ratkaisuiksi kehitettiin kaksi erillistä suoritettavaa ohjelmaa. Toisen ohjelman suorittaminen käynnistyy, kun TortoiseSVN kutsuu sitä tallennuksen yhteydessä. Toinen ohjelma suoritetaan taas, kun TwinCAT 3 kutsuu sitä projektia kääntäessä. Näiden käyttöön ja toimintaan perehdytään paremmin seuraavissa kappaleissa.

5.2 Skriptien käyttöönottaminen ja käyttäminen

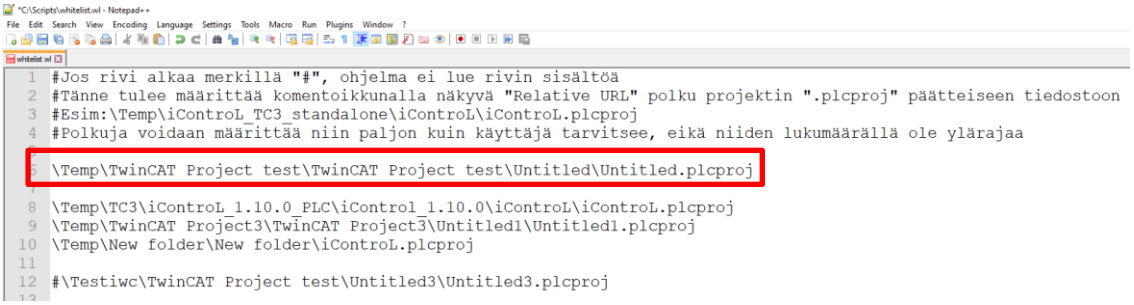
5.2.1 Skriptien käyttöönotto

Kuten aikaisemmin mainittiin, suoritettavia ohjelmia toteutettiin kaksi kappaletta. Seuraavaksi käydään läpi, kuinka nämä tulee ottaa käyttöön oikein, jotta ne saadaan toimimaan. Oletuksena on, että käyttäjällä on koneellansa asennettuna TwinCAT 3 sekä TortoiseSVN, johon asennettu myös komentorivityökalut. Esimerkkitapauksissa suoritettavat ohjelmat ovat sijoitettuna kuvassa 10 näkyvään sijaintiin paikallisesti käyttäjän koneella.



KUVA 10. Suoritettavien ohjelmien sijainti käyttäjän koneella

Kuvan 10 kansio sisältää myös listan poluista (whitelist) projektien .plcproj-päätteen omaaviin tiedostoihin, joihin käyttäjä haluaa toimillansa vaikuttaa suorittaessaan ennen tallennusta tapahtuvan ohjelman. Tässä yhteydessä käytettävän whitelist-tiedoston sisältö esitetään kuvassa 11.

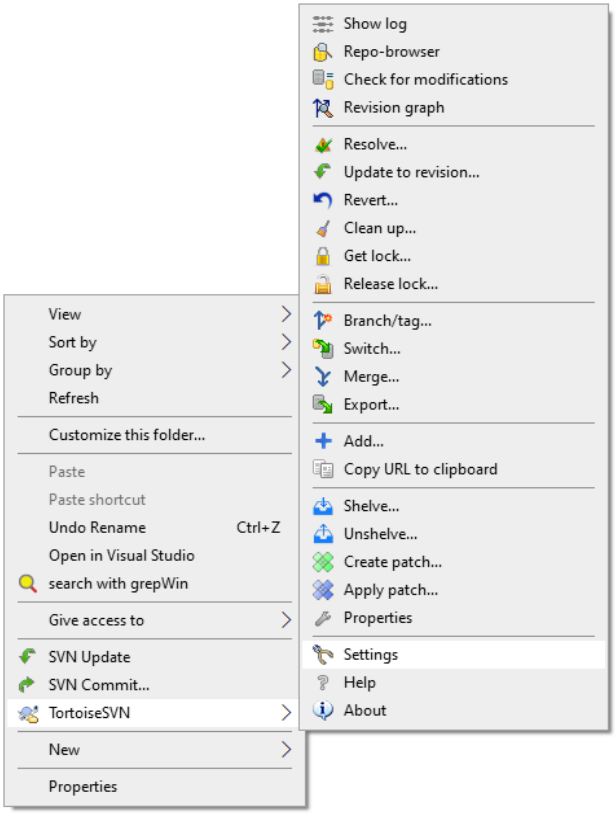


KUVA 11. Esimerkki whitelist-tiedoston sisällöstä

Kuten kuvasta 11 nähdään, voi tiedostoon lisätä kommentin tai poistaa jonkin polun käytön asettamalla alkuun #-merkin. Tällöin suorittava ohjelma ei lue tiedostosta kyseistä riviä, vaan siirtyy seuraavaan riviin. Kuvassa 11 näkyvä polku vastaa myöhemmin esimerkeissä olevaa polkua. Tämän polun saa selville avaamalla projektin hakemistosta komentoikkunan ja ajamalla svn info -komennon, jolloin tulostuvien tiedostojen joukosta löytyy suhteellinen URL (Relative URL) -polku. Jos polku sisältää välilyöntejä, näkyy se komento ikkunalla %20-merkinnöillä. Nämä tulee korvata tavallisiksi välilyönneiksi whitelist-tiedostoon. Tiedostoon voi myös lisätä haluamansa määrän polkuja, joista suoritettava ohjelma etsii omatoimisesti tapauskohtaisen polun tarpeeseensa.

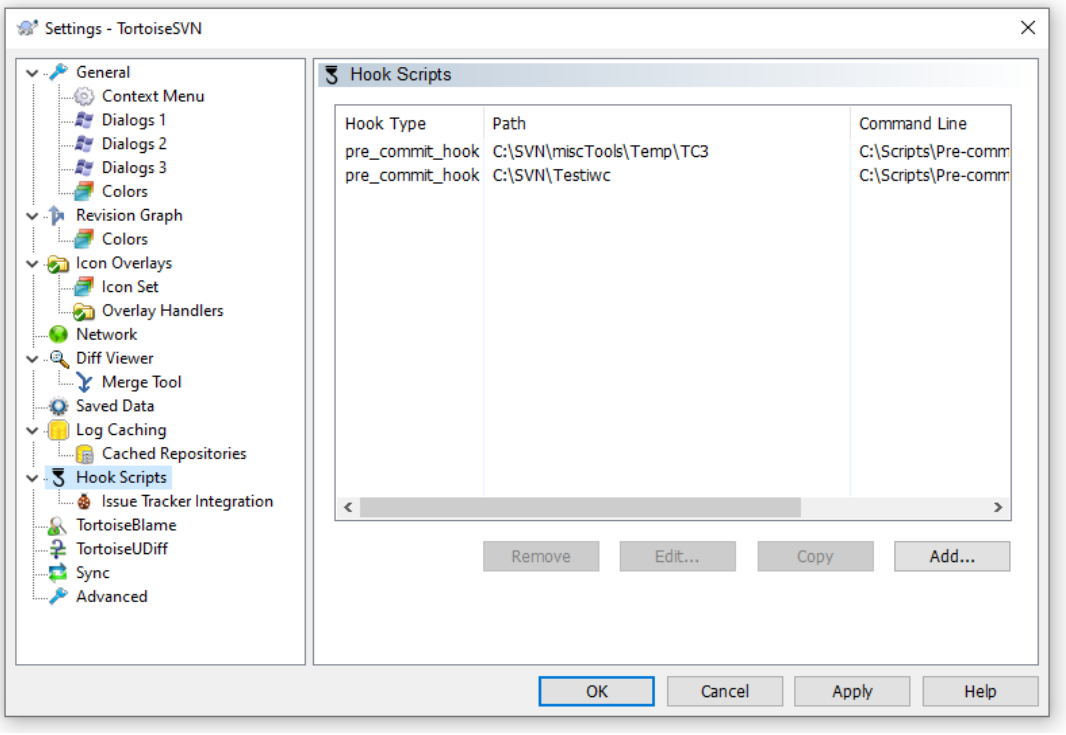
TortoiseSVN asiakaspuolen skriptin käyttöönotto

Käyttöönotto aloitetaan painamalla hiiren oikeanpuoleista painiketta, jolloin avautuu kuvassa 12 näkyvä valikko. Kuten kuvasta nähdään, on TortoiseSVN:än valikko integroitu suoraan Windowsin-valikkoon.



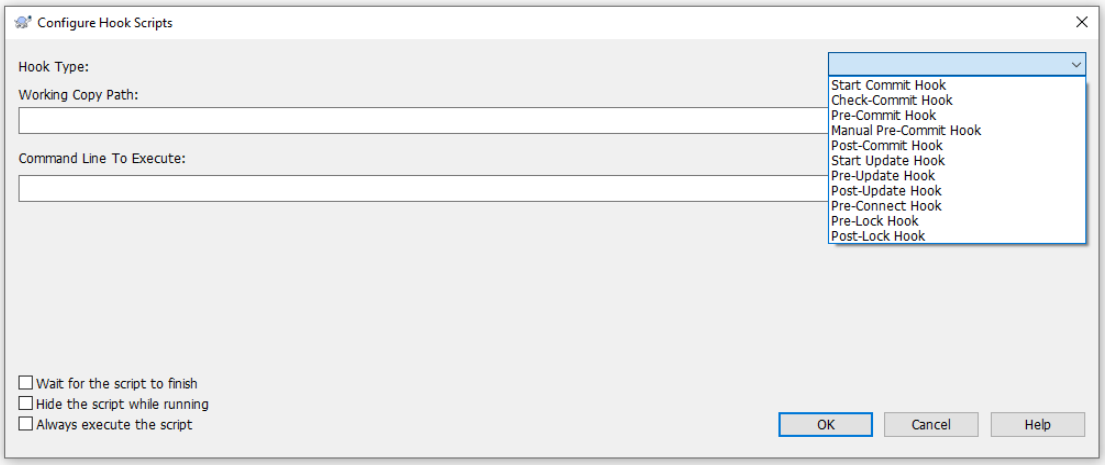
KUVA 12. TortoiseSVN valikko integroituna Windows-valikkoon

Kuvan 12 valikosta valitaan TortoiseSVN ja sieltä hakeudutaan asetuksiin (Settings). Tällöin saadaan avautua kuvan 13 TortoiseSVN:n asetukset.



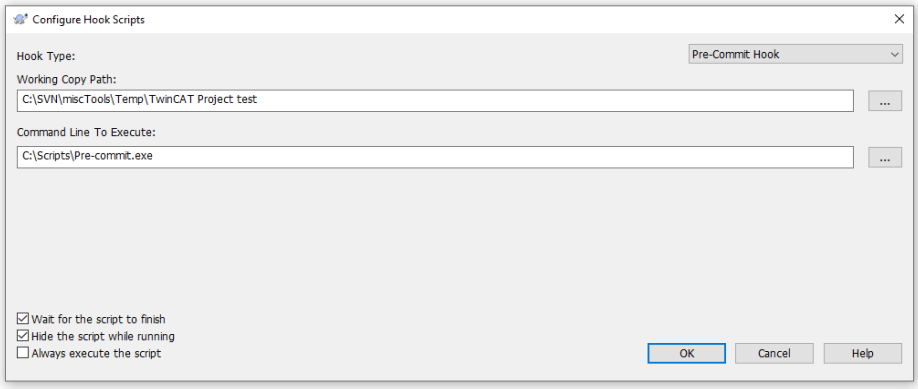
KUVA 13. TortoiseSVN asetukset

Valitaan kuvassa 13 näkyvällä tavalla suoritettavien ohjelmien kytkimet (Hook Scripts). Kuvassa 13 näkyvät aikaisemmin lisätyt ennen tallennusta suoritettavat kytkimet sekä niiden määrittelyt. Valitaan seuraavaksi lisäys (Add) oikeasta alakulmasta, jolloin päästään lisäämään uusi kytkin suoritettavalle ohjelmalle. Tällöin avautuu kuvan 14 suoritettavan ohjelman kytkimen konfigurointi (Configure Hook Scripts) –ikkuna.



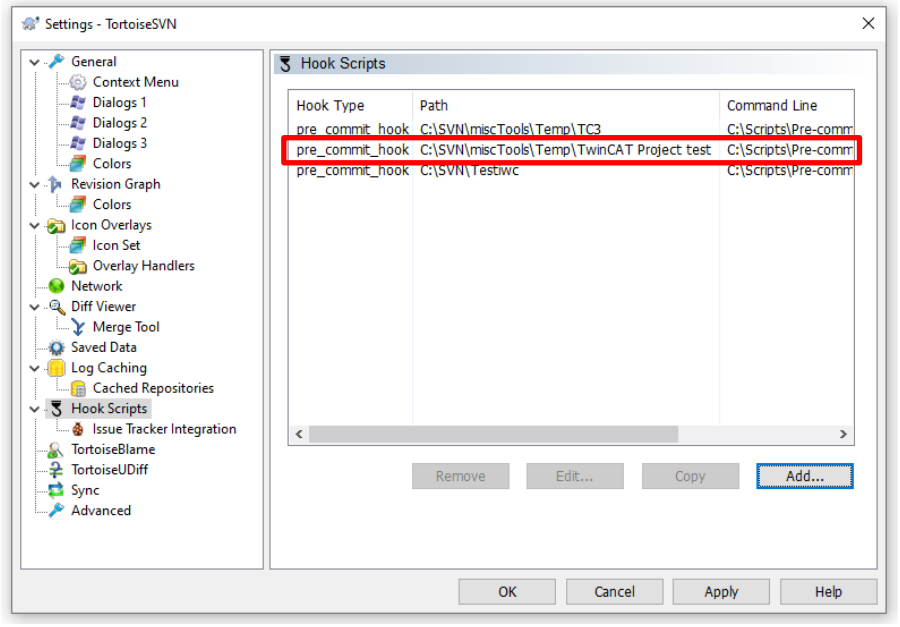
KUVA 14. Suoritettavan ohjelman konfigurointi-ikkuna

Kuvan 14 oikeassa yläkulmassa on valmiiksi avattuna liukuvalikko, josta valitaan kytkimen tyyppi taulukon 3 valikoimasta. Valitaan ennen tallennusta toimiva kytkin (Pre-commit Hook). Tämän jälkeen täytyy määritellä polku työkopioon (Working Copy Path), jonka hakemistossa tallennusta suorittaessa suoritettava ohjelma kytkeytyy päälle. Määritellään myös polku tiedostoon, jonka Subversion kytkimellä suorittaa (Command Line To Execute). Viimeisenä asetuksiin tulee laittaa vasemmassa alakulmassa merkinnät ohjelman suorituksen odottamiselle (Wait for the script to finnish) ja piilotetaan suoritettava ohjelmisto ajon aikana (Hide the script while running). Olisi vielä mahdollista pakottaa ohjelman ajo jokaisen tallennuksen yhteyteen (Always execute the script), mutta erilaisten konfliktitilanteiden ja muiden ongelmien helpottamiseksi kannattaa jättää käyttäjälle mahdollisuus ohittaa ohjelma. Lopputuloksen tulisi näyttää kuvaa 15 vastaavalta. Polut kuitenkin vaihtelevat käyttäjän omien tiedostojen ja projektien sijainnin mukaisesti.



KUVA 15. Ennen tallennusta suoritettavan ohjelman ajon konfiguroinnit

Kun painetaan kuvan 15 OK-painiketta, sulkeutuu ikkuna, jolloin palataan edelliseen ikkunaan. Konfiguroinnin jälkeinen tilanne näkyy kuvan 16 asetusikkunassa.



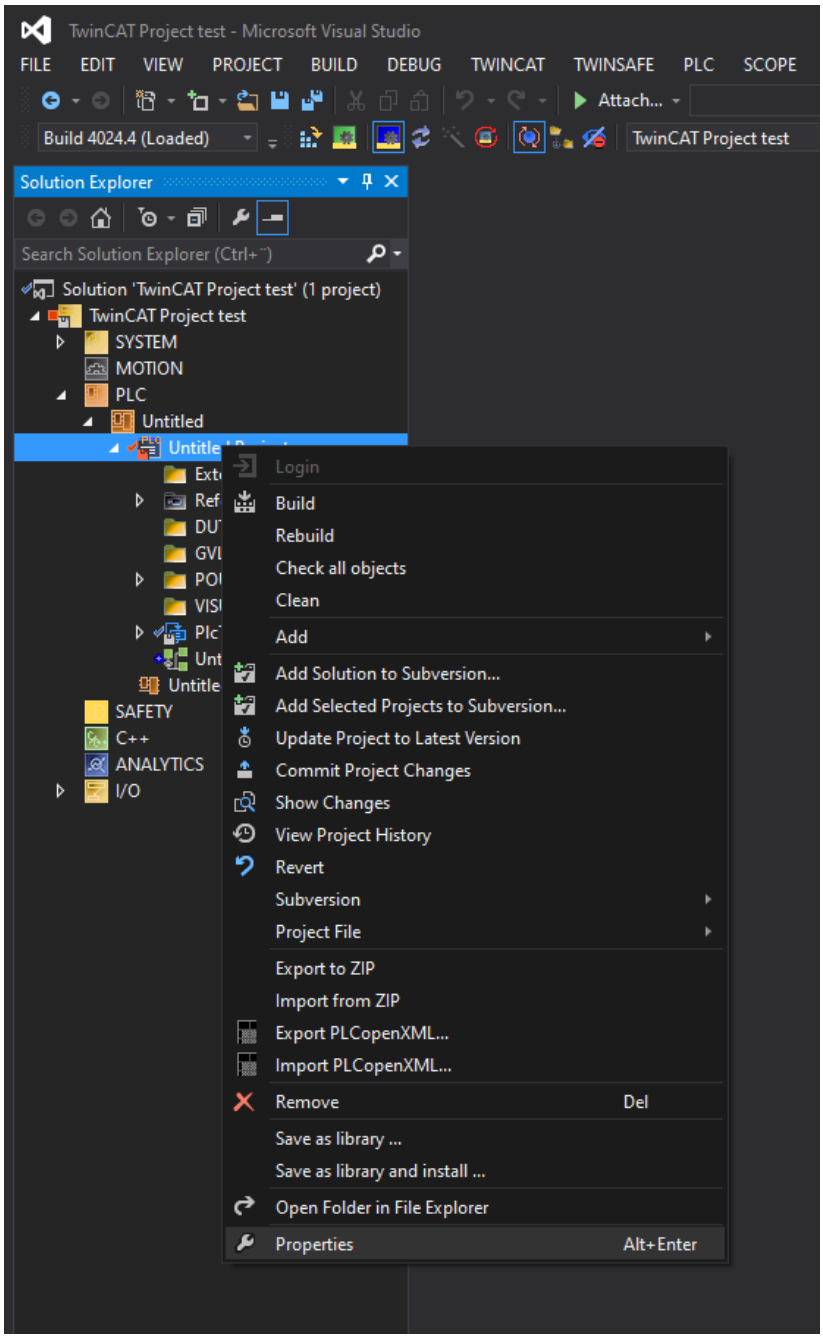
KUVA 16. Uusi ennen tallennusta suoritettavan ohjelman kytkin

Kuvan 16 erona kuvaan 13 havaitaan, että kytkinten joukkoon on ilmestynyt uusi kytkin, joka näkyy kuvassa 16 punaisella ympäröitynä. Uudet asetukset saa käyttöön, kun painetaan käyttöönotto (Apply) -painiketta.

Kun edellä mainitut toimenpiteet ovat tehty, on ennen tallennusta suoritettava ohjelma käytettävissä. Tiedostopolkujen kanssa on suositeltavaa olla erittäin tarkka, sillä muuten suoritettava ohjelma ei käynnisty, koska se ei voi ilman näitä tietoja tietää hakemistojen sijainteja tai käyttäjän tarpeita.

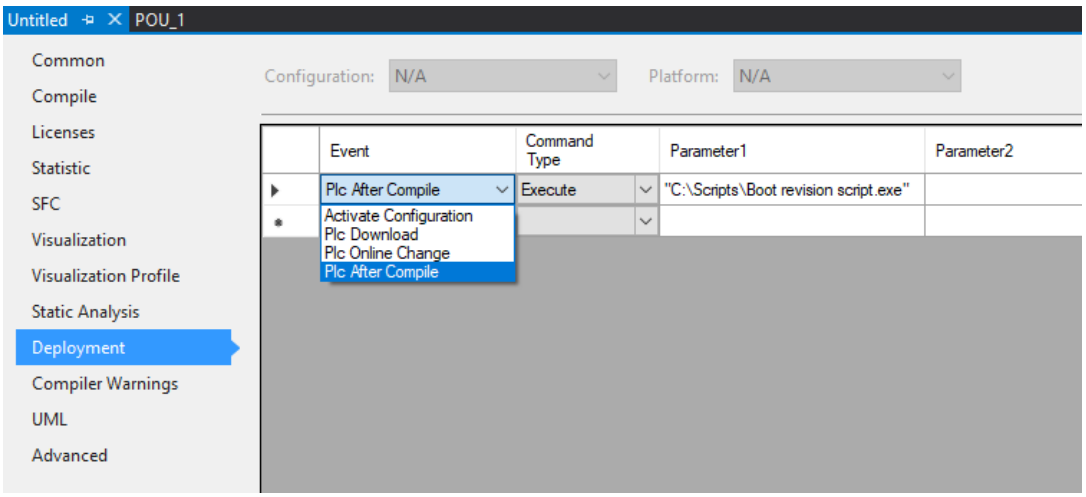
TwinCAT 3 käännöksen yhteydessä skriptin käyttöönotto

Kuvassa 17 näkyy osa TwinCAT 3:n käyttöliittymästä. Käyttöönotto aloitetaan TwinCAT 3:n puolella avaamalla kuvan 17 osoittamalla tavalla hiiren oikealla valikko projektiosiosta ja täältä tarpeisto (Properties).



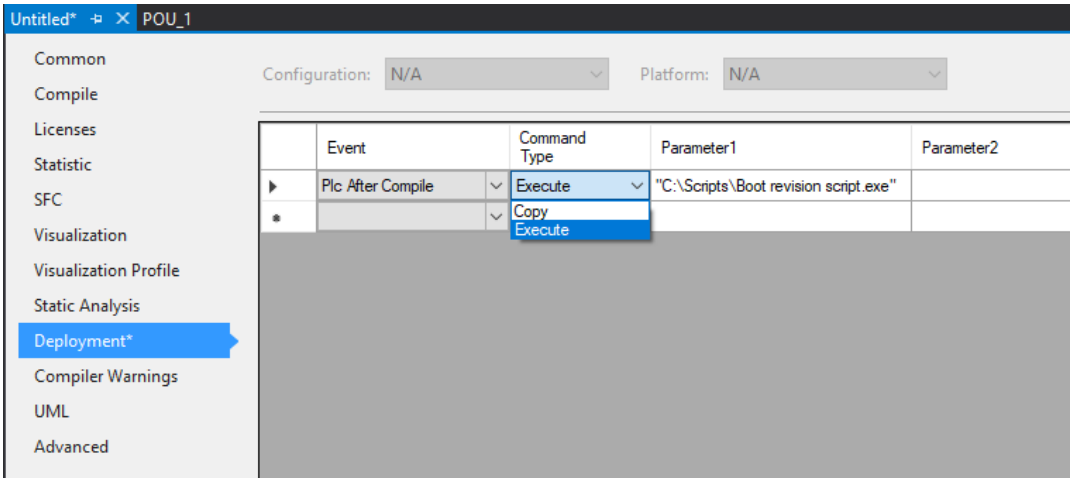
KUVA 17. TwinCAT 3:n valikko

Aukeaa ikkuna, josta valitaan kuva 18 mukaisesti asettelut (Deployment). Täältä määritellään ensimmäisenä, minkä tapahtuman (Event) yhteydessä toiminto toteutetaan. Ohjelman tarkoitus on toimia silloin, kun ohjelmoitavan logiikan ohjelma käännetään, joten valitaan Plc After Compile.



KUVA 18. Asettelutikkuna, tapahtumavalikko auki

Seuraavaksi siirrytään kuvassa 19 näkyvään komennon tyypin (Command Type) määrittelyyn. Tässä on kaksi vaihtoehtoa, joista toisella olisi mahdollista kopioida kuvassa 19 näkyvästä Parameter1 polusta Parameter2 polkuun tietty tiedosto halutessa. Tavoitteena on kuitenkin suorittaa ohjelma, joten valitaan suoritus (Execute).



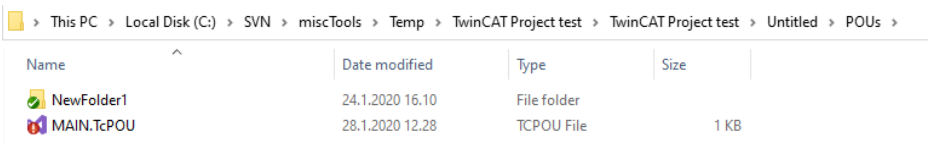
KUVA 19. Asetteluikkuna, komennon tyyppivalikko auki

Viimeisenä määrittelynä tarvitsee asettaa polku tiedostolle kuvassa 19 näkyvään Parameter1-kohtaan, joka halutaan suorittaa. Tämä tulee laittaa lainausmerkkien sisään, jotta vältytään välilyöntien ja muiden erikoismerkkien kohdalla polun lukuun liittyviltä virheiltä. Näiden toimenpiteiden jälkeen, suoritettava ohjelma suoritetaan aina kun tehdään ohjelmoitavan logiikan käännös kyseiselle projektille TwinCAT 3:lla.

5.2.2 Skriptien käyttäminen

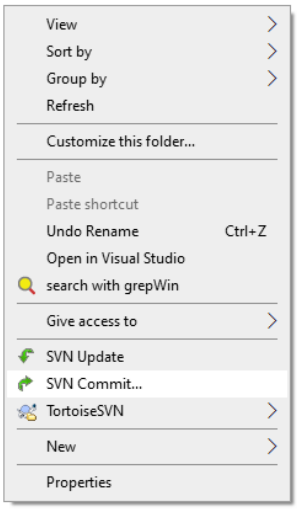
TortoiseSVN asiakaspuolen skriptin käyttäminen

Ohjelmaa käytetään, kun tarvitsee siirtää tietoa paikallisesta koneella olevasta työkopiosta arkistoon uudeksi versioksi. Kuten aikaisemmin sanottu, työkopioissa esitetään tiedostojen ja kansioiden tilatieto palvelimeen verrattuna. Kuvassa 20 on tilanne, jossa yksi tiedosto projektin alla olevissa kansioissa ei ole ajan tasalla. Tätä symboloi kuvan 20 MAIN.TcPOU-tiedoston kuvakkeen päällä näkyvä punaisella taustalla oleva huutomerkki. NewFolder1-kansiossa olevat tiedostot ovat ajan tasalla, jolloin sen symbolina on vihreällä taustalla V-kirjainta muistuttava merkki.



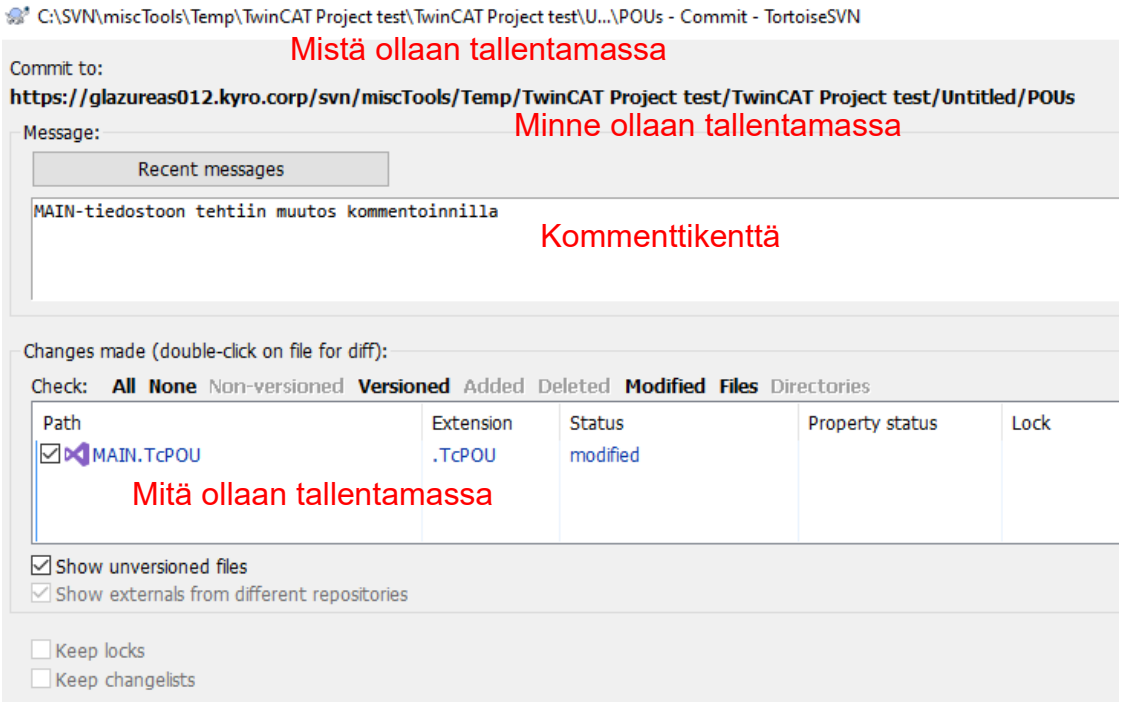
KUVA 20. Yksi tiedosto hakemistossa ei ole ajan tasalla

Painetaan hiiren oikealla painikkeella kyseisessä hakemistossa valitsematta tiettyä kohdetta. Aukeaa kuvan 21 Windows valikko, joka sisältää Subversion komentoja sekä ominaisuuksia.



KUVA 21. Windows valikko SVN ominaisuuksilla, tallennuspainike maalattu

Painetaan kuvan 21 maalattua Subversion tallennus (SVN Commit) –painiketta. Avautuu kuvan 22 mukainen ikkuna, josta nähdään mitä ollaan tallentamassa ja mihinkä tallennusta ollaan tekemässä.

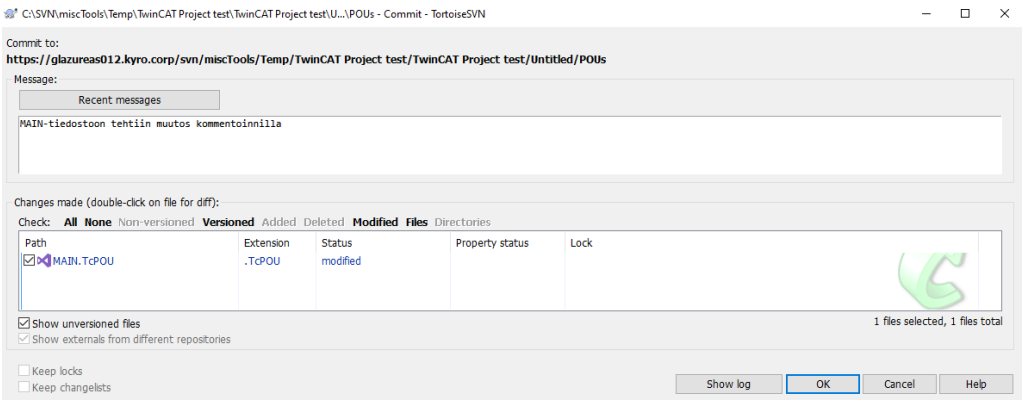


KUVA 22. Subversionin tallennusikkuna

Kuvaan 22 on merkitty selitteiksi eri osioille, mitä ne merkitsevät. Aivan ikkunan ylälaidassa näkyy työkopion sijainti paikallisesti käyttäjän koneella kansiotasolla ja sen alapuolella mihin tallennus ollaan tekemässä arkistossa.

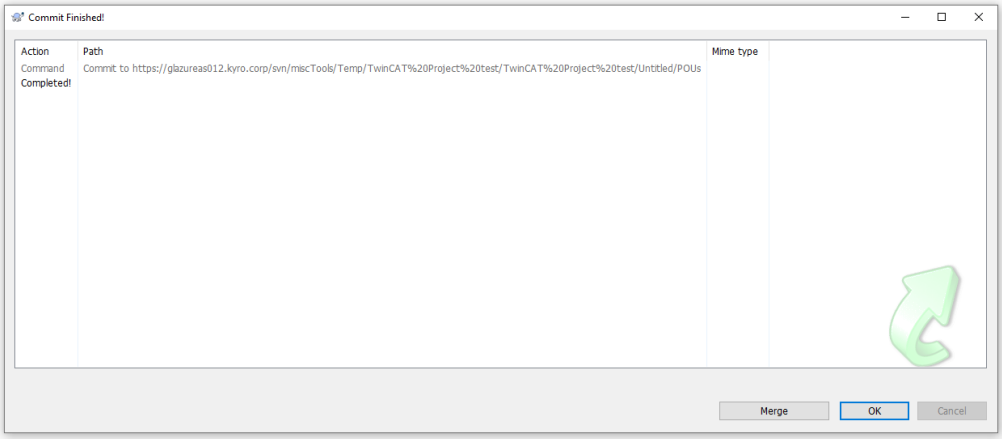
Kommenttikenttään käyttäjä voi taas lisätä oman kommenttinsa, mitä muutoksia hän tekee tiedostoon. Tämä tulee muille käyttäjille näkyviin, kun he tarkastelevat arkiston sisältöä.

Mitä ollaan tallentamassa osiossa, näkyy mitä tiedostoja ollaan kyseiseen kansioon nähden tallentamassa. Käytännössä tässä voisi olla todella suuri määrä tiedostoja tallennettavana, mutta asian yksinkertaistamiseksi tässä esimerkissä niin ei ole. Kuitenkin mikäli esimerkiksi kuvan 20 hakemistokuvassa näkyvässä NewFolder1-kansion alla olisi ollut muokkauksia tiedostoihin, näkyisi se tässä polkuna tiedostoon, joka alkaisi kyseisestä työkopiokansiosta. Tässä osiossa näkyy myös eriteltynä aina kyseisten tiedostojen tiedostopääte (Extension) sekä sen tila (Status). Tilan avulla voidaan havaita, että tiedostoon on tehty muokkauksia arkistossa olevaan versioon nähden. Tämä on syy, jonka takia tallennus tulee tehdä. Kuvassa 23 nähdään tämä ikkuna kokonaisuudessaan.



KUVA 23. Subversionin mitä ja minne ollaan tallentamassa

Kuvan 23 ikkunasta tulee seuraavaksi painaa OK-painiketta. Tällöin Subversion aloittaa tallennusprosessin ja aukeaa ikkuna, joka vastaa pääasiassa kuvassa 24 näkyvää ikkunaa. Aluksi kuvan 24 teksti kentässä ei näy mitään, koska se suorittaa ennen tallennusta ulkoisen ohjelman ennen tallennusta kytkimen avulla.

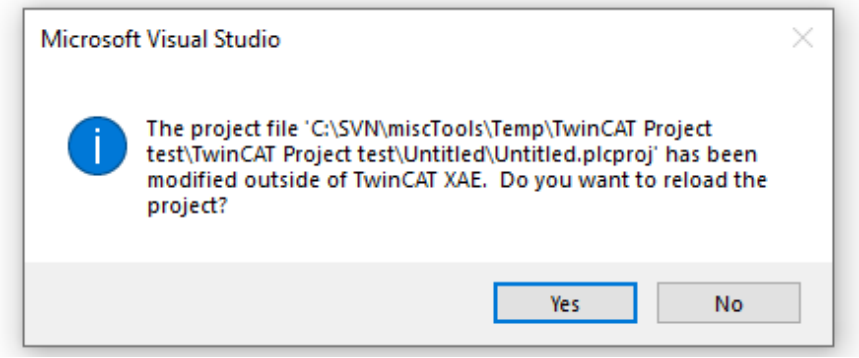


KUVA 24. Tallennus valmis -ikkuna

Lopuksi siitä muodostuu kuvan 24 mukainen, jos tallennus onnistui. Ilman ohjelman ajoa tässä olisi nähtävissä, mitkä tiedostoista ovat tallennettu ja millä versionumerolla. Tämä ei kuitenkaan päde kyseisessä tilanteessa, kun suoritetaan erillinen ohjelma, vaan näkyy vain polku, josta tallennus tehtiin. Syy tälle selitetään myöhemmin ohjelman toimintaa käsittelevässä vaiheessa.

Kuvan 24 ikkunasta tulee seuraavaksi painaa OK-painiketta. Tällöin tallennusprosessi päättyy lopullisesti ja tehdyt muutokset ovat siirtyneet arkistoon.

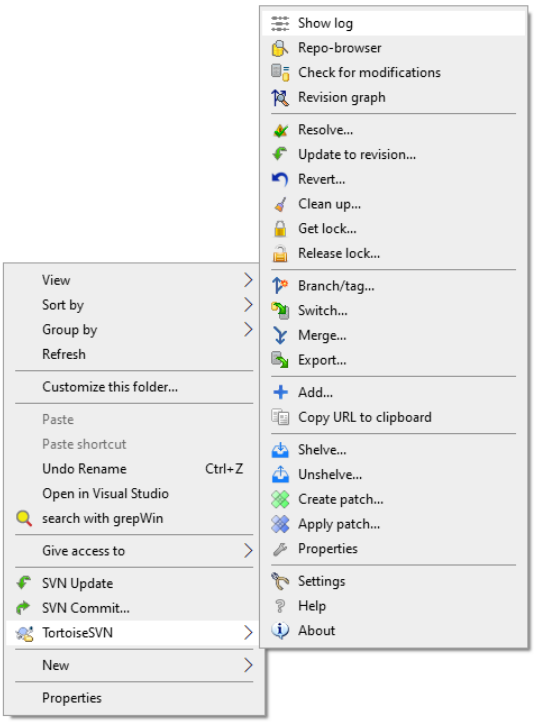
Mikäli käyttäjällä on kyseinen projekti TwinCAT 3:ssa auki, tulee kuvan 25 ilmoitus hetken kuluttua tallennuksen aloittamisesta. Tämä tulee hyväksyä (Yes).



KUVA 25. TwinCAT 3 ilmoittaa tiedostoihin ulkopuolelta tehdystä muutoksesta

Tällöin tehdyt muutokset kyseisestä plcproj-tiedostosta päivittyvät myös TwinCAT 3:n puolelle projektin tietoihin. Kuvan 25 ilmoitus kertoo käyttäjälle, että sen tiedostoihin on kohdistunut ulkopuolisia muutoksia.

Seuraavaksi kerrotaan miten käyttäjä voi tarkistaa sen, mitä on tallennettu ja millä versionumerolla. Tämä onnistuu painamalla hiiren oikella hakemistossa ja menemällä TortoiseSVN-painikkeen kautta alavalikkoon, josta valitaan näytä loki -painike (Show log) kuvan 26 esittämällä tavalla.

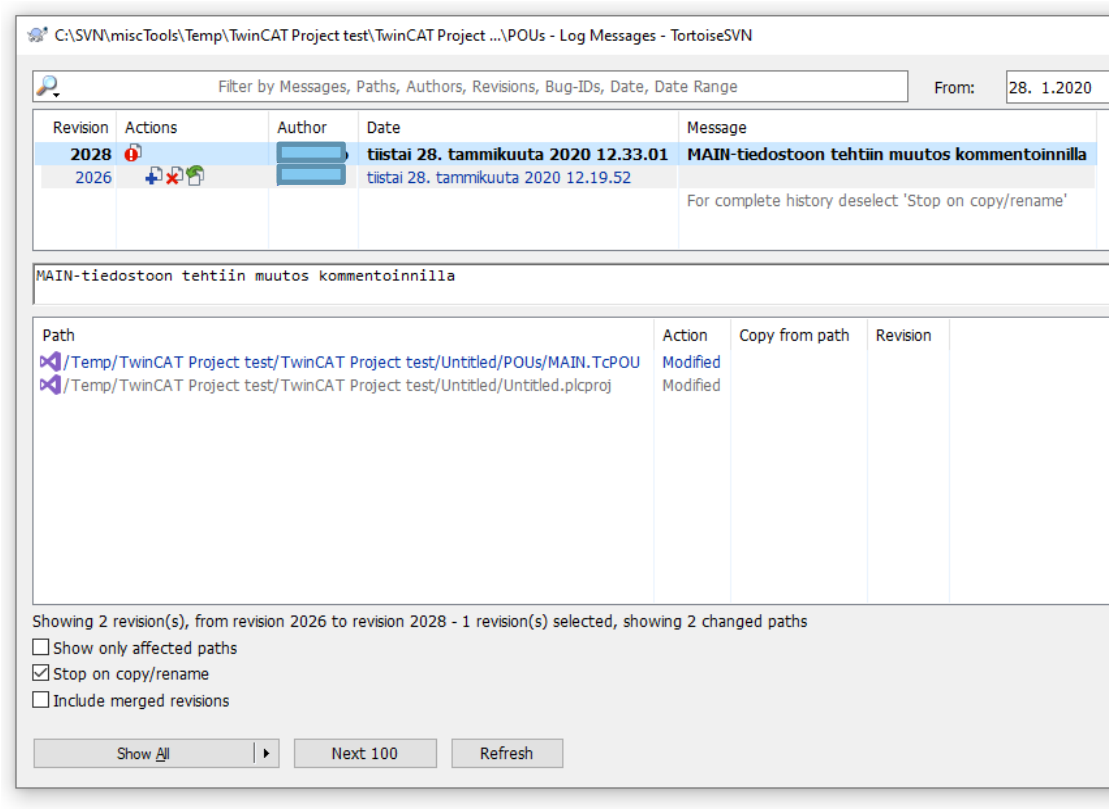


KUVA 26. Show log valikosta valittuna

Painettaessa kuvan 26 näytä loki -painiketta, avautuu kuvan 27 mukainen loki. Lokissa on mahdollista selata eri versioita, mitä tallennuksia ne sisältävät sekä niiden kommentteja. Kuvan 27 lokissa nähdään myös aikaisemmin tätä hakemistoa koskeva tallennus, jonka versionumero on 2026. Kyseessä ei ole siis tässä esimerkissä tehty tallennus, vaan sitä edeltävä, kuten versionumerosta huomataan. Toiminta (Actions) osiosta nähdään, mitä tallennuksessa on tapahtunut. Symbolit kuvan 27 tapauksessa versiota 2026 koskien oikealta vasemmalle ovat:

- Sininen plus-merkki, joka merkitsee sitä, että arkistoon ollaan lisätty tiedosto.
- Punainen X-merkki, joka merkitsee sisällön poistamista arkistosta.
- Vihreä kaareva nuoli, joka merkitsee jonkin korvaamista.

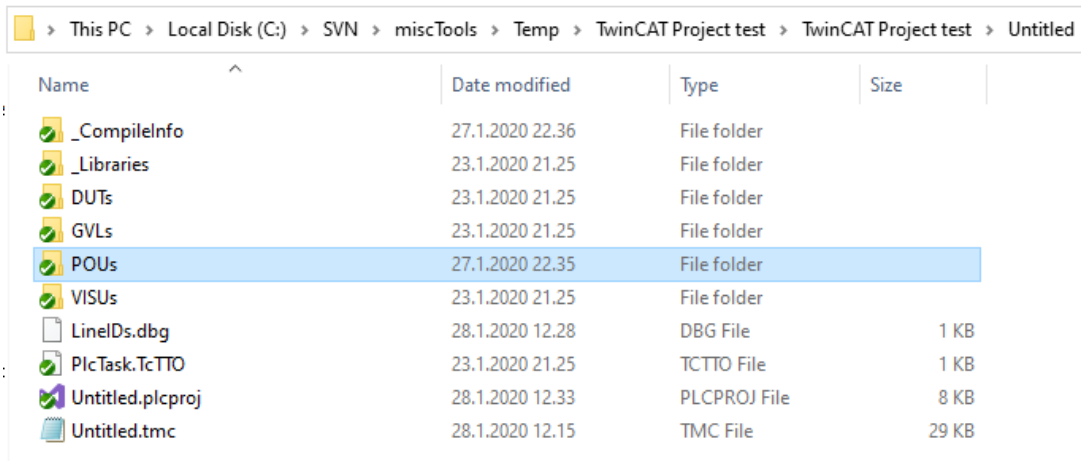
2026. versiota koskenut operaatio on liittynyt projektin uudelleen nimeämistä, joka vaatii edellä mainitut toiminnot. Subversion osaa kuitenkin tehdä nämä toiminnot itsenäisesti, kun käytetään uudelleen nimeämiseen (Rename) tarkoitettua toimintoa.



KUVA 27. Subversionin loki

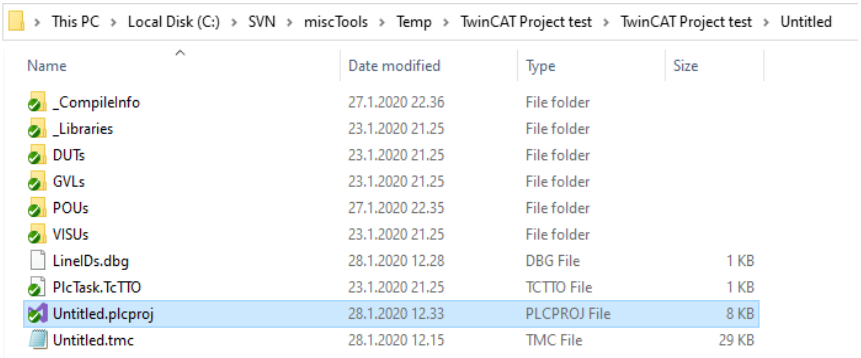
Äsken esitetty tallennus oli siis 2028 tallennus arkistoon, kuten kuvasta 27 nähdään. Tällöin se sai versionumeron (Revision) 2028. Toimintaosiosta (Actions) nähdään, että kyseiset tiedostot eivät ole olleet ajan tasalla symbolin perusteella. Tallennuksen tekijä (Author) on myös nähtävillä. Subversion hakee tähän automaattisesti käyttäjän tunnuksen tiedostaistaan. Ajankohtaosiosta (Date) on nähtävillä, milloin toiminto on tehty. Viestiosiosta (Message) nähdään nopeasti, laajempaa lokia selattaessa, mikä viesti tallennuksen yhteydessä ollaan jätetty. Viestin tarkempaa lukemista varten täytyy valita kyseinen versio, jolloin kuvassa 27 näkyvällä tavalla kommentti voidaan lukea laajemmasta tekstikentästä. Nämä viestit ovat siis käyttäjien tekemiä ja ne näkyvät kaikille arkiston lokia selaaville.

Polkuosiossa (Path) on taas nähtävillä, mitä tiedostoja kyseisellä tallennuksella on siirretty arkistoon sekä mitä toimintoja (Action) niiden yhteydessä on tehty. Verrattaessa kuvaa 27 kuvaan 23, jossa nähdään mitä ja minne Subversion on tallentamassa, havaitaan yhden polun tulleen lisää ja se on harmaana. Ensimmäisenä sinisellä näkyvä tiedosto on siis se, minkä tallennus äsken tehtiin. Harmaana näkyvän tiedoston tallentamista ei käyttäjän toimesta tehty. Syy tälle harmaana näkyvälle tiedostolle on se, että ennen tallennusta suoritettava ohjelma muokkasi kyseisen tiedoston sisältöä ja lisäsi sen tallennettaviin tiedostoihin. Itse harmaus johtuu siitä, että se sijaitsee polussa tallennuksen tekopaikkaa ylemmässä kansiossa. Esimerkkitapauksessa tallennuksen tekopaikka on kuvan 28 POUs-kansiossa.



KUVA 28. Kansio ylemmältä tasolta. Tallennettava kansio maalattuna

Harmaa tiedosto lokissa on kuvan 28 kansio tasolta, eikä POUs-kansiossa. Normaalisti Subversion ei kykenisi tekemään tallennuksia ylemmistä kansioista, mutta versionhallintaa varten tehty suoritettava ohjelma ajetaan ennen tallennusta kytkimen avulla. Tällöin suoritettava ohjelma hakee tämän tietyn kuvassa 29 maalatun .plcproj-tiedoston whitelist-tiedoston sisällön avulla ja muokkaa sen sisältöä sekä lisää sen tallennettaviin tiedostoihin.



KUVA 29. Projektin .plcproj-tiedosto maalattuna

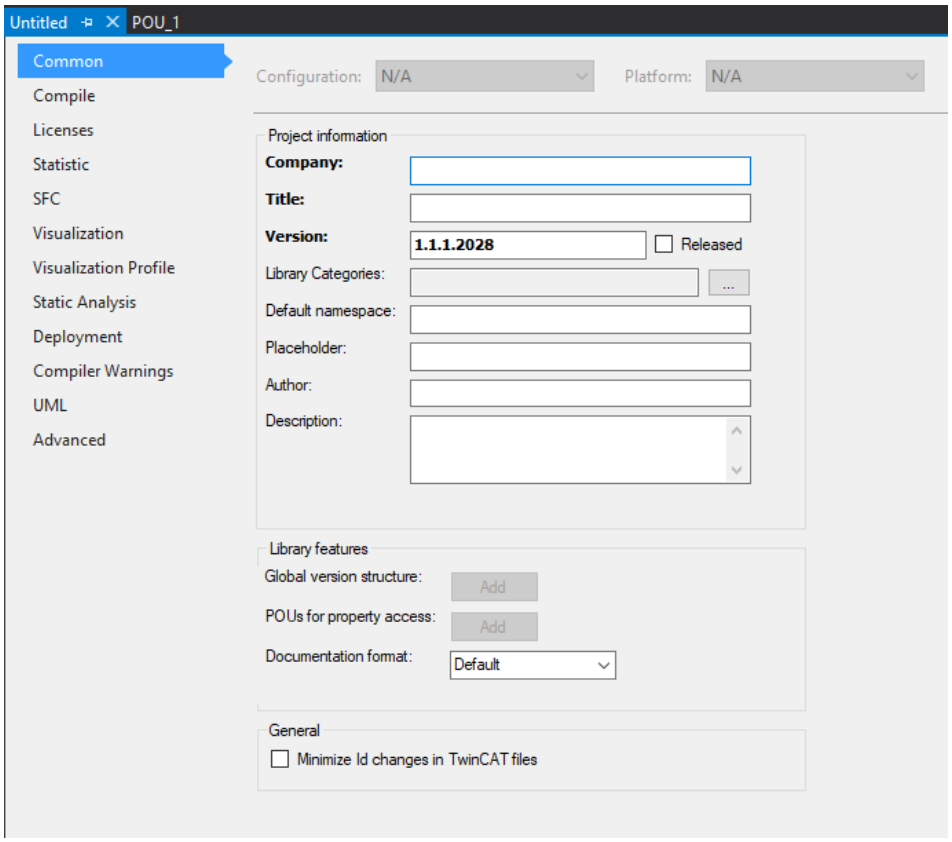
Tarkastellaan, mitä suoritettava ohjelma on tehnyt kuvan 29 .plcproj-tiedostossa. Kuvassa 30 nähdään otanta tiedoston sisällöstä. Tiedostoa voisi lukea millä tahansa tekstikäsittelyohjelmalla, mutta avataan se Visual Studion avulla, jolloin sen sisällön tarkastelu on helpompaa.

```
<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <FileVersion>1.0.0.0</FileVersion>
    <SchemaVersion>2.0</SchemaVersion>
    <ProjectGuid>{cea6f2a2-95d5-48cd-a1c3-6ae9948b9cdd}</ProjectGuid>
    <SubObjectsSortedByName>True</SubObjectsSortedByName>
    <DownloadApplicationInfo>true</DownloadApplicationInfo>
    <WriteProductVersion>true</WriteProductVersion>
    <GenerateTpy>false</GenerateTpy>
    <Name>Untitled</Name>
    <ProgramVersion>3.1.4023.0</ProgramVersion>
    <Application>{7b47f407-26b5-46b5-8530-992bca2e6984}</Application>
    <TypeSystem>{0e6147ff-8b4a-458c-84f9-478290dcf250}</TypeSystem>
    <Implicit_Task_Info>{092352b1-beb3-48b3-9c68-6b3ec2946ea3}</Implicit_Task_Info>
    <Implicit_KindOfTask>{03405f95-9fb0-406b-bf0f-b8639c1b7e54}</Implicit_KindOfTask>
    <Implicit_Jitter_Distribution>{7d957b26-9d2e-4355-8e2d-c46fa1fd1643}</Implicit_Jitter_Distribution>
    <LibraryReferences>{c71a8426-7545-4e70-82d2-59df2f81aac8}</LibraryReferences>
    <Released>false</Released>
    <ProjectVersion>1.1.1.2028</ProjectVersion>
    <DeploymentEvents>
      <Events>
        <Event>
          <CommandType>Execute</CommandType>
          <DeploymentType>PlcAfterCompile</DeploymentType>
          <Command1>"C:\Scripts\Boot revision script.exe"</Command1>
          <Command2 />
        </Event>
      </Events>
    </DeploymentEvents>
  </PropertyGroup>
```

KUVA 30. Ote .plcproj-tiedoston sisällöstä

Kuvan 30 sisällöstä havaitaan, kyseessä on XML-tiedosto. .plcproj-tiedosto on TwinCAT 3:n ohjelmoitavan logiikan projektitiedosto, joka sisältää projektikohtaista tietoa kootusti. Versionhallintaohjelma muokkaa tästä tiedostosta punaisella ympäröidystä projektin versiotiedosta viimeistä numeroa sarjasta. Käytännössä ohjelma tuo tähän Subversionin uuden tallennuksen versionumeron ja kuten tästä kuvasta nähdään, vastaa se aikaisemmin lokissa tarkasteltua versionumeroa. Koska tähän tiedostoon tehdään muutoksia, tulee se myös tallentaa arkistoon, jotta se on myös myöhemmin tarkasteltavissa.

Siirrytään TwinCAT 3:n puolelle. Täältä avataan projektin ominaisuudet, jolloin avautuu kuvan 31 mukainen ikkuna projektin yleisistä tiedosta, joihin voidaan asettaa haluttua tietoa projekti kohtaisesti.

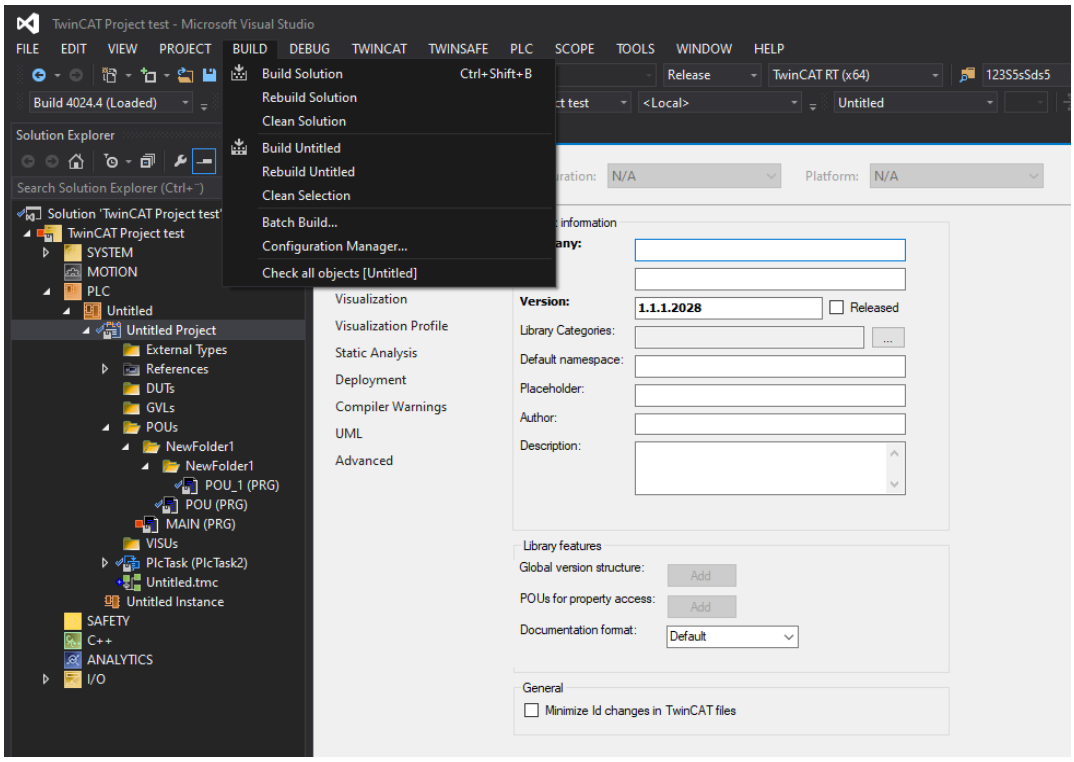


KUVA 31. TwinCAT 3 projektin yleiset tiedot

.plcproj-tiedoston versionumeron seurauksena uusi versionumero esiintyy myös kuvan 31 TwinCAT 3:n puolella olevissa tiedoissa. Tämä on versionhallinnan erillisen ohjelman tarkoitus, koska tämän perusteella ohjelmoitavien logiikoiden ohjelmistokehittäjät pystyvät seuraamaan, mitä versiota projektista he työstävät.

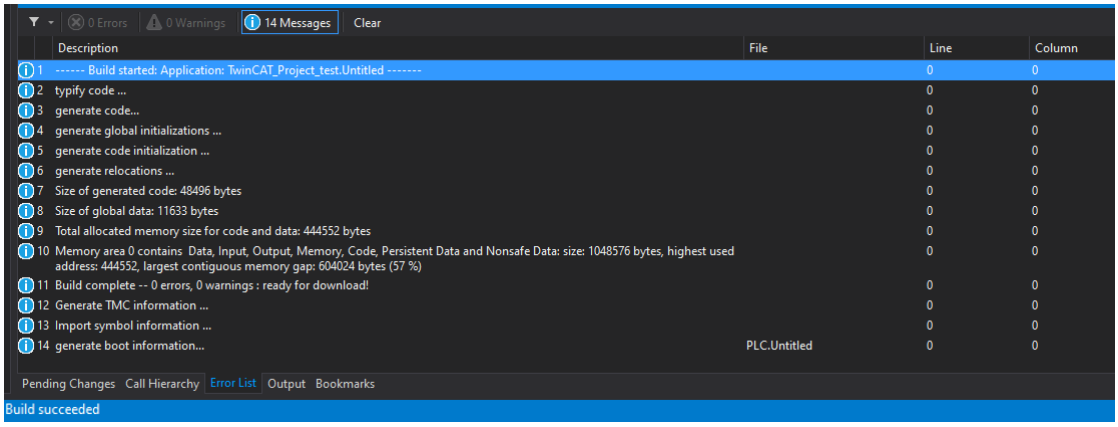
TwinCAT 3:n käännöksen yhteydessä skriptin käyttäminen

Ohjelman käyttö on todella yksinkertaista. Se suoritetaan automaattisesti, kun tehdään projektista tai koko TwinCAT 3 ratkaisusta (Solution) koontiversio (Build) tai uudelleenkoontiversio (Rebuild) kuvan 32 valikosta.



KUVA 32. Koontiversio-valikko TwinCAT 3:ssa

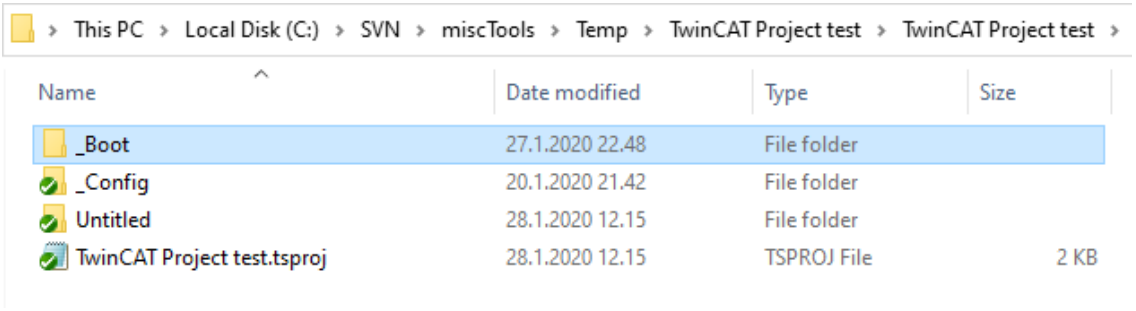
Valitaan esimerkiksi koko ratkaisun koontiversion tekeminen kuvassa 32 näkyvässä tilanteessa. Tällöin TwinCAT 3:ssa alareunaan tulee näkyviin mitä koonnin aikana tapahtuu kuvassa 33 näkyvällä tavalla.



KUVA 33. Koonnin tapahtumaviestit

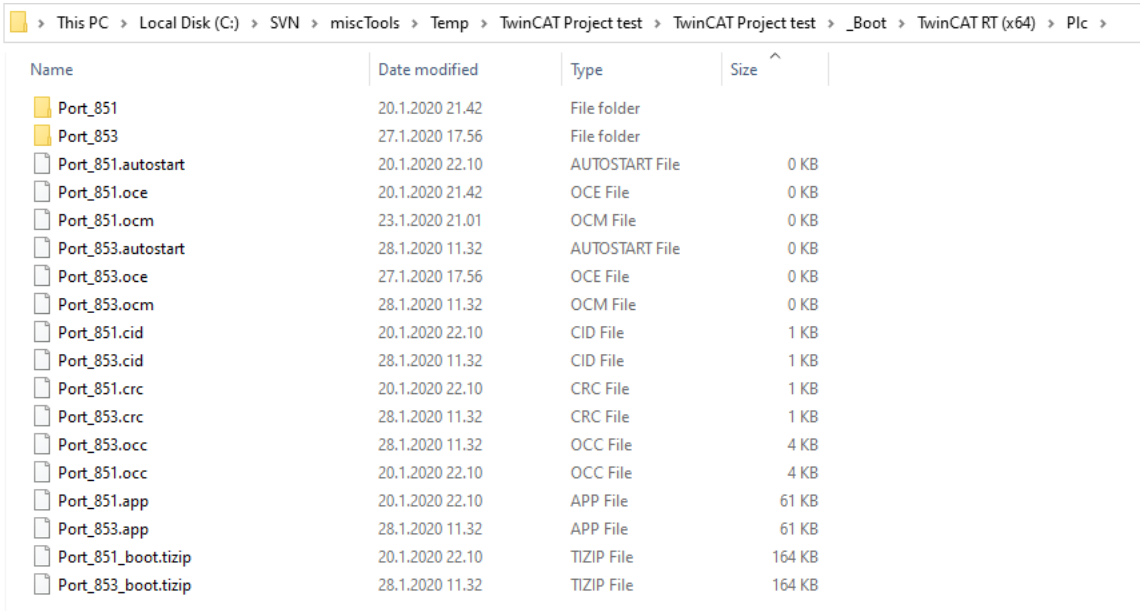
TwinCAT 3:ssa alhaalla on sininen alue kuvassa 33. Mikäli koontionnistuu, sinne tulee ilmoitus koonnin onnistumisesta (Build succeeded). Alaosasta näkee siis nopeasti, onko koontiversion tekeminen onnistunut vai ei.

Seuraavaksi tarkastetaan, mitä suoritettava ohjelma teki käännöksen yhteydessä. Kuvassa 34 näkyvä _Boot-kansio sisältää tarvittavan tiedon, jotta ohjelmoitava logiikka voi käyttökohteessa toimia itsenäisesti.



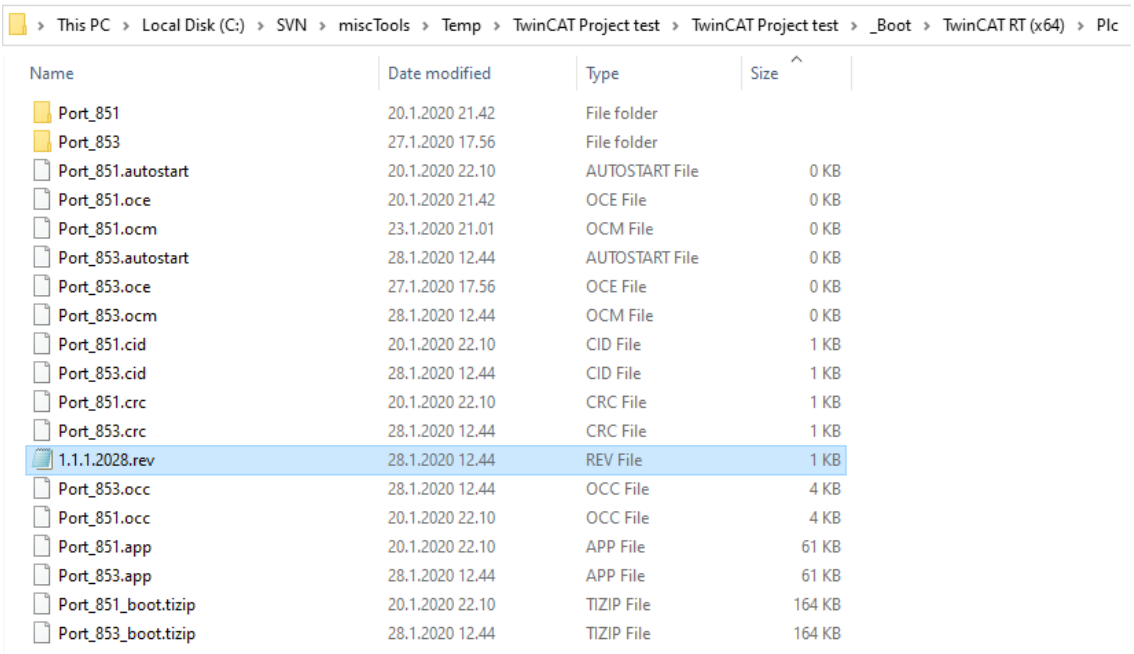
KUVA 34. Projektin alaisia kansioita. Käynnistystiedot kansio maalattu

Siirrytään _Boot-kansion alla olevaan kuvan 35 Plc-kansioon. Tällöin nähdään sen tilanne ennen kuin on suoritettu käännöksen yhteydessä toimivaa ohjelmaa.



KUVA 35. Plc-kansio ennen käännöstä

Kansio sisältää normaaleita tiedostoja, jotka tulevat sinne projektin käännöksen yhteydessä. Kuvassa 36 näkyy tilanne, kun käännös on tehty ja sen yhteydessä on käytetty suoritettavaa ohjelmaa.



KUVA 36. Plc-kansio käännöksen jälkeen, joka on suorittanut ohjelman

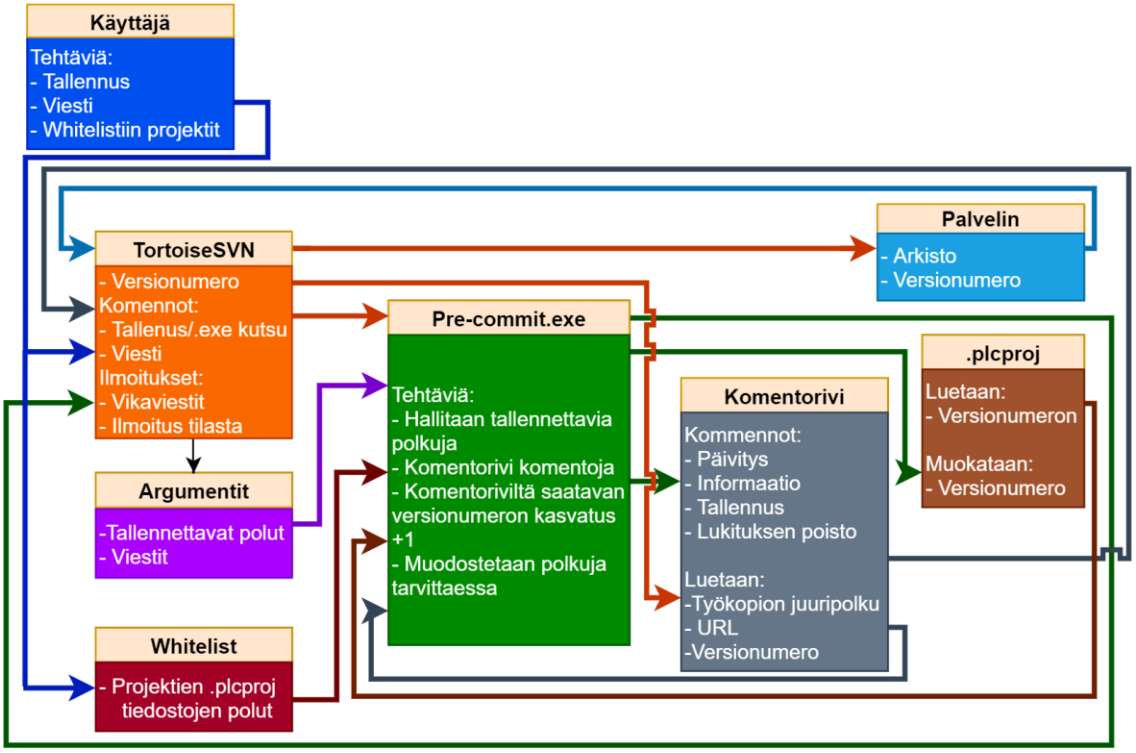
Kuvaan 36 on maalattu tiedosto, jonka nimi vastaa TwinCAT 3:n puolella olevaa versionumeroa. Tämän tarkoituksena on helpottaa asiakkaalla olevan version seuraamista, jotta tuotetuki ja päivitys tilanteissa saadaan nopeasti selville, mikä versio ohjelmoitavan logiikan ohjelmasta on asiakkaan koneella käytössä.

Mikäli olisi aikaisemmin kirjoitettu TwinCAT 3 projektitietoihin jokin kuvaus (Description), niin se ilmenisi maalatun tiedoston versionumeron jälkeen muodossa ”_kuvaus”. Skripti lisää siis myös kaiken tekstin kuvauskentästä versionumeron nimeä kantavan tiedoston nimeen.

5.3 Versionhallinnan skripti TortoiseSVN:ään

Suoritettava ohjelma on käytännössä suoritettava tiedosto nimeltään Pre-commit.exe. Ohjelma on toteutettu olio-ohjelmointia hyödyntäen C#-ohjelmointikielellä, joka on tilaajayrityksellä laajasti käytössä. Tämän avulla on pyritty välttämään ohjelman kehityksen henkilöityminen pitkällä aikavälillä, joka helpottaa ylläpitoa.

Toimenpiteiden suorittamiseksi useissa eri tilanteissa, tarvitsee ohjelman muodostaa tarpeellisia polkuja tallennettaville tiedostoille sekä tiedostaa, mitkä ovat ne projektit, joiden yhteydessä halutaan ohjelma suorittaa. Tämän lisäksi sen täytyy lukea versionumero palvelimelta, jotta voidaan ennustaa seuraava versionumero, joka lisätään projektin .plcproj-tiedostoon. Kun tiedostoa muokataan, tulee se vielä lisätä tallennettavien tiedostojen joukkoon automaattisesti, jotta se saataisiin siirrettyä palvelimella olevaan arkistoon. Kuvassa 37 esitetään, kuinka ohjelma toimii muun järjestelmän kanssa vuorovaikutuksessa.



KUVA 37. Pre-commit suoritettavan ohjelman vuorovaikutus

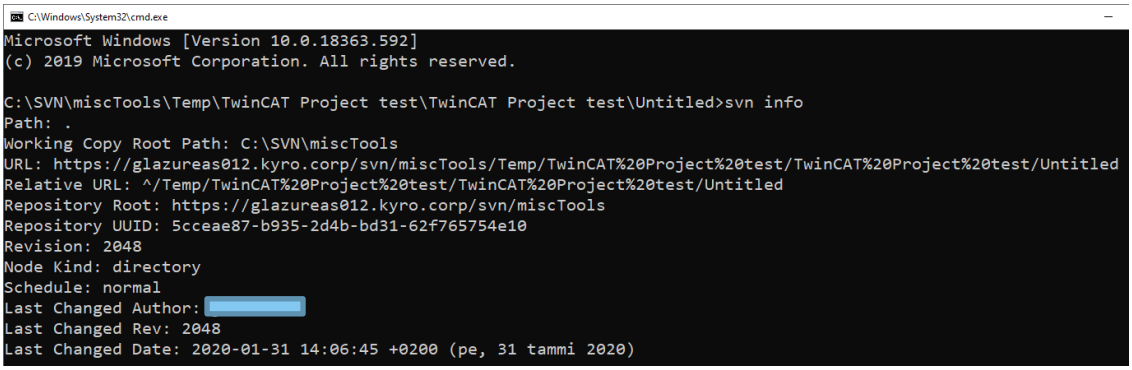
Kuvasta 37 havaitaan käyttäjällä olevan kaksi mahdollisuutta vaikuttaa suoritukseen. Otetaan käsittelyyn ensimmäisenä Whitelist-tiedosto, joka sisältää suhteelliset URL-polut arkistosta projektien .plcproj-tiedostoihin. Nämä URL-polut ovat siis lista projekteista, joiden yhteydessä halutaan ohjelman toimivan. Suhteellisia URL-polkuja tahdotaan käyttää tässä tapauksessa, sillä näin voidaan hyödyntää listaa sallituista projekteista useamman käyttäjän koneilla, koska se vastaa palvelimella olevan arkiston polkua. Tällöin käyttäjän paikallinen hakemistorakenne tai se, että onko arkistosta vain tietty osa käytössä, ei haittaa toimintaa vaan aina löydetään oikeaan kohteeseen. Koska tiedoston nimi on suhteellisen yleisesti käytössä eri ohjelmien yhteydessä, toiminnan varmistamiseksi

halutaan, että se sijaitsee samassa kansiossa kuin Pre-commit.exe-tiedosto. Tämän paikallistamiseen käytetään suoritettavassa ohjelmassa toimintoa, joka hakee ohjelman sijainnin hakemistosta ja täältä etsitään whitelist.wl-tiedosto. Ohjelma käy tiedoston sisällön läpi ja lukee polut. Myöhemmin se valitsee arkistoon tallennettavien tiedostojen avulla halutun polun. Jos whitelist-tiedosto ei sisällä mitään tietoa tai sitä ei ole olemassa, ohjelma ei voi suorittaa toimintoja.

Toinen mistä käyttäjä voi vaikuttaa toimintaan on TortoiseSVN:än käyttäjärajapinta. TortoiseSVN:ssä tulee olla tehtynä aikaisemmin käsitellyt käyttöönotto-asettelut, jotta toiminto voidaan suorittaa. Käyttö tapahtuu samalla tavalla kuin käytönopastus vaiheessa on kerrottu eli tallennuksen yhteydessä kutsutaan suoritettavaa ohjelmaa. Kun tallennusprosessi aloitetaan, luo TortoiseSVN automaattisesti neljä väliaikaista tiedostoa, jotka ovat nähtävillä %TEMP%-kansiosta paikallisesti. Näiden polut saadaan TortoiseSVN:ltä argumenttien avulla. Ohjelman kannalta tarpeellisia ovat 0- ja 2-argumentit. 0-argumentti sisältää tallennettavat tiedostot ja 2-argumentti taas viestin, jonka käyttäjä on kirjoittanut tallennuksen yhteydessä. TortoiseSVN-käyttäjärajapintaan tulevat myös vikaviestit näkyviin, mikäli ohjelman suorituksessa tai muissa tallennukseen liittyvissä asioissa on ongelmia. Lisäksi se ilmoittaa tallennuksen onnistumisesta, kun tallennus on tehty.

TortoiseSVN on ainoa yhteys palvelimelle, jossa arkisto sijaitsee. Arkiston tietoa saadaan tuotua ohjelmalle käyttämällä komentoriviä suoritettavalla ohjelmalla. Ohjelma avaa komentorivin hakemistossa, jossa .plcproj-tiedosto sijaitsee. Komentorivin avulla hankitaan tietoa siitä, mikä on kyseisestä arkistoa palvelimella vastaava uusin versionumero, missä URL-polussa kansio sijaitsee palvelimella ja mikä on juuripolku paikallisen käyttäjän koneella.

Ohjelma käyttää tekstistä etsimiseen säännöllisiä lausekkeita (Regular expressions). Tämän avulla voidaan siis etsiä tiettyjä merkkijonoja (String), kun tiedetään niiden tyypillinen esitystapa. Esimerkiksi tässä ohjelmassa säännöllisillä lausekkeilla parsitaan haluttuja osuuksia komentorivin tekstistä. Kuvassa 38 nähdään, kuinka asiat esitetään komentorivillä.



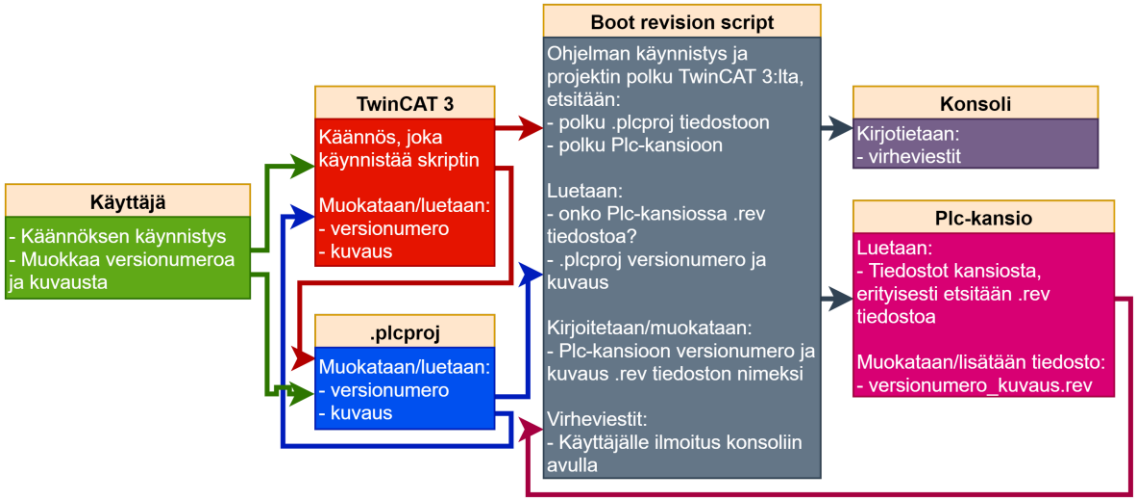
KUVA 38. Komentorivin tuloste svn info -komennolla

Ohjelma suorittaa komentorivin avulla lisäksi päivityksen, lukituksen poisto sekä tallennuskomentoja. Näiden avulla saadaan siirrettyä halutut tallennettavat tiedostot palvelimen arkistoon.

Suoritettavan ohjelman rooli on yhdistellä edellä mainittuja asioita toisiinsa toimivaksi kokonaisuudeksi, jolloin se tarkoittaa tarpeen mukaan tiettyjen polkujen muodostamista saadun tiedon avulla. Se hankkii ajankohtaisen versionumeron ja kasvattaa tätä yhdellä numerolla, joka taas injektoidaan .plcproj-tiedostossa projektin versionumero-osuuden viimeiseen osioon. Projektin versionumero osuus asiakastapauksessa koostuu käytännössä ”X.X.X.X”-mallisesti, jossa X-kirjaimet esittävät mitä tahansa numeroita. Ohjelma korvaa siis viimeisen X-kirjaimen uudella versionumerolla. Kun tämä korvaus on suoritettu, suoritetaan tallennus, joka sisältää pakotetusti .plcproj-tiedoston, sillä myös sen tahdotaan siirtyvän palvelimen arkistoon. Samalla tallennetaan myös aikaisemmin 0-argumentistä saatu lista tallennettavista tiedostoista. Käytännössä tallennus suoritetaan siis ennen kuin TortoiseSVN asiakaspuolen ohjelma olisi sitä normaalisti suorittamassa. Tämän takia käyttäjälle ei ilmesty tavallista syötettä tallennuksen onnistuneen suorittamisen jälkeen, vaan ilmoittaa ainoastaan onnistuneesta tallennuksesta.

5.4 PLC käännöksen skripti

PLC-ohjelman käännöksen yhteydessä suoritettava ohjelma on toteutettu C#-ohjelmointikielellä samaan tapaan kuin versionhallinta skripti TortoiseSVN:ään. TwinCAT 3:n PLC-ohjelman käännöstä tehtäessä kutsutaan kyseistä suoritettavaa ohjelmaa aikaisemmin esiteltyjen ennakkoasetuksien perusteella. Tämän yhteydessä saadaan myös selville, mikä TwinCAT 3:n projekteista kutsuu käännöstä ja mikä on polku projektiin. Tällöin voidaan etsiä kyseisen projektin alaisesta kansiorakenteesta halutut kansiot sekä tiedostot. Nämä ovat tässä tapauksessa plcproj-tiedosto sekä Plc-kansio. Kuva 39 esittää, kuinka ohjelma toimii vuorovaikutuksessa muun järjestelmän kanssa.



KUVA 39. Boot revion scriptin vuorovaikutus

Kuvasta 39 nähdään, että käyttäjä voi toiminnallansa vaikuttaa kahteen asiaan suoritettavan ohjelman ympäristössä. .plcproj-tiedosto sisältää kuvauksen ja versionumeron, joita on mahdollista joko muokata täällä tai TwinCAT 3:n puolella. Käyttäjän voi lisätä kuvauksen näiden kautta, mikäli sitä ei ole aikaisemmin projektille annettu. Versionumeroakin voi myös muokata halutessaan. Tämän ohjelman käyttö on kuitenkin suunniteltu käytettäväksi ennen tallennusta suoritettavan ohjelman kanssa, jolloin palvelimelta saadaan tuotua versionumero ja liitettyä se mukaan.

Ohjelma hankkii siis .plcproj-tiedostosta nämä halutut tiedot, etsii Plc-kansion projektin _Boot-kansion alta ja selvittää onko täällä olemassa jo tiedostoa .rev-päätteellä. Jos ohjelmaa ei ole aikaisemmin ajettu, se luo sinne tiedoston, joka

sisältää versionumeron sekä projektin kuvauksen, jonka tiedosto pääte on .rev. Toisella suorituskerralla se vain muokkaa tiedoston nimeä halutuksi. Syntaksi, missä muodossa tiedosto ilmenee, on ”versionumero_kuvaus.rev” eli esimerkiksi ”1.1.10.2304_päivitys.rev”.

Ohjelma sisältää myös vikaviestejä, jotka antavat käyttäjälle todennäköisimmissä virhetapauksissa ilmoitukset, mistä vika todennäköisesti johtuu ja mitä tulisi tehdä, jotta voidaan suorittaa kyseinen toiminto onnistuneesti. Lisäksi jos ohjelma kaatuisi jostakin syystä, se palauttaa virheviestit ohjelmasta. Nämä molemmat tulevat käyttäjän koneella näkyviin konsolille, joiden ohjeita seuraamalla ongelma voidaan ratkaista joko itsenäisesti tai haastavissa vikatilanteissa ohjelman ylläpitäjä kykenee selvittämään ongelman näiden avulla.

6 POHDINTA

Työ vaati perehtymistä Glastonin aikaisempiin käytänteisiin, jotta saatiin selville, mikä on tarpeellista tietoa ja missä sen tulee olla saatavilla. Ensimmäisenä oli käytännössä selvitettävä, onko TwinCAT 3:n yhteydessä mahdollista toteuttaa Glastonin tarvitsemia versionhallinnallisia ominaisuuksia suunnittelun ja asiakaan kanssa toimimisen tarpeisiin sekä kuinka helpoksi se olisi mahdollista tehdä verrattuna aikaisempiin käytänteisiin. Nämä saatiin kuitenkin selvitettyä opinnäytetyöprosessin yhteydessä ja näiden avulla pienellä lisäpanoksella aikaisempaan työskentelyyn verrattuna, suunnittelussa onnistutaan toteuttamaan versionhallinnan kannalta tarpeelliset asiat automaattisesti TwinCAT 3:a käytettäessä.

Koska tarkoituksena oli myös tarkastella versionhallintoja yleisesti, oli syytä vertailla keskitettyä sekä hajautettua versionhallintajärjestelmää, jotta voidaan jatkossa tulevien tarpeiden mukaisesti miettiä, olisiko esimerkiksi Git versionhallintajärjestelmä hyvä vaihtoehto TortoiseSVN versionhallintajärjestelmälle. Tämä ei kuitenkaan tässä kohtaa ollut vielä ajankohtainen asia, mutta alustavaa selvitystä asian suhteen tehtiin, mikäli tähän suuntaan olisi joskus tarpeita siirtyä. Voidaan kuitenkin todeta, että verkkoyhteydet ovat riittävän hyvät tällä hetkellä ympärimaailmaa, ettei Gitin tarjoamille ominaisuuksille ole Glastonilla tällä hetkellä tarvetta.

Haasteena työssä oli luoda jokaisen käyttäjän koneella toimivat suoritettavat ohjelmat, jotka eivät ole suoraan sidoksissa käyttäjällä oleviin polkurakenteisiin, vaan osaavat toimia oikealla tavalla palvelimelta saamansa tiedon mukaan. Tämä estää käytännössä kaikkien polkujen kovan ohjelmoinnin ja ohjelmien tulee osata suunnistaa muualta saatavien tietojen avulla. Lisäksi TortoiseSVN versionhallintajärjestelmiin liittyvän uudemman tiedon ja varsinkin monimutkaisiin toimintoihin pyrkittäessä oli haastavaa löytää tietoa.

Näiden opinnäytetyössä tehtyjen suoritettavien ohjelmien testaus muiden kuin kehittäjän toimesta, on ollut vielä kohtalaisen suppeata. Tämä johtuu siitä, että Glaston ei ole vielä siirtynyt käyttämään Beckhoff TwinCAT 3:a, jolloin testaus on lähinnä ollut kehittäjän vastuulla. Ohjelmasta on hiottu pieniä vaivoja pois näiden

testien myötä koko kehityskaaren aikana ja tällä hetkellä on tilanne, ettei ainakaan käyttäjä, joka tietää mitä tekee, saa toiminnallansa siihen ilmenemään virheitä. Ohjelma on myös kommentoitu erittäin laajasti, joka mahdollistaa ja helpottaa virheiden korjausta sekä ohjelman muokkaamista, vaikka alkuperäinen kehittäjä ei olisikaan enää yrityksen palveluksessa.

Lisäksi näkisin TwinCAT 3:n tai minkä tahansa muun Visual Studioon pohjautuvan järjestelmän käytön yhteydessä kannattavaksi käyttää integrointiohjelmia, esimerkiksi VisualSVN tai AnkhSVN, mikäli käyttää versionhallintaa. Tämä nopeuttaa ja selkeyttää työskentelyä huomattavasti, sillä ei tarvitse jatkuvasti olla hakemistoympäristössä tekemässä tallennuksia, vaan ne onnistuvat nopeasti sovelluksen kautta.

LÄHTEET

Apache Subversion. Apache Subversion Features. Luettu 12.1.2020. https://subversion.apache.org/features.html

Atlassian Bitbucket n.d. Versionhallintaohjelmisto ammattilaistiimeille. Luettu 8.1.2020. https://bitbucket.org/product/fi/version-control-software

Beckhoff 2012. TwinCAT3 – eXtended Automation (XA). Luettu 14.1.2020. https://download.beckhoff.com/download/Document/catalog/Beckhoff_TwinCAT3_042012_e.pdf

Beckhoff n.d. TwinCAT 3 Source Control, Project Files. Luettu 5.2.2020. https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_sourcecontrol/18014398915785483.html&id=

Collins-Sussman, B., Filzpatrick, B. & Pilato, C. 2011. Version Control with Subversion For Subversion 1.7.: (Compiled from r6028). Kustannus O'Reilly Media. http://svnbook.red-bean.com/en/1.7/svn-book.pdf

Ernst, M. 2012. Version control concepts and best practices. Computer Science & Engineering, University of Washington. Luettu 8.1.2020. https://homes.cs.washington.edu/~mernst/advice/version-control.html#Introduction_to_version_control

Glaston n.d. INVESTORS, Luettu 16.1.2020 https://glaston.net/glaston-as-an-investment/

Glaston n.d. OFFERING, Machines. Luettu 16.1.2020. https://glaston.net/offering/#Machines

Lubański, M. 2019. Centralized vs Distributed Version Control System. Luettu 16.1.2020. https://medium.com/faun/centralized-vs-distributed-version-control-systems-a135091299f0

SFS-EN 61131-3. 2013. Ohjelmoitavat logiikat – Ohjelmointi kielet. Helsinki: Suomen Standardisoimisliitto SFS. Luettu 8.1.2020. Vaatii käyttöoikeudet. https://online-sfs-fi.libproxy.tuni.fi/