Tampereen ammattikorkeakoulu

Prosessiohjaimien kahdennus-testauksen automatisointi

Santeri Mäkelä

OPINNÄYTETYÖ
Huhtikuu 2020

Sähkö- ja automaatiotekniikan koulutus
Automaatiotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Sähkö- ja automaatiotekniikan koulutus
Automaatiotekniikka

MÄKELÄ, SANTERI:
Prosessiohjaimien kahdennustestauksen automatisointi

Opinnäytetyö 36 sivua, joista liitteitä 0 sivua
Huhtikuu 2020

Opinnäytetyö käsittelee Valmet DNA -automaatiojärjestelmän prosessiohjaimien kahdennustestauksen automatisointia. Automaatiojärjestelmiä integroivalla ja testaavalla osastolla oli tarve vähentää käsin tehtävän testauksen määrää järjestelmätestauksessa ja näin nopeuttaa järjestelmän testaamista. Tarkoituksena oli automatisoida prosessiohjaimien kahdennustestauksen osuus järjestelmätestaukseen sisältyvästä kokonaisuudesta ja luoda automaattiselle testaukselle käyttöympäristö. Käyttöympäristöön voidaan myöhemmin lisätä testitapauksia kattamaan koko järjestelmätestaus. Automaattisen kahdennustestauksen luomisessa hyödynnettiin järjestelmätestaukseen tarkoitettua testausohjetta. Lisäksi automaattisen testauksen käyttöympäristön valintaa ja suunnittelua varten haastateltiin järjestelmän testaajia sekä tuotekehitysosaston testaajia.

Työn tuloksena syntyi automaattinen prosessiohjaimien kahdennustestaus vikatilanteeseen, jossa tehonsyöttö pääprosessiohjaimelta katkeaa. Tämä on yksi neljästä prosessiohjaimien kahdennustestauksen osuudesta järjestelmätestausohjeessa. Työn aikana keskityttiin ennen kaikkea automaattitestauksen käyttöympäristön luomiseen, johon järjestelmän testaajat voivat myöhemmin lisätä kolme jäljellä olevaa osuutta kahdennustestauksesta. Kahdennustestauksen neljästä osuudesta automatisointiin yksi osuus, koska kaikkien osuuksien automatisointi olisi ollut liian laaja kokonaisuus käsiteltäväksi opinnäytetyössä.

Tavoitteena oli vähentää käsin tehtävän työn määrää järjestelmätestauksessa. Työssä luotu kahdennustestaus tuo järjestelmätestaukseen automaattisuutta eli vähentää käsin tehtävää testausta. Työn varsinaiset tulokset ja hyödyt tulevat esiin vasta, kun kaikki kahdennustestauksen osuudet on automatisoitu ja testausta hyödynnetään Valmetin automaatioprojektien järjestelmä- ja tehdastestauksessa. Työn aikana suoritetun itsetestauksen perusteella voidaan kuitenkin todeta, että automatisoitu kahdennustestaus nopeuttaa testaukseen kuluvaa aikaa ja vapauttaa testaajien työaikaa muihin työtehtäviin.

Asiasanat: prosessiohjain, kahdennus, testaus, automatisointi

ABSTRACT

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

MÄKELÄ, SANTERI:
Automated Testing of Redundant Process Controllers

Bachelor's thesis 36 pages, appendices 0 pages
April 2020

This thesis covers the automation of testing a redundant process controller in the Valmet DNA automation system. The objective of this study was to reduce the amount of manual testing and thus reduce the time used in system testing by creating an automated redundancy test. The redundancy test is a part of system testing. The purpose was also to create an operating environment for automated tests, to which tests can later be added to cover the entire system testing. This study was carried out by utilizing the instruction on system testing. Interviews of the employees who test and develop the system were carried out as well for choosing the operating environment for automated tests.

As a result, an automated redundancy test for process controllers was made. It contains a test for the event of a fault in which the power supply to the main process controller is interrupted. This is one of the four process controller redundancy testing sections in the system testing instruction. During the study, the focus was on creating the operating environment for automated tests. People who test the system can later add the three remaining parts of the redundancy test to the automated testing environment. The reason for this was that the automation of all four sections would have been too extensive to be implemented during the thesis.

The objective of this thesis was to reduce the amount of manual testing in system testing. The process controller redundancy test created in this study makes the system testing procedure more automated, so the objective of the study was achieved. The actual results and benefits of the study will appear after all four sections of redundancy test have been automated and testing automation can be utilized in system testing and Factory Acceptance Tests of Valmet’s automation projects.

Key words: process controller, redundancy, testing, automation

SISÄLLYS

LYHENTEET JA TERMIT

ACN	Valmetin sovellus- ja ohjausasema, (Application and Control Node)
Automaatioverkko	Ethernet-verkko, joka yhdistää automaatiojärjestelmän laitteet toisiinsa
Avainsana	Menetelmä, jolla skriptiä voidaan käyttää sitä vastaavaa sanaa käyttäen
DCS	Hajautettu automaatiojärjestelmä, (Distributed Control System)
Debugger-ohjelma	Valmetin järjestelmän ja sovellusten testaamiseen ja virheenkorjaamiseen tarkoitettu työkalu
DNA	Valmetin automaatiojärjestelmä, (Dynamic Network of Applications)
Ethernet-kortti	Fyysinen komponentti, joka yhdistää tietokoneen tietoliikenneverkkoon
FAT	Tehdastestaus, (Factory Acceptance Test)
IBC	I/O-väylän ohjain, (I/O Bus Controller)
IPSP	ACN-laitteiden teholähde, (Integrated Power Supply)
I/O	Tulo ja lähtö, (input/output)
I/O-järjestelmä	Yhdistää automaatiojärjestelmän toimilaitteet ja anturit ohjaukseen
I/O-kehikko	Alusta johon I/O-kortit kiinnitetään
I/O-kortti	Fyysinen komponentti, joka yhdistää tulot ja lähdöt automaatiojärjestelmään
Järjestelmätestaus	Automaatiojärjestelmän laitteiston ja ohjelmistojen testaus
Jenkins	Ohjelmistotuotannon automaatiopalvelin
Metatieto	Tieto, joka tuodaan testikokonaisuuden ulkopuolisesta tietosisällöstä
Nokia Networks	Nokia Oyj:n liiketoimintayksikkö, joka suunnittelee ja valmistaa tietoliikenneverkoissa käytettäviä laitteita ja ohjelmistoja
Testitapaus	Avainsanoista muodostuva ohjelmakoodi

Testikokonaisuus	Testitapauksista muodostuva kokonaisuus
Toimituskeskus	Valmetin osasto, joka suunnittelee, integroi, alustaa ja testaa automaatiojärjestelmän
PCS	Valmetin prosessiohjain, (Process Control Server)
Python	Korkean tason ohjelmointikieli
RCOK-kaapeli	Kahdennuksen tilanteen seurantaan tarkoitettu kaapeli, (Redundant Condition OK)
Robot Framework	Hyväksymistestaukseen tarkoitettu testausautomaatiokehys
SFS-EN 62381	Suomen Standardoimisliiton julkaisema englanninkielinen standardi (Automation systems in the process industry)
Skripti	Ohjelmallinen komentosarja toimintojen automatisointiin

1 JOHDANTO

Automaatiojärjestelmän testauksen automatisointiin panostetaan, jotta testausta saadaan tehostettua suoritusajan, virheiden havaitsemisen ja tulosten tasalaatuisuuden suhteen. Testaamisen automatisointi soveltuu yleensä parhaiten useasti toistuvien testien suorittamiseen. Automatisoinnilla ei kuitenkaan voida korvata manuaalista testausta kokonaan, vaan tarkoituksena on vapauttaa testaajien työaikaa vaativuudeltaan korkeamman tason manuaaliseen testaukseen.

Aihe opinnäytetyölle syntyi työn tilaajan tarpeesta automatisoida järjestelmätestausta. Testattavalle järjestelmälle suoritetaan useasti toistuvia testejä, joten sen automatisointi olisi hyödyllistä. Järjestelmätestauksessa varmistetaan muun muassa automaatiojärjestelmän prosessiohjainten ja palvelintietokoneiden konfiguroinnin, ohjelmaversioiden ja lisenssien oikeellisuus sekä järjestelmän laitteiden välinen kommunikointi ja toiminta vikatilanteissa.

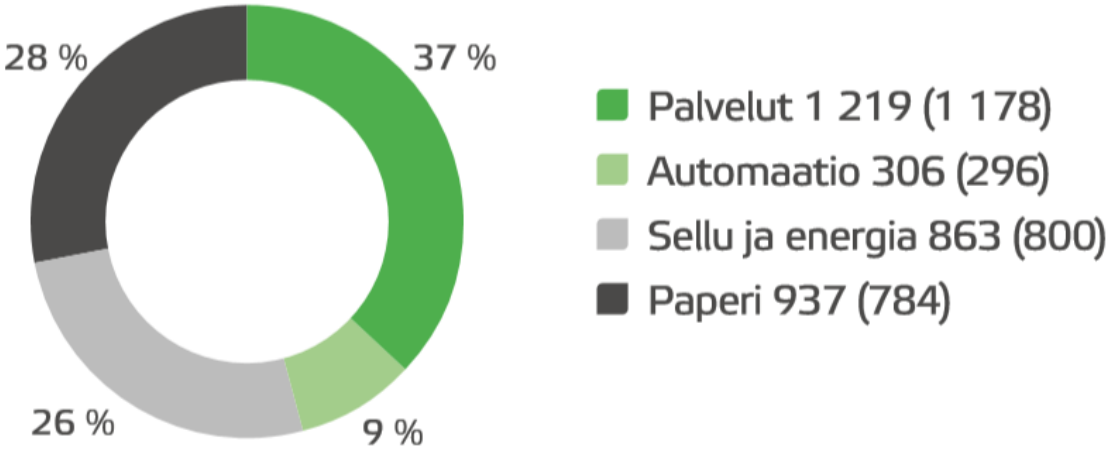
Opinnäytetyön tarkoituksena on automatisoida Robot Framework -testausautomaatiokehyksellä prosessiohjaimien kahdennustestaus, joka on yksi osa järjestelmätestausta. Työ rajattiin sisältämään kahdennustestaus vikatilanteessa, jossa tehonsyöttö pääprosessiohjaimelta katkeaa. Tässä tilanteessa tapahtuu prosessiohjaimien vaihto, jolloin varalla oleva prosessiohjain siirtyy ohjaamaan prosessia ilman prosessin pysähtymistä.

Työn tuloksena syntyvä automaattinen kahdennustestaus nopeuttaa järjestelmätestausta ja vähentää käsin tehtävää testaustyötä. Lisäksi työn yhteydessä syntyy alusta testauksen automatisointiin, johon voidaan lisätä uusia testejä kattamaan koko järjestelmätestaus. Automaattista kahdennustestausta voidaan myös hyödyntää tehdastestaukseen, johon sisältyy prosessiohjaimien kahdennuksen testaus.

.

2 VALMET AUTOMATION OY

Opinnäytetyön toimeksiantaja Valmet Automation Oy toimii osana Valmet-konsernia, joka on maailman johtava prosessiteknologian, automaatioratkaisujen ja palvelujen toimittaja ja kehittäjä sellu-, paperi- ja energiateollisuudelle. Valmet työllistää yli 13 000 ihmistä ympäri maailmaa ja konsernin liikevaihto vuonna 2019 oli 3,5 miljardia euroa. Valmet on jaettu neljään liiketoimintalinjaan: palvelut, sellu ja energia, automaatio sekä paperit. Automaatio-liiketoimintalinjan osuus vuoden 2019 liikevaihdosta oli 10 prosenttia. Kuviossa 1 on esitetty Valmet Oyj:n liikevaihto toimilinjoittain prosentteina sekä miljoonina euroina. (Valmet vuosikatsaus 2020, 6-8.)

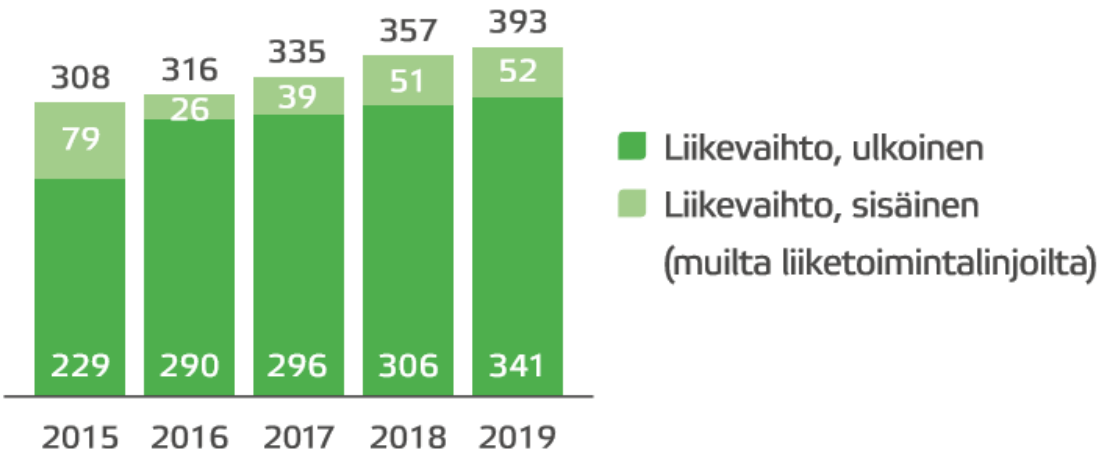


KUVIO 1. Valmet Oyj:n toimintalinjojen liikevaihdot vuonna 2019 (Valmet vuosikatsaus 2020, 7)

Valmetilla on pitkä teollinen historia ja sen juuret ulottuvat 1750-luvulle asti. Valmet-nimi syntyi vuoden 1951 alussa, kun Valtion Metallitehtaista tuli Valmet Oy. Yhtiön tuotevalikoimaan on pitkän historian aikana kuulunut mm. laivat, lentokoneet, aseet, veturit, traktorit, laivamoottorit sekä paperikoneet. Kansainvälisesti merkittäväksi paperikoneidenvalmistajaksi Valmet nousi 1960-luvun puolivälissä ja 20 vuotta myöhemmin yhtiö alkoikin keskittyä entistä voimakkaammin paperikoneisiin ja niihin liittyvään teknologiaan. Yhtiön nimi vaihtui Metsoksi vuonna 1999, kun Valmet ja Rauma yhdistyivät, mutta vaihtui takaisin Valmetiksi vuonna

2013 Metson ja Valmetin jakautuessa erillisiksi pörssiyhtiöiksi. Valmetin Automaatio-liiketoimintalinja syntyi vuonna 2015, kun Valmet osti Metson Prosessiautomaatiojärjestelmät-liiketoiminnan. (Valmet Historia 2019.)

Automaatio-liiketoimintalinja toimittaa automaation ja tiedonhallinnan järjestelmiä ja palveluja sellu-, energia-, paperi- ja prosessiteollisuudelle sekä meri- ja kaasuteollisuudelle. Automaation päätuotteet ovat hajautetut ohjausjärjestelmät, laadunhallintajärjestelmät sekä analysaattorit ja mittaukset. Automaatioratkaisujen tehtävänä on parantaa asiakkaiden liiketoimintojen kannattavuutta tehostamalla tuotannon suorituskykyä sekä kustannus-, energia- ja materiaalitehokuutta. Vuonna 2015 perustettu Automaatio-liiketoimintalinja on kehittynyt joka vuosi ja nykyisin se työllistää lähes 2000 ihmistä yli 30:ssä maassa. Kehitys voidaan havaita myös liikevaihdon kasvusta viimeisenä viitenä vuotena (kuvio 2). (Valmet vuosikatsaus 2020, 18-19.)

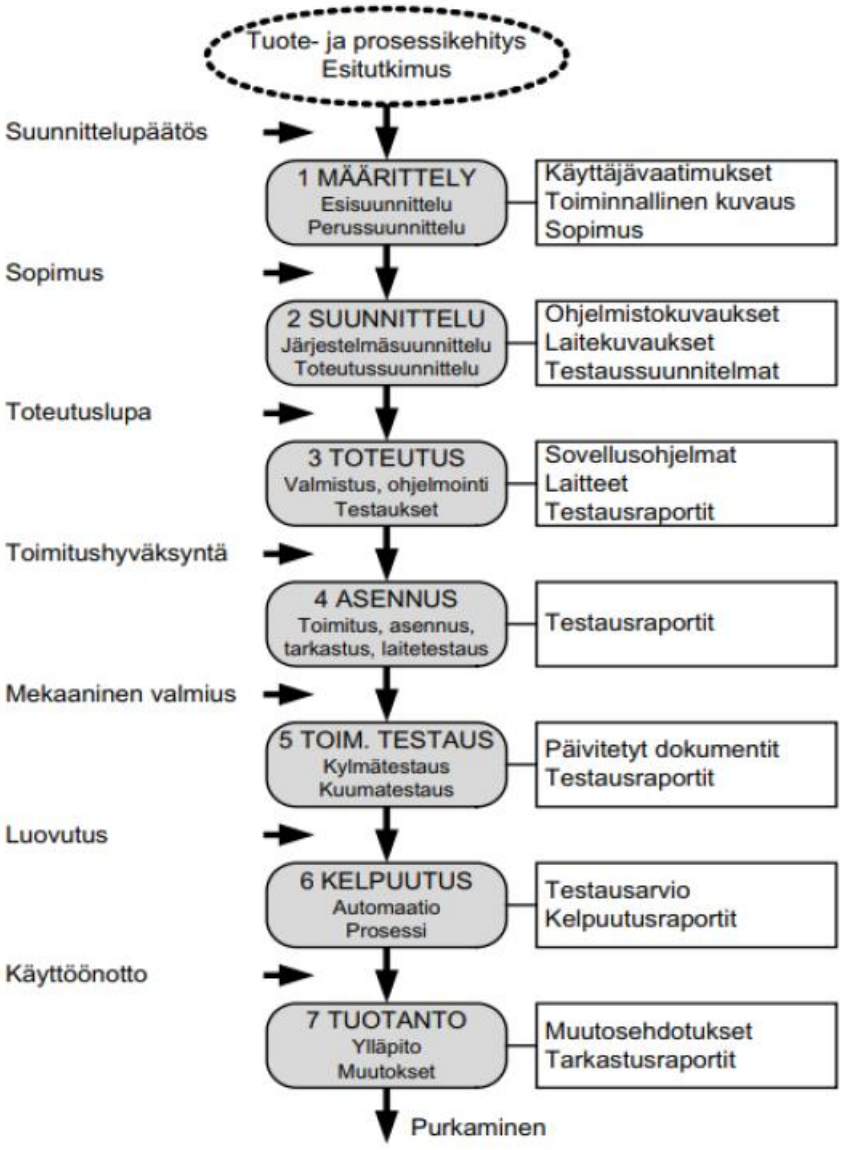


KUVIO 2. Automaation liikevaihto (Valmet vuosikatsaus 2020, 18)

.

3 AUTOMAATIOJÄRJESTELMÄN TOTEUTUS

Ennen toteutusvaiheen tarkempaa läpikäyntiä esitellään lyhyesti automaatiojärjestelmän elinkaaren muut vaiheet. Automaatiojärjestelmän elinkaari voidaan jakaa yleisesti seitsemään vaiheeseen (kuva 1), joista ensimmäinen on järjestelmän toimintojen ja vaatimusten määrittely. Tämän pohjalta toimittaja ja asiakas luovat sopimuksen siitä, mitä osapuolet lupautuvat toimittamaan toisilleen projektin aikana. Kun osapuolet ovat allekirjoittaneet sopimuksen, voidaan siirtyä suunnitteluvaiheeseen. (Strömman, Hirvonen, Hukki & Tommila 2007, 16.)



KUVA 1. Automaatiojärjestelmän elinkaari (Strömman, Hirvonen, Hukki & Tommila 2007, 16)

Suunnitteluvaiheessa toimittaja luo edellisessä vaiheessa tehtyjen määrittelyiden pohjalta suunnitelman järjestelmän toteutusta varten. Suunnitteluvaiheen päätehtäviä ovat:

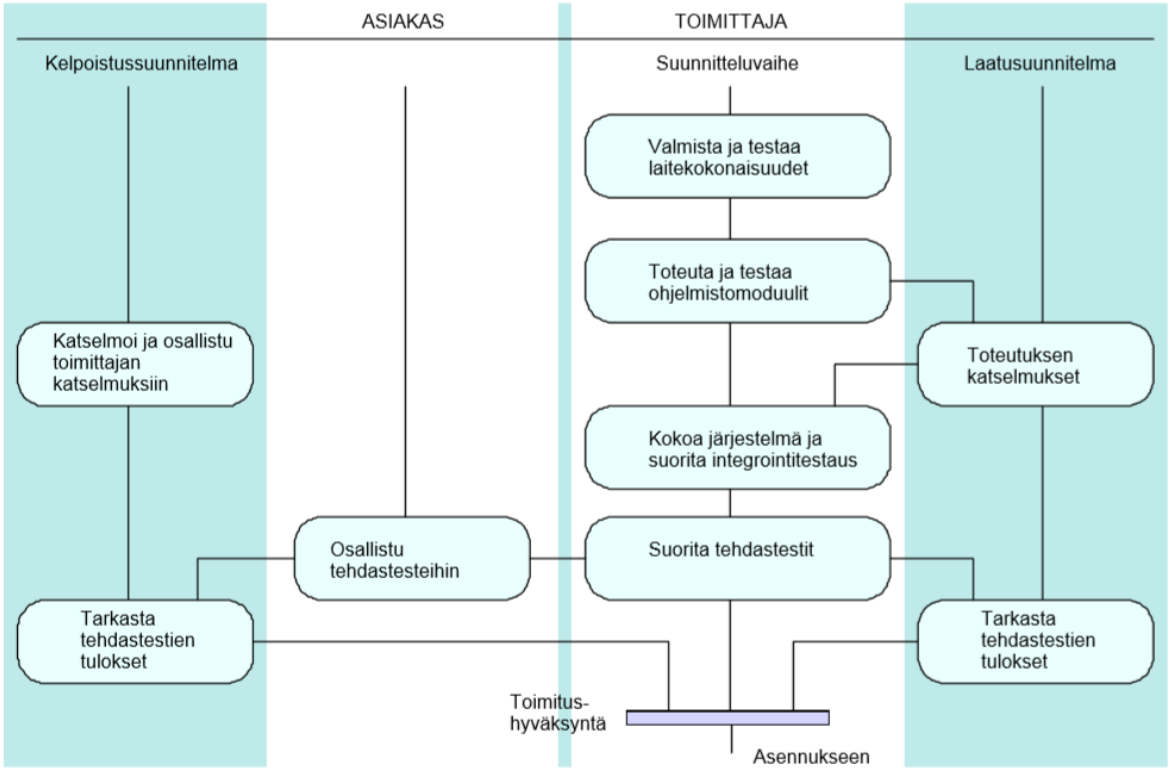
- Järjestelmäsuunnittelu, jossa määritellään järjestelmän kokonaisarkkitehtuuri
- Toteutussuunnittelu, jossa tuotetaan yksityiskohtaiset määrittelyt järjestelmälle
- Testaussuunnitelmien laatiminen

Seuraavaksi siirrytään toteutusvaiheeseen, jossa toimittaja valmistaa, kokoaa ja testaa automaatiojärjestelmän. Toteutusvaiheen päätyttyä voidaan aloittaa asennusvaihe, jossa automaatiojärjestelmä toimitetaan ja asennetaan lopulliseen käyttöympäristöön. Järjestelmälle suoritetaan laitteistotestaus, jolla varmistetaan järjestelmän olevan valmis toiminnalliseen testaukseen. Toiminnallisessa testausvaiheessa toimittaja osoittaa testauksien avulla, että asennettu järjestelmä vastaa määrittelyvaiheen toiminnallista kuvausta ja sopimusta. Hyväksyttyjen testausten perusteella järjestelmä voidaan luovuttaa asiakkaalle ja aloittaa tuotanto. Tuotantovaihe päättyy, kun järjestelmä tulee elinkaarensa loppuun, jolloin järjestelmä puretaan. (Strömman, Hirvonen, Hukki & Tommila 2007, 16.)

Seuraavaksi esitellään automaatiojärjestelmän toteutusvaihe (kuva 1) tarkemmin, koska se liittyy tämän työn sisältöön olennaisesti. Toteutusvaihe tarkoittaa automaatiojärjestelmän toimittajan tekemää ohjelmistojen ja laitteiden hankintaa tai valmistusta ja laitteiden kokoonpanoa. Toteutusvaiheesta vastaa automaatiojärjestelmän toimittaja. Yleensä toimittaja käyttää kaupallisesti saatavilla olevia laitteisto- ja ohjelmistokomponentteja, jotka se hankkii ulkopuoliselta valmistajalta. Toimittaja vastaa myös toteutusvaiheen dokumentoinnista, johon kuuluu laitteiden tekniset dokumentit sekä asennus- ja käyttöohjeet. Dokumentointi sisältää myös testaussuunnitelmien laatimisen sekä tehdastestien dokumentaation. (Tommila 2001, 19.)

Laitteiston saatuaan toimittaja tekee tarvittavan kokoonpanon ja laiteasetukset eli konfiguroinnin. Työn edetessä toimittaja tekee moduulitestauksia erilaisille laite- ja ohjelmamoduuleille sekä integraatiotestausta koko järjestelmälle. Viimeisenä suoritetaan tehdastestaus eli FAT (Factory Acceptance Test) yhdessä asiakkaan

kanssa. Toteutusvaiheen osatehtäviä on havainnollistettu kuvassa 2. (Tommila 2001, 55-56.)



KUVA 2. Toteutusvaiheen tehtävät (Tommila 2001, 56)

Tommilan (2001, 57) mukaan toteutusvaihe voidaan jakaa kahteen alavaiheeseen, valmistukseen ja kokoonpanoon sekä testaukseen.

3.1 Valmistus ja kokoonpano

Tässä vaiheessa hankitaan tarvittavat automaatio- ja tietokonelaitteet alihankkijoilta ja valmistajilta laitteiston toteutussuunnittelun tuottamien määrittelyjen mukaisesti. Tämän jälkeen järjestelmä kootaan ja asennetaan toimittajan tiloissa suunnitelman mukaiseksi kokonaisuudeksi. Tähän kuuluu laitteiston ja verkon kokoaminen sekä ohjelmistojen lataus ja niiden konfigurointi. Laitteiston kokoamisen jälkeen suoritetaan yleensä laitekokoonpanon katselmus tai testaus. Tällä varmistetaan, että kokoaminen ja kytkennät on toteutettu toimittajan laatimien suunnitelmien mukaisesti. Lopuksi suoritetaan testaukset laitekokoonpanon toi-

mivuuden toteamiseksi. Toimivan ja suunnitelmien mukaisesti toteutetun laitekokoonpanon testauksen jälkeen alkaa ohjelmiston testaus. Ohjelmiston perustoimintoja ovat esimerkiksi säätöpiirit, sekvenssit, hälytykset ja lukitukset sekä laiteohjaukset. Perustoimintojen lisäksi siihen kuuluu mm. prosessin optimointi ja erilaiset raportit. (Tommila 2001, 56-59.) Opinnäytetyö kohdistuu laitteiston testaukseen, joten ohjelmistoa ja sen testausta ei esitellä tämän tarkemmin.

3.2 Testaus

Testausvaiheen tarkoitus on ensin integraatiotestauksella varmistaa, että ohjelmisto ja laitteisto on koottu oikein. Tämän jälkeen tehdastestauksella osoittaa asiakkaalle, että järjestelmäkokonaisuus toimii määritellyllä tavalla. (Tommila 2001, 60.)

Integrointitestaus

Integrointitestauksella on tarkoitus varmistaa, että automaatiojärjestelmän ohjelmisto ja laitteisto on koottu oikein. Testauksessa yhdistetään ohjelmisto- ja laitteistoelementtejä tai molempia, kunnes koko järjestelmä on tullut testatuksi. Integrointitestaus tapahtuu suunnitteluvaiheessa laaditun testaussuunnitelman mukaan ja se voi alkaa jo järjestelmän kokoonpanon aikana. Testausta varten laitteisto kootaan mahdollisimman täydellisenä testaustilaan ja se suoritetaan projektin omilla laitteilla aina, kun se on mahdollista. Integrointitestauksen päätyttyä järjestelmätoimittaja laatii testausraportin, jonka jälkeen aloitetaan tehdastestaukset. (Tommila 2001, 61-62.)

Integrointitestauksesta käytetään Valmetilla nimeä järjestelmätestaus. Järjestelmätestaukseen tarkoitettua testausohjetta suunniteltaessa on huomioitu järjestelmän toiminnalliseen turvallisuuteen liittyvien direktiivien ja standardien vaatimukset. Testejä on myös kehitetty asiakasprojekteissa muodostuneiden kokemusten perustella. Järjestelmätestauksen suorittaa Valmetin toimituskeskus, joka myös suunnittelee ja tilaa laitteiston, josta automaatiojärjestelmä rakentuu. Toimituskeskus toteuttaa myös järjestelmän kokoamisen ja alustuksen eli asentaa käyttöjärjestelmät ja tarvittavat ohjelmat sekä konfiguroi palvelimet. Tässä yhteydessä palvelimella tarkoitetaan kaikkia tietokoneita, jotka liittyvät Valmet DNA

-automaatiojärjestelmään. DNA (Dynamic Network of Applications) on lyhenne Valmetin automaatiojärjestelmän nimestä. Toimituskeskus testaa järjestelmän, jonka jälkeen se siirtyy projektiorganisaatiolle. Projektiorganisaatio vastaa järjestelmän sovelluksen testauksesta sekä tehdastestauksesta. (Jokinen 2020a.)

Tehdastestaus

Hyväksytyn integrointitestauksen jälkeen suoritetaan tehdastestaus, joka on tilaajan ja järjestelmätoimittajan yhteistyössä toteuttama toiminnallinen testaus. Tehdastestaus suoritetaan yleensä järjestelmätoimittajan tiloissa ja siinä varmistetaan, että kenttälaitteita lukuun ottamatta järjestelmän ohjelmistot ja laitteet täyttävät niille asetetut vaatimukset. Testaukset perustuvat määrittelyvaiheessa tehtyyn sopimukseen ja suunnitteluvaiheessa laadittuihin testaussuunnitelmiin. Integraatiotestauksen tapaan myös tehdastestauksen päätyttyä laaditaan siitä testausraportti. Toteutusvaihe päättyy, kun asiakas ja toimittaja yhdessä hyväksyvät tehdastauksen. Tämän jälkeen automaatiojärjestelmä voidaan toimittaa lopulliseen käyttöympäristöön eli automaatiojärjestelmälle on saatu toimitushyväksyntä. (Tommila 2001, 62.)

Laitteiston kahdennuksen testaus kuuluu myös osana tehdastestausta ja siihen on annettu ohje standardissa SFS-EN 62381. SFS tarkoittaa Suomen Standardisoimisliittoa, joka on suomalainen standardisoinnin keskusjärjestö. SFS standardin 62381 mukaan prosessiohjaimien, yhteyden ja verkon sekä tehonsyötön ja I/O-järjestelmien kahdennus tulisi testata ja dokumentoida (SFS-EN 62381 2012, 25).

4 KAHDENNUKSEN TOTEUTUS

Prosessiautomaatiojärjestelmissä on jokaisella toimialalla erilaisia vaatimuksia järjestelmän luotettavuuteen. Järjestelmiltä vaaditaan kahdennusta esimerkiksi ihmisten ja laitteiden turvallisuuden varmistamiseksi tai jatkuvan tuotannon takaamiseksi. Jatkuvat prosessit vaativat useiden eri vaiheiden synkronointia. Yhden vaiheen pysähtyminen voi aiheuttaa ongelmia seuraavissa vaiheissa ja näin vahingoittaa koko prosessin toimivuutta. Kriittisillä teollisuuden toimialoilla kuten kaivostoiminta tai ydinvoima, on prosessin toimittava ilman suunnittelemattomia käyttökatkoksia tai turvallisuuden valvonnan keskeytyksiä. (Lynch 2013.)

Kaikille prosessiautomaatiojärjestelmien osille ei ole välttämätöntä toteuttaa kahdennusta. Kahdennuksen toteuttamatta jättäminen ei ole kuitenkaan kannattava säästötoimenpide. Jos kahdennusta ei toteuteta, saattaa se aiheuttaa ongelmia prosessin hallinnassa tai altistaa suuremmille ongelmille. Nämä ongelmat aiheuttavat yleensä merkittäviä taloudellisia kustannuksia. Seuraavaksi käydään läpi muutamia käytännöllisiä ja tehokkaita kahdennuksen tasoja prosessin ohjauksessa. Ensimmäinen kahdennuksen taso on tehonsyötön kahdentaminen, jossa laitteen tehonsyöttö varmistetaan kahdella teholähteellä. Sähkökatkon tai toisen teholähteen vikaantuessa varalla oleva teholähde alkaa syöttää laitteille virtaa. (Stemple 2003.)

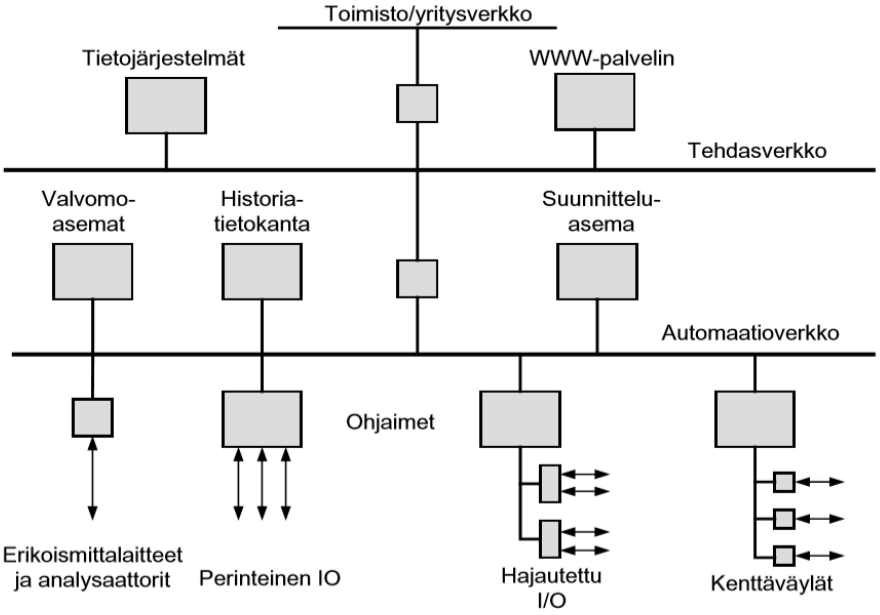
Kahdennuksen toinen taso on verkkokytkimien kahdentaminen, jossa toiminnan kannalta kriittiset laitteet kahdennetaan kahdella verkkokytkimellä. Esimerkiksi prosessiohjain on kriittinen laite ja sen toiminta on varmistettava, jos toinen verkkokytkimistä ei ole toiminnassa. Tämä asettaa kuitenkin vaatimuksen, jossa prosessiohjaimen on tuettava kahdennettuja verkkokytkimiä ja ohjaimen on kyettävä valitsemaan toiminnan kannalta edullisempi verkkokytkin tiedonsiirtoreitin muodostamiseen. Verkkokytkimien kahdentaminen vähentää järjestelmässä syntyviä käyttökatkoksia. Seuraava taso on koko verkon kahdentaminen, jossa tiedonsiirtoon luodaan kaksi vaihtoehtoista reittiä. (Stemple 2003.)

Viimeisenä tasona voidaan pitää koko järjestelmän kahdentamista. Se koostuu kahdennetuista verkkokytkimistä ja laitepareista eli automaatiojärjestelmän kaikki

laitteet on kahdennettu. Täysin kahdennettu järjestelmä takaa erittäin luotettavan automaatioverkon, jossa tiedon häviäminen on minimoitu. Koko järjestelmän kahdentaminen ei välttämättä ole mahdollista budjetin ja tilarajoitteiden puitteissa. (Stemple 2003.)

4.1 Prosessiohjain

Yleisnimitys automaatiojärjestelmän tiedonkäsittelylaitteille on ohjain. Se voi olla esimerkiksi DCS-järjestelmän prosessiohjain, ohjelmoitava logiikka, tietokone tai mikrokontrolleri. (Tommila 2001, 110.) Tässä opinnäytetyössä käsitellään DCS-järjestelmän (Distributed Control System) eli hajautetun automaatiojärjestelmän prosessiohjainta. Prosessiohjaimen tehtävä on liittää automaatiojärjestelmä ohjattavaan prosessiin sekä suorittaa mittaus-, säätö, logiikka- ja ohjaustoiminnot. Prosessiohjaimet välittävät prosessitiedot järjestelmän muille laitteille ja vastaavasti saavat prosessin toimintaa koskevat ohjaukset muilta laitteilta. Muilla laitteilla tarkoitetaan automaatiojärjestelmän hallintalaitteita, joita voivat olla esimerkiksi kuvassa 3 olevat laitteet. Ohjain liittyy prosessiin joko suoraan siihen asennettujen prosessiliitäntäyksiköiden kautta tai väylän avulla lähelle prosessia hajautettujen erillisten I/O-järjestelmien kautta. (Kippo & Tikka 2008, 48.) Esimerkki automaatiojärjestelmän rakenteesta on esitetty kuvassa 3.



KUVA 3. Automaatiojärjestelmän rakenne (Strömman, Hirvonen, Hukki & Tommila 2007, 11)

Kokonaisjärjestelmä voidaan jakaa kolmeen tasoon (kuva 3). Kenttälaitteet ja -väylät ovat alimpana, joista käytetään yleisesti nimitystä instrumentointi. Keskellä ovat varsinaiset prosessiohjaimet ja muut hallintalaitteet. Ylimmällä tasolla automaatiojärjestelmässä ovat tuotannonhallinnan tietojärjestelmät. (Strömman, Hirvonen, Hukki & Tommila 2007, 10.)

Valmet DNA -automaatiojärjestelmän prosessiohjaimien tuoteperheestä käytetään lyhennettä ACN (Application and Control Node). ACN-tuoteperheeseen kuuluu useampi prosessiohjain eli PCS (Process Control Server), joita voidaan käyttää eri laajuisissa automaatiojärjestelmissä. Kuvassa 4 on esitetty esimerkki ACN-tuoteperheen prosessiohjaimesta. (Valmet DNA 2019.)

KUVA 4. ACN-prosessiohjain ja I/O-kehikko (Control point, n.d.)

Kuvassa 4 vasemmalla on ACN-prosessiohjain, jonka oikealla puolella on I/O-kehikolle jännitteen syöttävä teholähde (IPSP). Sen oikealla puolella on I/O-väylän ohjain (IBC). I/O-väylän ohjaimen jälkeen kuvassa on I/O-kortit, joiden kautta automaatiojärjestelmä liitetään kenttälaitteisiin.

4.2 Prosessiohjaimien kahdennus

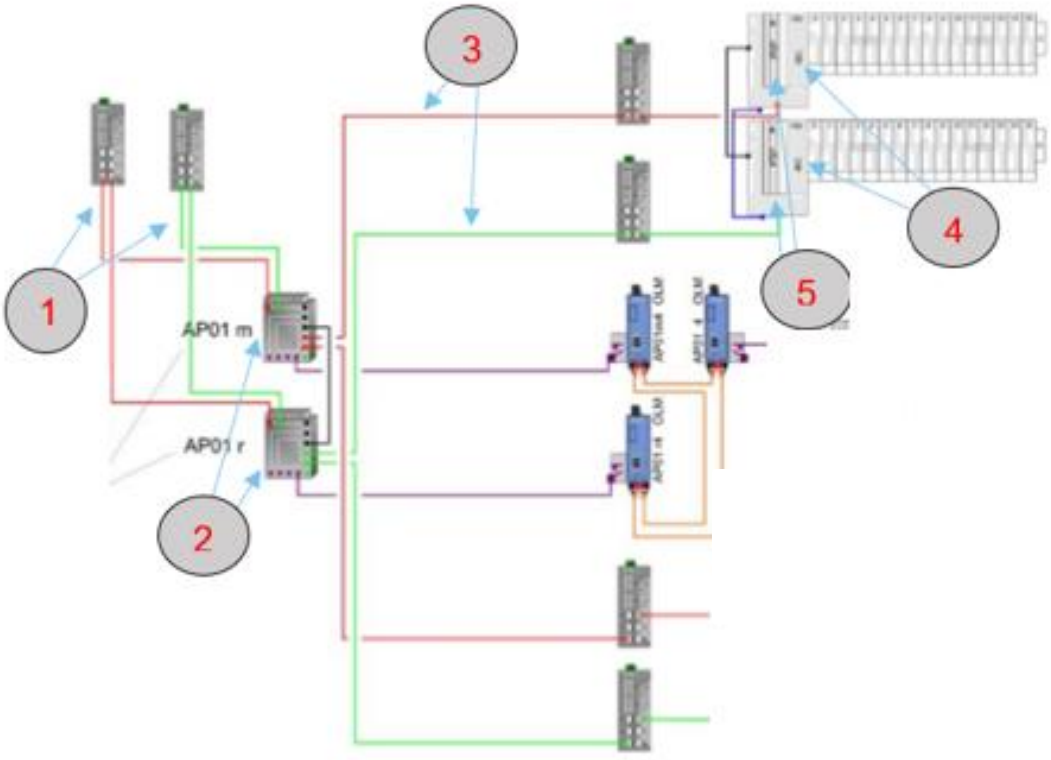
Valmet DNA on hajautettu automaatiojärjestelmä, jossa osaprosessit on jaettu eri ohjaimille. Jos yksi ohjain pysähtyy tai vikaantuu, se ei vaikuta muihin prosessin osiin. Valmet DNA -järjestelmä käyttää yleisesti rengas-verkkotopologiaa, mutta myös tähtitopologia on mahdollinen toteutustapa. Molemmat verkkotopologiat tuottavat vakaan ja kahdennetun verkon sekä nopean vaihdon varayhteyksille.

Verkkoyhteys palvelintietokoneiden ja ACN-prosessiohjaimien välillä on kahdennettu, joten varmuuskopiopolku on aina käytettävissä. Pää- ja varayhteydet käyttävät erillisiä Ethernet-kortteja ja verkkokytkimiä. Varayhteys siirtyy automaattisesti käyttöön, jos pääyhteys vikaantuu. Valmet DNA -automaatioverkko tulee erottaa normaalista toimistoverkosta reitittimellä. (Valmet DNA 2019.)

Valmet DNA -automaatiojärjestelmässä prosessiohjaimien kahdennus on suunniteltu seuraavien periaatteiden mukaan: (kuva 5) (Jokinen 2020b.)

1. Kaksi Ethernet-yhteyttä laitteiden välillä
2. Kaksi prosessiohjainta
3. Kaksi Ethernet-pohjaista I/O-väylää
4. Kaksi I/O-väylän ohjainta (IBC)
5. Kaksi I/O-teholähdettä (IPSP)

Kuvassa 5 on esitetty myös kahdennettu PROFIBUS-liityntä sekä kahdennettu liityntä kolmannen osapuolen ohjauslaitteeseen. Näiden esittely ei ole kuitenkaan tämän työn kannalta olennaista.



KUVA 5. Kahdennettujen prosessiohjaimien periaatekuva (Jokinen 2020b, muokattu)

Kahdennetut ACN-prosessiohjaimet nimetään pää- ja varaohjaimeksi. Nimeämisellä ei ole ohjaimien toiminnan kannalta merkitystä. Toinen ohjaimista toimii aktiivisena ohjaimena ja toinen passiivisena. Aktiivinen ohjain suorittaa kaikki ohjaukseen liittyvät toiminnot. Prosessiohjaimien vaihto eli passiivisen ohjaimen siirtyminen aktiiviseksi tapahtuu seuraavin perustein:

- Aktiivinen ohjain vikaantuu
- Yhteys I/O-kehikkoon vikaantuu
- Komento

Molemmat prosessiohjaimet seuraavat toistensa tilaa RCOK-kaapelin (Redundant Condition OK) eli kahdennuksen tilanteen seurantakaapelin avulla, joka on kuvassa 5 oleva musta kaapeli prosessiohjaimien välillä. Kun passiivinen ohjain havaitsee RCOK-kaapelin välityksellä aktiivisen ohjaimen vikaantumisen, se ottaa aktiivisen roolin ja siirtyy ohjaamaan prosessia. Molemmat prosessiohjaimet laskevat myös jatkuvasti I/O-korttien lukumäärää. Prosessiohjaimien vaihto tapahtuu, jos aktiivinen ohjain havaitsee vähemmän I/O-kortteja kuin passiivinen ohjain. Viimeinen prosessiohjaimien vaihdon aiheuttava peruste on komento eli käyttäjä voi komennon avulla suorittaa prosessiohjaimien vaihdon. (Jokinen 2020b.)

5 KAHDENNUSTESTAUKSEN AUTOMATISOINTI

Opinnäytetyön tarkoituksena oli automatisoida Robot Framework -testausautomaatiokehyksellä prosessiohjaimien kahdennustestaus vikatilanteessa, jossa tehonsyöttö pääprosessiohjaimelta katkeaa. Tässä tilanteessa tapahtuu prosessiohjaimien vaihto ja passiivinen prosessiohjain siirtyy aktiiviseksi ohjaimeksi ilman prosessin pysähtymistä. Testauksen automatisoinnin pohjana käytettiin Valmetin toimituskeskuksen manuaaliseen testaukseen tarkoitettua testausohjetta.

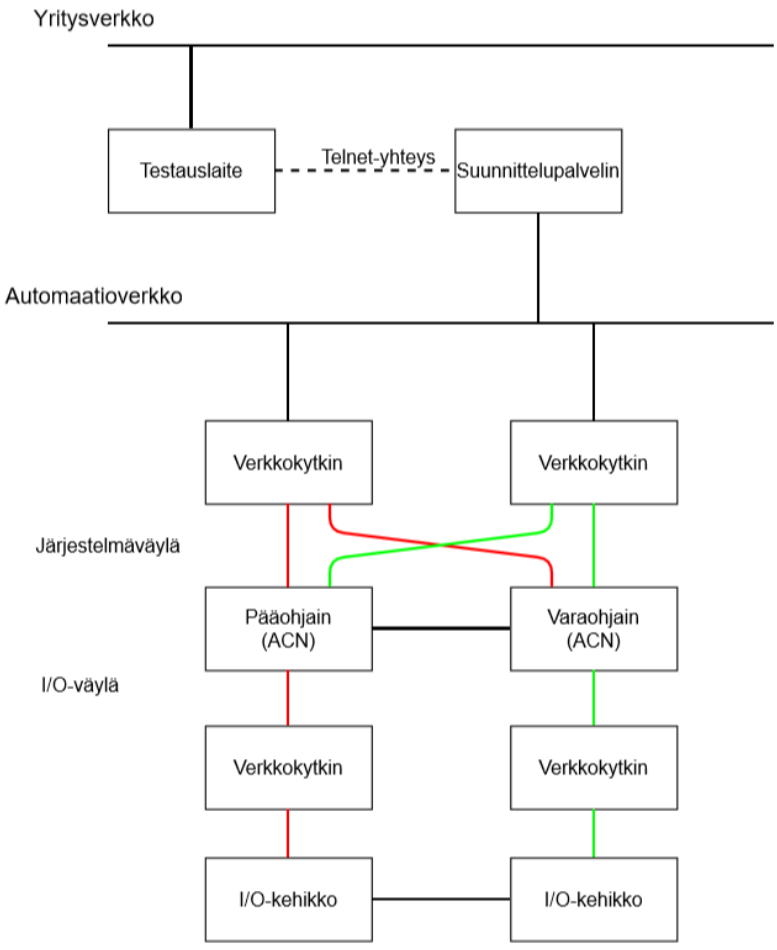
Ensimmäisenä työssä määritettiin käyttöympäristö, josta automaattinen kahdennustestaus suoritetaan. Tätä varten haastateltiin toimituskeskuksen integraatiotestaajia sekä tuotekehitysosaston testaajia. Integraatiotestaaja suorittaa järjestelmätestauksen ja on automaattisen kahdennustestauksen käyttäjä. Tuotekehitysosaston testaajat auttoivat automaattisen testauksen luomisessa ja he toimivat myöhemmin sen jatkokehittäjinä yhteistyössä integraatiotestaajien kanssa. Molempien osastojen toiveiden ja vaatimusten yhteensaatto vei aikaa, mutta se oli kuitenkin työn kannalta tärkeä vaihe. Haastatteluiden pohjalta käyttöympäristöksi valittiin tuotekehitysosaston käytössä oleva ohjelmistotestausympäristö. Tämä tarkoittaa käytännössä sitä, että integraatiotestaaja käynnistää testauksen ohjelmistotestausympäristöstä. Näin varmistettiin, että luotu automaattitestaus on myöhemmin helposti saatavilla ja kehitettävissä.

5.1 Käyttöympäristö ja vaatimukset

Automaattisen prosessiohjaimien kahdennustestauksen toteutustavaksi valittiin käyttöympäristö, jossa on erillinen testilaite. Testilaitteella sijaitsee Robot Framework -ohjelmakoodi, jonka avulla kahdennustestaus suoritetaan. Testilaitteena toimii kannettava tietokone, johon on asennettu testaukseen vaadittavat ohjelmat. Testilaitteelle asennetut ohjelmaversiot ja muut vaatimukset ovat:

- Python 3.8.1
- Verkkoyhteys yritysverkkoon ja automaatioverkkoon
- Oikeudet Jenkins-automaatiopalvelimelle
- Robot Framework 3.1.2

Prosessiohjaimien kahdennustestauksen käyttöympäristössä on kahdennetut prosessiohjaimet ja muut laitteet sekä kahdennettu automaatioverkko (kuva 7). Automaatioverkossa punainen väri kuvaa niin kutsuttua pääyhteyttä ja vihreä väri varayhteyttä. Testilaite on kytkettynä automaatioverkkoon sekä yritysverkkoon. Testilaite voidaan yhdistää yritysverkkoon langallisesti Ethernet-kaapelilla tai langattomasti. Yhteys automaatioverkkoon muodostetaan suunnittelupalvelimen kautta TELNET-etäyhteydellä. Suomisen (2002, 486) mukaan TELNET-etäyhteydellä käyttäjä voi kirjautua yhteisverkon muihin tietokoneisiin. TELNET muodostaa yhteyden ja siirtää näppäinkomennot käyttäjän näppäimistöltä suoraan etätietokoneeseen aivan kuin ne olisi kirjoitettu suoraan etätietokoneeseen liitetyltä näppäimistöltä. Yhteys etätietokoneeseen luodaan IP-osoitteen avulla.



KUVA 7. Rakennekuva testiympäristöstä

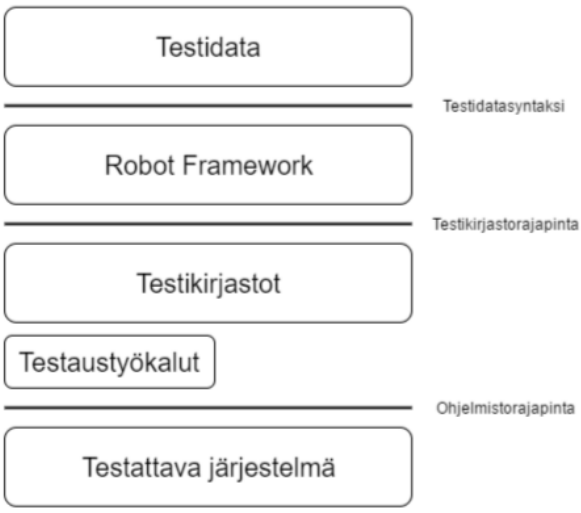
TELNET-etäyhteyden avulla käytetään Valmetin suunnittelupalvelimelta Debugger-ohjelmaa, joka on järjestelmän ja sovellusten testaamiseen ja virheenkorjaamiseen tarkoitettu työkalu. Debugger-ohjelmaan syötetyiden näppäinkomentojen

avulla voidaan lukea prosessiohjaimien diagnostiikkaa ja tällä tavoin varmistua, että vikatilanteessa kahdennetut prosessiohjaimet toimivat oikealla tavalla ja antavat tarvittavat virheilmoitukset.

5.2 Robot Framework -testausautomaatiokehys

Robot Framework (RF) pohjautuu Pekka Klärckin vuonna 2006 kirjoittamaan diplomityöhön, joka käsittelee suuren mittakaavan testausautomaatiokehysympäristöjä. Klärck loi osana diplomityötään avainsanapohjaisen prototyypin, jota hän oli mukana jatkokehittämässä Nokia Networksin asiakasprojekteissa. RF julkaistiin kesäkuussa 2008 avoimen lähdekoodin ohjelmistona, jonka kehittämisessä Klärck on edelleen vahvasti mukana. (Klärck, n.d.)

RF on Python-ohjelmointikieleen pohjautuva hyväksyntätestaukseen suunniteltu testiautomaatiokehys ja sillä on modulaarinen hierarkia, joka on esitelty kuvassa 6. Käynnistyessään RF käsittelee testidatan ja suorittaa käyttäjän määrittämät testitapaukset, joiden perusteella testauksen tulos määräytyy. RF hyödyntää laajennettavaa avainsanapohjaista menetelmää ja sen sovelluskehyksen ympärillä on laaja kokonaisuus erilaisia geneerisiä testikirjastoja ja työkaluja. Testiautomaatiokehyksen ydin ei tiedä mitään testattavasta järjestelmästä vaan yhteys järjestelmään tapahtuu testikirjastorajapintojen avulla. (Robot Framework User Guide 2019.)

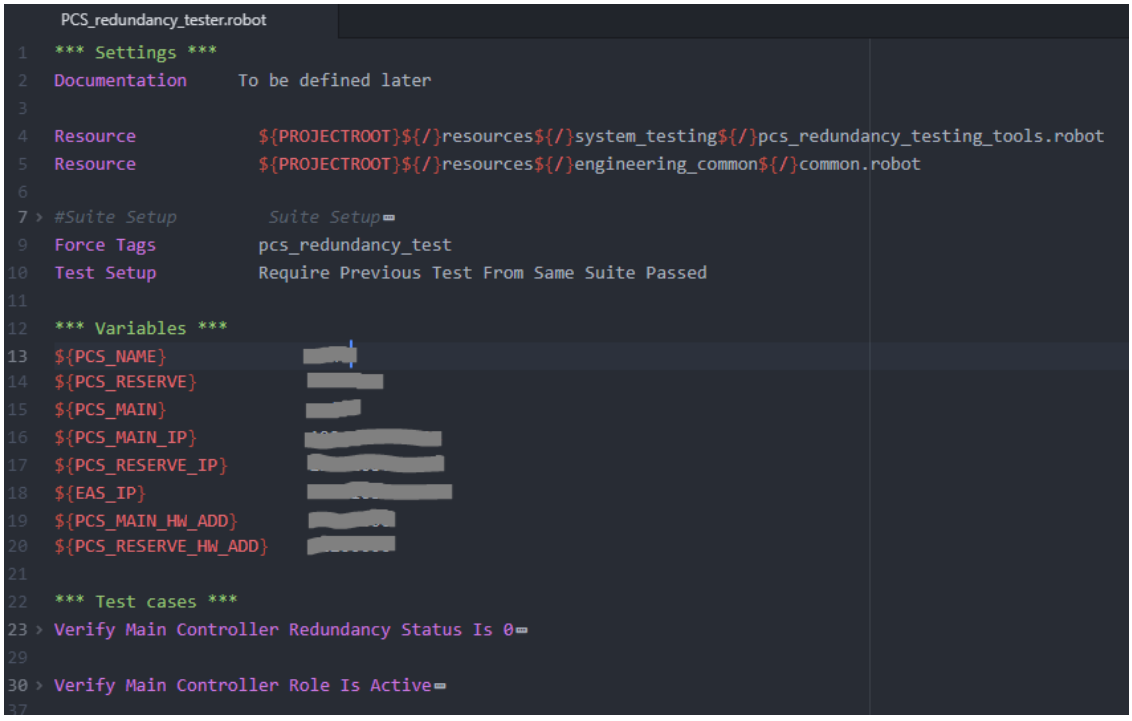


KUVA 6. Robot Framework -hierarkia (Robot Framework User Guide 2019)

Avainsanoihin perustuvan testauksen tarkoitus on määrittää testiskriptejä vastaavia avainsanoja, joiden yhdistävänä linkkinä testityökalu toimii. Skriptillä tarkoitetaan komentoriviohjelmaa, joka on kirjotettu esimerkiksi Python-ohjelmointikielellä. Avainsanojen avulla voidaan parametrien määrittäminen toteuttaa ilman monimutkaisten skriptien muokkaamista. Avainsanat vähentävät testien teknisyyttä ja monimutkaisuutta, joten testit voi suorittaa myös ihminen, jolla ei ole kattavaa ohjelmointitietämystä. (Hass 2008, 377-378.)

RF tarjoaa käyttäjälle mahdollisuuden luokitella testejä joko testikokonaisuus tai testitapaus tasolla. Näitä tunnisteita hyödyntämällä voidaan luoda monimutkaisiakin testejä tai suorittaa kerrallaan vain yksi testitapaus. Yhden testitapauksen suorittaminen kerrallaan tekee virheiden ja ongelmien selvittämisestä helpompaa. (Robot Framework User Guide 2019.)

Automaattisen kahdennustestauksen ohjelmointi toteutettiin Robot Framework-testiautomaatiokehyksellä. Kuvassa 8 on esitetty ohjelmakoodin yleisrakenne, jonka tiedoista osa on peitetty tietoturvallisuussyistä. Vihreällä tekstillä olevat osuudet jakavat ohjelmarakenteen eli testikokonaisuuden eri osa-alueisiin, joita ovat asetukset, muuttujat, testitapaukset ja avainsanat.



KUVA 8. Robot Framework -ohjelmakoodin yleisrakenne

Asetuksiin voidaan tuoda resurssi- ja muuttujatiedostoja ja testikirjastoja sekä määrittää metatietoja testikokonaisuudelle ja testitapauksille. Metatiedot ovat tietoa, joka tuodaan jostain muusta tietosisällöstä. Esimerkiksi kuvassa 8 oleva testiasetus (Test Setup), jossa hyödynnetään jo olemassa olevaa ohjelmakoodia. Ohjelmakoodi sisältää toiminnallisuuden, joka vaatii edellisen testitapauksen läpäisyn ennen kuin testikokonaisuudessa siirrytään seuraavaan testitapaukseen. Resurssitietoihin määritellään tiedostopolku, josta tieto tuodaan. Muuttujat-osuuden tiedot ovat uniikkeja globaaleja muuttujia, joita voidaan käyttää missä tahansa testikokonaisuudessa. Testitapauksiin on myös mahdollista määrittää paikallisia muuttujia, jotka toimivat ainoastaan tietyn testitapauksen sisällä. Muuttujien käytön selkeyden vuoksi globaalit muuttujat on kirjoitettu suuraakkosin ja testitapauksen sisäiset muuttujat pienaakkosin.

Testikokonaisuuden toiminnallisuus luodaan testitapauksien avulla (kuva 9). Testikokonaisuus etenee yksi testitapaus kerrallaan, ja testitapaus joko hyväksytään tai hylätään. Kuten aiemmin mainittiin, tässä testikokonaisuudessa vaaditaan hyväksytty edellinen testitapaus tai muutoin testin suoritus keskeytyy. Kuvassa 9 on esitetty yksi testitapaus, jossa luetaan pääohjaimelta kahdennuksen tilannetta. Ensimmäisenä muodostetaan yhteys suunnittelupalvelimen Debugger-ohjelmaan, joka on määritetty käyttäjän luomaksi avainsanaksi ja se esitellään myöhemmin tarkemmin kuvassa 11. Käyttäjän määrittämät avainsanat ovat niin kutsuttuja korkean tason avainsanoja ja ohjelman sisäisten kirjastojen avainsanat matalan tason avainsanoja. Kuvassa 9 on esimerkki matalan tason avainsanasta *Sleep*, joka kuluu RF:n sisäänrakennettuun kirjastoon.

```
Verify Main Controller Redundancy Status Is 0
    Connect To EAS Debugger
    Telnet.Write    p v :e:${PCS_MAIN}:${PCS_NAME}:redundancy:fault
    Sleep    2
    ${promt}=    Telnet.Read
    Should Contain    ${promt}    MEMBER IS binstat IS bin <0>
```

KUVA 9. Esimerkki Robot Frameworkin testitapauksesta

Yhteyden muodostamisen jälkeen hyödynnetään Robot Frameworkin TELNET-kirjaston avainsanaa *Write*, jonka avulla voidaan kirjoittaa Debugger-ohjelman

komentoriville. Avainsanan perässä olevalla komennolla saadaan prosessiohjaimen diagnostiikalta kahdennuksen vikatieto. Seuraavaksi luetaan saatu vaste Debugger-ohjelman komentoriviltä TELNET-kirjaston avainsanalla *Read* ja sen odotetaan sisältävän arvo nolla eli kahdennusvika ei ole päällä. Seuraavassa kuvassa on esitetty avainsanan *Read* perustana oleva ohjelmakoodi.

```
def read(self, loglevel=None):
    """Reads everything that is currently available in the output.

    Read output is both returned and logged. See `Logging` section for more
    information about log levels.
    """
    self._verify_connection()
    output = self._decode(self.read_very_eager())
    if self._terminal_emulator:
        self._terminal_emulator.feed(output)
        output = self._terminal_emulator.read()
    self._log(output, loglevel)
    return output
```

KUVA 10. Read-avainsanan ohjelmakoodi

Kuvassa 10 esitetty ohjelmakoodi ei näy käyttäjälle testitapauksen rakenteessa vaan se on piilotettu taustalle. Avainsanojen perustana olevia ohjelmakoodeja tarvitsee tulkita, kun halutaan tietää avainsanan toiminnasta tarkemmin tai selvittää vaatiiko avainsana joitakin parametreja toimiakseen.

```
*** Keywords ***
Suite Setup

Suite Teardown

Connect To EAS Debugger
    Open Debugger Connection To Node    ${EAS_IP}
    Set Newline    LF
    Telnet.Write    applmode
    Sleep    2
```

KUVA 11. Esimerkki Robot Frameworkin avainsanasta

Yhteyden muodostaminen suunnittelupalvelimen Debugger-ohjelmaan on määritetty käyttäjän avainsanaksi (kuva 11). Tämän avainsanan sisällä on toinen käyttäjän avainsana, jonka tiedostopolku on määritetty testikokonaisuuden resurssitiedostoihin. Sille on asetettu määrittelyksi suunnittelupalvelimen IP-osoite, jonka avulla yhteys luodaan. Tämän jälkeen kirjoitetaan komento *applmode*, jolla saadaan Debugger-ohjelman sovellustilan komennot käyttöön.

Avainsanojen käyttö mahdollistaa, että ohjelmoinnista tietämätön käyttäjä ymmärtää ohjelmakoodin käyttötarkoituksen sekä tekee ohjelmakoodin yleisrakenteesta helposti tulkittavaa. Avainsanojen perustana on kuitenkin skripti, jonka tulkitseminen vaatii ohjelmointiosaamista. RF:n sisäänrakennetuiden kirjastojen avainsanojen käyttö nopeuttaa ohjelmakoodin kirjoittamista, koska kaikkia skriptejä ei tarvitse luoda itse. Tästä hyvänä esimerkkinä voidaan verrata kuvia 9 ja 10. Kuvassa 9 *Read*-avainsana käyttää vain yhden rivin testitapauksesta, kun avainsanan perustana oleva koodi (kuva 10) vaatii useamman rivin käyttöä.

Korkeamman tason eli itse määritettyjen avainsanojen käyttö mahdollistaa skriptin uudelleenkäytön uusien testikokonaisuuksien rakentamisessa. Tämä nopeuttaa ja selkeyttää uusien testikokonaisuuksien luontia, kun ei tarvitse kopioida alkuperäistä koodia, vaan voidaan käyttää sen sijasta pelkästään avainsanaa. Toisaalta tämä tekee testikokonaisuudesta niin sanotusti monitasoisen, kun käytetyn avainsanan toiminallisuus on haettu toisesta testikokonaisuudesta.

5.3 Testin suoritus

Tarkoituksena oli toteuttaa automaattinen prosessiohjaimien kahdennustestaus tilanteessa, jossa tehonsyöttö pääprosessiohjaimelta katkeaa. Automaattisen testaussovelluksen luontiin hyödynnettiin järjestelmätestausohjetta. Testit olivat siis valmiiksi määritettyjä, joten tässä opinnäytetyössä keskityttiin testien automatisointiin. Automatisointi toteutettiin RF-testausautomaatiokehyksellä vaiheittain, aloittaen yhteyden luomisesta suunnittelupalvelimelle ja sen Debugger-ohjelmaan. Toimivan yhteyden jälkeen testikokonaisuuteen lisättiin järjestelmätestausohjeen mukaisesti testitapaukset kahdennuksen testaukseen.

Kuvassa 12 on esitetty automaattisen kahdennustestauksen testitapaukset supistettuna eli testitapauksien varsinainen toiminallisuus on piilotettuna. Ensimmäisenä prosessiohjaimien diagnostiikasta varmistetaan, että pääprosessiohjain toimii aktiivisena ohjaimena (Main Controller Role Is Active) sekä tarkistetaan kahdennusvian tilanne (Main Controller Redundancy Status Is 0). Tämän jälkeen tulostetaan I/O-konfiguraatio, jonka pääprosessiohjain havaitsee väylällä (Print Main Controller Field Bus Configuration). Varsinainen kahdennustestaus voidaan aloittaa, jos vika ei ole päällä eli prosessiohjaimet toimivat normaalisti.

```
*** Test cases ***
Verify Main Controller Redundancy Status Is 0

Verify Main Controller Role Is Active

Print Main Controller Field Bus Configuration

Reboot Main Controller

Verify From Reserve Controller Switch Over Status

Verify Reserve Controller Redundancy Fault Is 1

Verify Reserve Controller Redundancy Fault Indicator Is 1

Verify Reserve Controller Redundancy Fault Text

Verify Reserve Controller Role Is Active

Verify That I/O Units Are Visible On Reserve Field Bus
```

KUVA 12. Automaattisen kahdennustestauksen testitapaukset

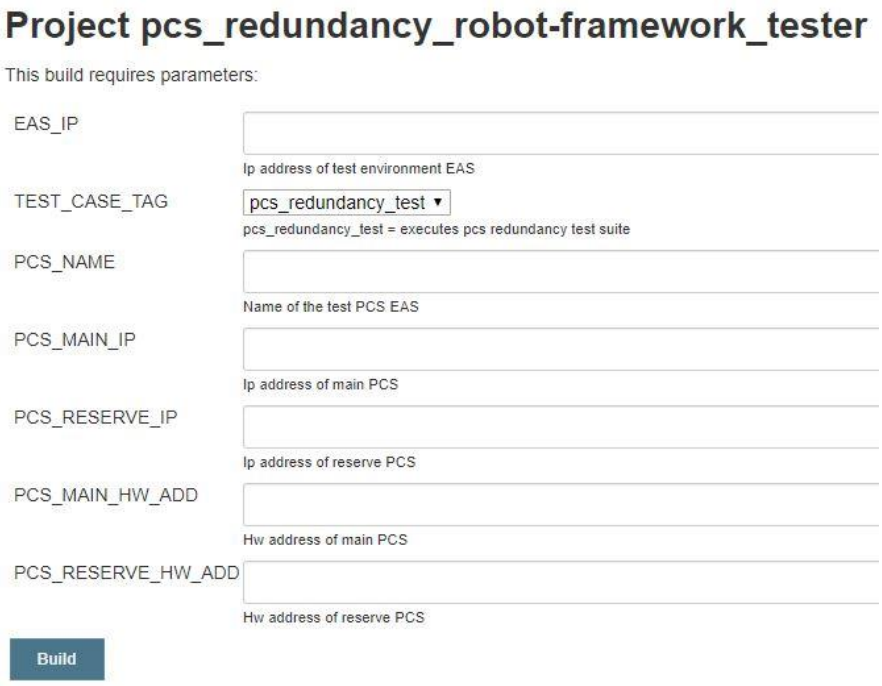
Seuraavaksi prosessiohjaimien kahdennukseen aiheutetaan vikatilanne hyödyntämällä kappaleessa 4.2 esitettyjä prosessiohjaimien vaihdon aiheuttavia tilanteita. Aktiiviseen prosessiohjaimeen otetaan TELNET-yhteys ja ohjain pakotetaan sammumaan komennolla (Reboot Main Controller). Tässä tilanteessa passiivinen prosessiohjain havaitsee vian ja se siirtyy ohjaamaan prosessia eli tapahtuu prosessiohjaimien vaihto. Tämän jälkeen Debuggerin avulla varmistetaan

diagnostiikasta, että vaihto on tapahtunut (Reserve Controller Role Is Active) ja tarvittavat hälytykset ovat päällä (Redundancy Fault, Fault Indicator, Fault Text). Viimeisenä tulostetaan I/O-konfiguraatio, jonka varalla oleva prosessiohjain havaitsee väylällä (I/O Units Are Visible On Reserve Field Bus) ja sitä verrataan aiemmin tulostettuun. Molempien I/O-konfiguraatioiden pitäisi olla identtiset eli molemmat prosessiohjaimet havaitsevat saman määrän I/O-laitteita väylällä.
Seuraavaksi odotetaan, että tilanne normalisoituu ja molemmat prosessiohjaimet ovat toiminnassa. Kun molemmat ohjaimet ovat taas toiminnassa toistetaan samat testausvaiheet vielä uudelleen, mutta tällä kertaa prosessiohjaimien roolit ovat päinvastaiset eli varalla oleva prosessiohjain toimii aktiivisena.

Automaattitestaus käynnistetään Jenkins-automaatiopalvelimen avulla, joka on avoimeen lähdekoodiin perustuva ohjelmistotuotannon automaatiopalvelin. Sen avulla yritysten on myös helpompi hallita ja kontrolloida eri prosesseja ohjelmistokehityksen elinkaaren ajalla. Näitä prosesseja ovat esimerkiksi ohjelmiston kääntäminen, testaaminen, dokumentointi ja julkaisu. (CloudBees, n.d.) Jenkins-automaatiopalvelin mahdollistaa nopean ja vakaan tavan integroida ohjelmistokehitysketjun kaikki työkalut yhteen. Sen suosio on kasvanut ja ominaisuudet ovat kehittyneet huomattavasti julkaisuvuodesta 2004 ja Jenkins onkin johtava avoimenlähdekoodin automaatiopalvelin. (Heller 2017.)

Jenkins-automaatiopalvelin toimii opinnäytetyössä myös automaattisen testauksen käyttöliittymänä (kuva 13). Käyttöliittymään on määritetty parametreja, jotka vaaditaan ennen kuin automaattitestaus voidaan suorittaa. Testauksen käynnistykseen vaadittavia parametreja ovat:

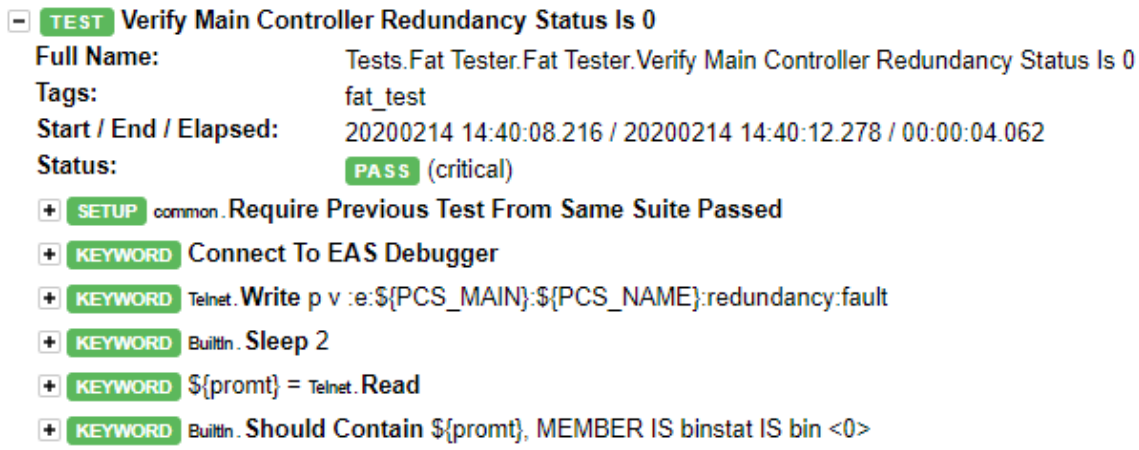
- Suunnittelupalvelimen IP-osoite (EAS_IP)
- Testikokonaisuus tai -tapaus (TEST_CASE_TAG)
- Prosessiohjaimien tunnus (PCS_NAME)
- Prosessiohjaimien IP-osoitteet (PCS_MAIN/RESERVE_IP)
- Prosessiohjaimien laiteosoitteet (PCS_MAIN/RESERVE_HW_ADD)



KUVA 13. Jenkins-käyttöympäristön parametrit

Parametrit ovat linkitetty RF:n ohjelmakoodiin, joten parametroinnin avulla voidaan testi helposti ja nopeasti toistaa seuraaville kahdennetuille prosessiohjaimille. Suuressa automaatiojärjestelmässä on useita kahdennettuja prosessiohjaimia ja kaikille tehdään samat kahdennustestaukset.

Testauksen suorituksen jälkeen Robot Framework luo käyttäjälle raportin, josta ilmenee testin tulokset. Raportti voidaan avata verkkoselaimella ja sen rakenne mukailee ohjelmakoodin rakennetta. Tämä helpottaa tulosten tulkintaa ja niitä ymmärtää henkilö, jolla ei ole ohjelmointiosaamista. Kuvassa 9 esitetyn testitapauksen tulokset on esitetty seuraavassa kuvassa (kuva 14).



KUVA 14. Esimerkki Robot Frameworkin testitapauksen tuloksista

Hyväksytyn testin tulokset esitetään vihreällä värillä ja hylätyn punaisella. Kuvan 14 testitulokset ilmaisevat, että testitapaus on suoritettu hyväksytysti. RF näyttää testikokonaisuuden kaikkien testitapausten tulokset samalla tyylillä kuin kuvan 14 testitapauksen tulokset. Koko testikokonaisuuden testitulos mukailee kuvassa 12 esitellyn testikokonaisuuden rakennetta, eikä testikokonaisuuden kaikkien testitapausten tulosten esittely näin ole oleellista. Kuvassa 15 on esitetty testikokonaisuuden yleiset tulokset, joista ilmenee testikokonaisuuden eli kahdennustestauksen suoritusaika, joka on 4 minuuttia ja 44 sekuntia. Testikokonaisuuden tuloksista ilmenee suoritusajan lisäksi testitapausten kokonaismäärä sekä hyväksyttyjen ja hylättyjen testitapausten lukumäärä.

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	20	20	0	00:04:43	
All Tests	20	20	0	00:04:43	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
noncritical (non-critical)	0	0	0	00:00:00	
pcs_redundancy_test	20	20	0	00:04:43	

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Tests	20	20	0	00:04:44	
Tests . Pcs Redundancy Tester	20	20	0	00:04:44	
Tests . Pcs Redundancy Tester . PCS redundancy tester	20	20	0	00:04:43	

KUVA 15. Kahdennustestauksen yleiset tulokset

5.4 Testaus

Automaattista kahdennustestausta ei päästy testaamaan työn aikana sen käyttötarkoituksen mukaisessa ympäristössä eli järjestelmätestauksessa. Tähän osasyynä oli Covid-19-viruksen aiheuttama poikkeustila, jonka vaikutuksesta järjestelmätestausta ei suoritettu normaaliin tapaan. Toinen syy oli se, että opinnäytetyössä kahdennustestauksen neljästä osuudesta automatisoitiin yksi osuus. Automaattisen kahdennustestauksen hyödyntäminen järjestelmätestauksessa vaatisi kaikkien neljän osuuden automatisoinnin.

Opinnäytetyön aikana suoritetun itsetestauksen perusteella voidaan kuitenkin todeta, että automatisoitu kahdennustestauksen osuus nopeuttaa järjestelmätestausta. Itsetestaus suoritettiin todellisessa ympäristössä ja kahdennetuilla prosessiohjaimilla. Itsetestauksessa käytetty ympäristö ei siis eroa toiminnan kannalta lopullisesta järjestelmätestauksen käyttöympäristöstä mitenkään. Suurin

aika kahdennustestauksessa kuluu, kun odotetaan sammutetun prosessiohjaimen palaavan normaaliin toimintatilaan. Tähän aikaan vaikuttaa merkittävästi prosessiohjaimella olevien sovelluspiirien ja muun tiedon määrä. Itsetestauksessa ei siis voitu vertailla automaattiseen kahdennustestaukseen kuluvaa aikaa eri prosessiohjaimilla.

Itsetestauksessa käytetyssä ympäristössä automaattiseen kahdennustestauksen suoritukseen kului noin viisi minuuttia. Automaattiseen testaukseen kulunut aika pysyy lähes vakiona jokaisella suorituskerralla. Vertailun vuoksi kahdennustestaus suoritettiin lisäksi käsin eli vanhalla testausmenetelmällä. Käsin tehtyyn testaukseen kului itsetestauksen perusteella noin kymmenen minuuttia. Käsin suoritetun testauksen aika saattaa vaihdella huomattavasti eri testauskertojen välillä. Tähän vaikuttaa esimerkiksi testauksen suorittava henkilö ja mahdolliset haittatekijät. Itsetestauksessa käsin tehdyn testauksen kymmenen minuutin aikaa voidaan pitää kuitenkin minimiaikana, joka siihen ainakin kuluu. Näiden tulosten perusteella voidaan arvioida automaattisen kahdennustestauksen vähintään puolittavan testaukseen kuluvan ajan. Tämä aikavähennys koskee siis kahdennustestauksen ensimmäistä osiota. Jäljellä olevien kolmen osuuden automatisoinnin ajansäästöä on vaikea arvioida ennen niiden automatisoinnin toteuttamista.

Automaattinen kahdennustestaus mahdollistaa myös integraatiotestaajan työajan hyödyntämisen muihin työtehtäviin testauksen ajaksi. Työn tarkemmat tulokset tulevat kuitenkin ilmi vasta, kun automaattista kahdennustestausta käytetään osana järjestelmätestausta. Silloin saadaan tietoa siitä, kuinka paljon automaattinen testaus keskimääriin nopeuttaa järjestelmätestausta ja mitkä ovat sen tuottamat taloudelliset hyödyt.

5.5 Jatkokehitys

Järjestelmätestauksen automatisointiin halutaan Valmetilla panostaa yhä enemmän ja tämän opinnäytetyön aikana tehtyä automaattista prosessiohjaimien kahdennustestausta kehittää edelleen. Seuraavana kehityksen vaiheena olisi automatisoida järjestelmätestausohjeen kahdennustestauksen jäljellä olevat kolme

osuutta. Näiden osuuksien automatisointiin vaaditaan tarkempaa perehtymistä verkkokytkimien hyödyntämiseen testauksessa. Verkkokytkimien avulla voitaisiin testata kahdennettujen prosessiohjaimien verkkoyhteyksiä katkomalla järjestelmä- ja I/O-väylän yhteyksiä. Järjestelmätestausohjeen muiden kohtien automatisointi on myös mahdollista RF-testausautomaatiokehyksen avulla. Näitä kohtia ovat muun muassa automaatiojärjestelmän prosessiohjaimien ja palvelintietokoneiden konfiguroinnin, ohjelmaversioiden ja lisenssien tarkistus.

Tässä opinnäytetyössä tehtyyn kahdennustestauksen automatisointiin voisi edelle mainittujen kolmen kohdan lisäksi kehittää myös lisäominaisuuksia. Kuvassa 13 esitetyn testauksen käyttöliittymän käsin suoritettavan parametroinnin voisi korvata automaattisemmalla toiminnolla. Nämä parametrit syötetään jo automaatiojärjestelmän suunnitteluvaiheessa järjestelmäkonfiguraation. RF-testausautomaatiokehyksen avulla nämä parametritiedot voitaisiin lukea automaattisesti järjestelmän konfiguraatiosta, mikä vähentäisi testaajan käsin tehtävää työtä.

Opinnäytetyössä luodun automaattisen kahdennustestauksen suoritukseen vaaditaan erillinen testauslaite eli kannettava tietokone, jossa ohjelmakoodi sijaitsee. Testilaitteella on oltava myös yhteys ja oikeudet Jenkins-automaatiopalvelimelle. Tätä voitaisiin jatkokehittää niin, että fyysinen testauslaite poistettaisiin kokonaan. Se voitaisiin korvata esimerkiksi sijoittamalla automaattinen testaussovellus pilvipalveluun, josta testauksen suoritus olisi vaivattomampaa eikä käyttöä rajoitettaisi vain yhteen fyysisen laitteeseen. Toinen vaihtoehto olisi integroida automaattinen kahdennustestaus suunnittelupalvelintietokoneen alustukseen. Alustuksessa tietokone konfiguroidaan Valmetin käyttötarkoituksen mukaan ja sinne asennetaan tarvittavat ohjelmat.

Kaikkien jatkokehitysehdotusten ja jo opinnäytetyön aikana luodun automaattisen kahdennustestauksen kohdalla on syytä ymmärtää, että ne vaativat jatkuvaa ylläpitoa toimivuuden takaamiseksi. Tämä tarkoittaa muun muassa testilaitteena toimivan kannettavan tietokoneen ja sen ohjelmien päivittämistä sekä niiden yhteensopivuuden varmistamista.

6 POHDINTA

Ennen opinnäytetyön alkua oli määriteltynä vain, että työn tarkoituksena oli automatisoida järjestelmätestauksen testausohjeen mukaista laitteistotestausta. Työn alettua määriteltiin työn sisältö tarkemmin ja se rajattiin prosessiohjaimien kahdennustestaukseen. Työn edetessä kuitenkin todettiin, ettei aika opinnäytetyön puitteissa riitä koko kahdennustestauksen toteuttamiseen. Tähän merkittävimpänä syynä oli, että kahdennettujen prosessiohjaimien järjestelmä- ja I/O-väylien automaattiseen testaamiseen vaadittaisiin huomattavan paljon enemmän aikaa ja osaamista, mitä opinnäytetyössä oli käytettävissä.

Työn tavoitteena oli nopeuttaa järjestelmän testaamista ja pienentää järjestelmätestauksen työkuormaa vähentämällä käsin tehtävää testausta. Järjestelmätestauksen testausohjeessa on neljä osuutta prosessiohjaimien kahdennustestaukseen, joista saatiin automatisoitua yksi osuus. Käsin tehtävää testausta saatiin näin vähennettyä, mutta ennen kaikkea työssä saatiin luotua käyttöympäristö ja pohja automaattisen testauksen toteutukseen. Testauspohjan ja käyttöympäristön luominen aiheuttikin haasteita, kun työn aikana toimittiin yhteistyössä kahden eri osaston kanssa. Toimituskeskus asetti automaattitestaukselle vaatimukset ja toiveet, jotka pyrittiin täyttämään yhteistyössä tuotekehitysosaston kanssa. Tuotekehityksellä oli myös omat näkemyksensä automaattitestauksen toteutukseen. Näiden toiveiden ja näkemysten yhteensaatto kulutti alkuvaiheessa paljon aikaa. Huolellinen käyttöympäristön ja testauspohjan suunnittelu oli kuitenkin tärkeä osa työtä, koska sillä on vaikutus automaattisen kahdennustestauksen käyttöön ja jatkokehitykseen.

Opinnäytetyössä luodun automaattisen kahdennustestauksen käyttöä ja testausta ei ehditty suorittaa sen oikeassa käyttöympäristössä eli järjestelmätestauksessa eikä tehdastestauksessa. Kun kahdennustestauksen kolme muuta osuutta saadaan lisättyä testauskokonaisuuteen, voidaan sen käyttöä testata ensin järjestelmätestauksessa ja myöhemmin osana tehdastestausta. Koska automatisoitua kahdennustestausta ei päästy käyttämään sen oikeassa käyttöympäristössä, on sen tuottamia kokonaistuloksia vaikea arvioida.

Työn aikana suoritettujen itsetestausten perusteella voidaan kuitenkin todeta kahdennustestauksen automatisoinnin vähentävän käsin tehtävää työtä ja näin nopeuttavan järjestelmätestausta. Itsetestauksen perusteella kahdennustestauksen ensimmäiseen osuuteen kuluva aika saatiin puolitettua. Tätä voidaan pitää merkittävänä kehityksenä, kun huomioidaan testauksen toistomäärät. Järjestelmä- ja tehdastestauksessa kahdennustestaus suoritetaan useille prosessiohjaimille, mistä syntyy moninkertainen vaikutus ajansäästöön.

Ajansäästön lisäksi automaattinen kahdennustestaus mahdollistaa testaajien työpanoksen hyödyntämisen muihin työtehtäviin automaattisen testauksen suorituksen ajaksi, mikä ei ollut mahdollista käsin tehdyn testauksen aikana. Työssä luodun automaattisen kahdennustestauksen tarkemmat tulokset ajansäästössä ja etenkin testauksen kustannuksissa tulevat ilmi, kun automaattista testausta hyödynnetään pidemmän aikaa järjestelmätestauksessa.

Jatkossa kahdennustestauksen kolme jäljellä olevaa osuutta automatisoidaan toimituskeskuksen testaajien toimesta. Automaattisen testauksen hyödyntäminen myös muissa järjestelmätestauksen osioissa on pohdinnan ja suunnittelun alla. Automaattista testausta voitaisiin hyödyntää järjestelmän laitteiden konfiguroinnin ja ohjelmistojen oikeellisuuden ja järjestelmän muiden laitteiden kommunikointiyhteyksien tarkastamiseen. Automaattisen kahdennustestauksen kehitystä jatketaan ja sen käyttö aloitetaan järjestelmätestauksessa, mutta tarkempaa ajankohtaa sille ei ole määritelty.

LÄHTEET

Control Point. N.d. Control point si Valmet. Luettu 20.3.2020. https://www.control-point.ro/proiectare-executie-instalatii-automatizare-valmet

Hass, A. M. J. 2008. Guide to advanced software testing. Norwood, Massachusetts: Artech House.

Heller, M. 2017. What is Jenkins?. Julkaistu 5.12.2017. Luettu 12.2.2020 https://www.infoworld.com/article/3239666/what-is-jenkins-the-ci-server-explained.html

Jokinen, S. Toimituskeskuksen esimies. 2020a. Haastattelu 19.3.2020. Haastattelija Mäkelä, S. Tampere. Valmet.

Jokinen, S. Toimituskeskuksen esimies. 2020b. Prosessiohjaimien kahdennus. Sähköpostiviesti. sampsa.jokinen@valmet.com. Luettu 19.3.2020

Kippo, A & Tikka, A. 2008. Automaatiotekniikan perusteet. Helsinki: Edita

Klärck, P. N.d. Experience. Luettu 17.1.2020. http://www.eliga.fi/experience.html

Lynch, G. 2013. Redundancy In Manufacturing Control Systems. Julkaistu 1.4.2013 Luettu 29.1.2020. https://www.manufacturing.net/home/article/13057107/redundancy-in-manufacturing-control-systems

Robot Framework User Guide version 3.1.2. 24.5.2019. Luettu 23.1.2020. http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html

Stemple, T. 2003. Redundancy In Automation. Julkaistu 7.2.2003. Luettu 16.3.2020 https://www.automation.com/en-us/articles/2003-1/redundancy-in-automation

SFS-EN 62381. 2012. Automation systems in the process industry. FAT, SAT, SIT. Helsinki: Suomen Standardoimisliitto SFS. Luettu 16.3.2020. Vaatii käyttöoikeuden. https://online.sfs.fi/

Strömman, M., Hirvonen, J., Hukki, K. & Tommila, T. 2007. Automaatiosuunnittelun prosessimalli. Yhteiset käsitteet verkottuneen suunnittelun perustana. Helsinki. Verkkojulkaisu 2010. www.automaatioseura.fi

Suominen, E. 2002. TCP/IP. Helsinki: Edita, IT Press

Tommila, T. 2001. Laatu automaatiossa. Parhaat käytännöt. Helsinki: Suomen Automaatioseura.

Valmet Oyj. 2019. Valmet Historia. Luettu 17.1.2019. https://www.valmet.com/fi/valmet-yrityksena/valmet-lyhyesti/historia/

Valmet Oyj. 2019. Valmet DNA. Luettu 3.2.2020.
https://www.valmet.com/automation/control-systems/valmet-dna/

Valmet Oyj. 2020. Valmet vuosijulkaisu 2019. Julkaistu 25.2.2020. Luettu 26.2.2020.
https://www.valmet.com/globalassets/investors/reports--presentations/annual-reports/2019/valmet-vuosikatsaus-2019.pdf

CloudBees. N.d. About Jenkins. Verkkosivu. Luettu 12.2.2020
https://www.cloudbees.com/jenkins/what-is-jenkins