Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Kasper Salo

Keittolaitteiden loppukoestusjärjes-telmä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

19.5.2019

Tekijä Otsikko Sivumäärä Aika	Kasper Salo Keittolaitteiden loppukoestusjärjestelmä 48 sivua + 3 liitettä 19.5.2019
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	sähkö- ja automaatiotekniikan koulutusohjelma
Ammatillinen pääaine	automaatiotekniikka
Ohjaajat	tuotannonkehityspäällikkö Risto Koskelainen lehtori Esko Tattari

Insinöörityön tavoitteena oli perehtyä sähkölaitteiden kappaletestaukseen, sekä määritellä ja toteuttaa uusi testausjärjestelmä tilaajayrityksessä valmistettavien laitteiden loppukoestukseen. Määrittelyn perustana käytettiin direktiivejä, standardeja, yrityksen omia laatudokumentteja ja eri osastojen edustajien näkemyksiä.

Työssä päädyttiin hyödyntämään olemassa olevaa testauslaitteistoa ja kehittämään selvitettyjen vaatimusten pohjalta uutta ohjelmistoratkaisua. Uuden ohjelmiston avulla testausjärjestelmän oli tarkoitus neuvoa käyttäjää suorittamaan vaaditut testit tarkasti määritetyllä tavalla. Sen oli kyettävä tunnistamaan huomattava määrä laitteista valmistettavia variantteja, sekä otettava huomioon varianttien välisten erojen vaikutus testaukseen. Ohjelmiston piti myös olla muokattavissa suorittamaan mittaukset toisella laitteistolla.

Järjestelmä päätettiin toteuttaa ratkaisulla, jossa yksittäiset, tarkasti määritellyt mittauksia ja tarkastuksia sisältävät testit kirjoitettiin erillisiksi pieniksi ohjelmikseen. Näitä testejä hallinnoitiin pääohjelmasta, johon oli määritelty testattavien laitteiden variantit, niille suoritettavat testit ja testeihin sovellettavat parametrit ja raja-arvot.

Työn tuloksena saatiin pääohjelman kaikki suunnitellut ominaisuudet implementoitua. Testeistä ehdittiin projektissa toteuttaa vain osa suunnitelluista.

Avainsanat	sähkölaitteet, kappaletestaus, testausautomaatio, tuotannon kehitys, ohjelmistokehitys

Author	Kasper Salo
Title	Appliance Testing System for Cooking Devices
Number of Pages	48 pages + 3 appendices
Date	19 May 2019
Degree	Bachelor of Engineering
Degree Programme	Electrical and automation engineering
Professional Major	Automation engineering
Instructors	Risto Koskelainen, Manager of Manufacturing Engineering Esko Tattari, Senior Lecturer

The goal of this work was to study appliance testing of electrical appliances and to define and implement a new test system for the verification process of equipment manufactured at the client's factory. The basis of the specification consisted of directives, standards, the client's own quality documents and opinions of employees representing various compartments in the client company.

The existing test hardware was discovered applicable to meet the standards, but the software controlling the process was outdated. Therefore, the project continued with software development for the existing hardware. With the aid of the new software, the system was intended to guide the user through the test process step by step. It was necessary for the system to be able to recognize a significant amount of different variants of the appliances produced and to determine which parameters, limit values and test procedures were applicable for each variant. In addition, the software had to be programmed in a way that would allow it to be transformed to function with entirely different hardware.

The problem was approached with a solution of dividing the system into small parts. Each test or inspection was conducted by a separate, hardware dependent program. These tests were managed from a main program which stored and handled the information about test procedures and parameters for each variant.

As a result, the main program was developed to functional condition with all the specified features implemented. Due to limitations of time, only some of the planned tests were implemented.

Keywords	electrical appliances, appliance testing, test automation, manufacturing engineering, software development

Sisällys

Lyhenteet

AD	Active Directory. Windows-käyttöjärjestelmien työkalu käyttäjien, laitteiden ja muiden resurssien hallintaa varten.
DNS	Domain Name System. Nimipalvelujärjestelmä, joka yhdistää tietokoneiden nimet IP-osoitteisiin.
DUT	Device Under Test. Testattava laite.
FPY	First Pass Yield. Suhteellinen osuus prosessin ensimmäisellä yrityksellä läpäisevistä tuotoksista kaikkiin prosessiin syötettyihin.
IPC	Inter Process Communication. Tietokoneessa suoritettavien sovellusten välinen tietoliikenne.
SQL	Structured Query Language. Komentokieli, jolla relaatiotietokannan tietoa haetaan, luodaan ja muokataan.
TCP	Transmission Control Protocol. Tietoliikenneprotokolla, jonka kautta tietokoneet ja sovellukset voivat lähettää toisilleen dataa.
XML	Extensible Markup Language. Rakenteellisen tiedon jäsentelyyn tarkoitettu merkintäkielen standardi.

1 Johdanto

Työ tehdään Metos Oy Ab:lle, joka on ammattikeittiölaitteita ja -kalusteita toimittava yritys. Ammattikeittiölaitteiden valmistajana Metos on toiminut jo yli 95 vuoden ajan. Laitevalmistuksen lisäksi yrityksen palveluihin kuuluu keittiösuunnittelu, laitteiden käyttökoulutus ja huoltopalvelu. Alkujaan suomalaisella Metoksella on nykyään toimintaa yhdeksässä eri maassa noin, 750 työntekijän voimin. Omia tehtaita Metoksella on Suomessa kaksi, Sorsakoskella ja Keravalla, sekä kolmas Virossa Tallinnassa. [1.]

Pääkonttorin yhteydessä Keravan tehtaalla toimiva keittolaitelinja valmistaa Metos Oy Ab:n keittopadat sekä painekeittokaapit. Niistä padat ovat yrityksen tärkein omaa tuotantoa oleva laiteperhe ja Proveno kaikista kehittynein malli. [1.]

Koestus on prosessi, jolla varmistetaan, että tuotantolinjalta valmistuvat laitteet täyttävät kaikkien sidosryhmien vaatimukset, sekä suoritetaan tehtaan sisäistä laadunvalvontaa. Koestus perustuu sähköisiin mittauksiin ja toiminnan analysoimiseen erilaisissa käyttö- ja vikatilanteissa, sekä silmämääräisiin tarkistuksiin. Samalla tarkastetaan aikaisemmat työvaiheet. Koestuksessa laite käynnistetään ensimmäisen kerran, joten sen aikana suoritetaan myös tarvittavat ohjelmistojen asennukset, parametrisoinnit, kalibroinnit ja käyttäjäkohtaisten esiasetusten valinta.

Työn tarkoituksena oli perehtyä nykyiseen testausjärjestelmään, suunnitella ja toteuttaa siihen tarvittavat muutokset. Muutosten tavoitteena oli tehdä järjestelmästä entistä luotettavampi, helppokäyttöisempi ja ylläpidettävämpi, sekä automatisoida järjestelmän tiedonkeruu tuotannonkehityksen tarpeisiin. Uudesta järjestelmästä haluttiin havainnollinen ja käyttäjää ilman erillisiä käyttöohjeita opastava. Tarvittaessa järjestelmän tulisi myös skaalautua uusille tuotteille ja mittalaitteille.

Samalla tarkistettiin, että loppukoestus täyttää osaltaan valmistettavien laitteiden soveltamisalaan kuuluvien direktiivien ja harmonisoitujen standardien vaatimukset. Standardeista on ilmestynyt uudet versiot nykyisen koestusjärjestelmän toimittamisen jälkeen, mutta niissä ei oleteta olevan suuria muutoksia. Tästä syystä työssä keskitytään ensisijaisesti uusien ominaisuuksien ideointiin ja kehitykseen.

Tavoitteena on toteuttaa järjestelmä pisteeseen, jossa sillä on mahdollista koestaa yksi tehtaan tuotteista alusta loppuun ja raportteina saadaan ulos laitteen koestuspöytäkirja, sekä tyypillisiä laatua mittaavia raportteja. Esiversiossa testattavaksi tuotteeksi on valittu sähkökallisteinen kombipata Proveno 2G, joka sisältää myös sekoitus- ja jäähdytystoiminnot, sekä automaattisen vedentäytön. Muiden tuotteiden tuominen järjestelmään, järjestelmän käyttöönotto, sekä ylösajo on rajattu pois työstä.

Työ tehdään, koska käytössä oleva koestusjärjestelmä on ominaisuuksiltaan vanhanaikainen. Sen käyttömukavuus on huono ja skaalautuvuus uusille tuotteille heikolla tasolla. Sen tehokas käyttö edellyttää järjestelmän tuntemista ja monen asian muistamista, sekä ulkoa opettelua. Näistä syistä koestuksesta saadut tulokset ovat alttiita virheille ja vaihtuvien käyttäjien erilaisille tottumuksille.

Koestuksen aikana saadaan merkittävää tietoa valmistetuista laitteista. Uuden järjestelmän kehitys antaa vapaat kädet määritellä mitä tietoja tallennetaan. Tiedonkeruun avulla on tarkoitus valjastaa koestus tuottamaan raportteja tuotannon ongelmista, sekä kehittää tuotteiden jäljitettävyyttä.

2 Loppukoestus lainsäädännössä

Lainsäädännön vaatimusten selvittämisessä käytettiin apuna laitteiden suunnitteludokumentteja, joista selvisi minkä direktiivien soveltamisalaan laitteet kuuluvat. Vaatimustenmukaisuusvakuutuksen [2] mukaan linjalla valmistettavat laitteet ovat konedirektiivin 2006/24/EY, pienjännitedirektiivi 2014/35/EU, painelaitedirektiivin 2014/68/EU, sekä EMC direktiivin 2014/30/EU alaisia.

Vaatimustenmukaisuus, joka tarkoittaa laitetta koskevien direktiivin noudattamista, voidaan osoittaa täytetyiksi usealla menetelmällä, joista helpoin on yleensä soveltamisalaan kuuluvan standardin noudattaminen [3, s. 6]. Koestuksessa suoritettavat testit ja muut toimenpiteet määriteltiin soveltamisalaan kuuluvien direktiivien mukaan noudattaen samoja yhdenmukaistettuja standardeja, joita oli käytetty jo laitteen suunnittelussa. Tapauksissa, joissa oli käytetty direktiivin täyttämiseksi muuta menetelmää kuin standardia, tutkittiin itse direktiiviä.

2.1 Painelaitteita koskevat vaatimukset

Painelaitteita koskevat vaatimukset määritetään painelaitedirektiivissä ja ne on toteutettu vaatimuksenmukaisuuden arviointimenettelyn moduulien B (suunnittelutyyppi) + D mukaan [2]. Moduuli B käsittelee laitteen suunnittelua. Koestuksessa tulee huomioida tuotantoprosessin laadunvalvontaa käsittelevä moduuli D [4, s. 228], jossa vaaditaan direktiivin liitteen I kohdan 3.2 noudattamista:

> **3.2 Lopputarkastus**
>
> Painelaitteille on tehtävä lopputarkastus jäljempänä tarkoitetulla tavalla.
>
> *3.2.1 Loppukoe*
>
> Painelaitteille on tehtävä loppukoe, jonka tarkoituksena on silmämääräisesti ja liiteasiakirjojen tarkastuksella varmistaa tämän direktiivin vaatimusten noudattaminen. Valmistusvaiheessa tehdyt tarkastukset voidaan tässä tapauksessa ottaa huomioon. Jos turvallisuussyistä on tarpeen, loppukoe on tehtävä laitteen kaikille osille sisä- ja ulkopuolisesti tarvittaessa valmistusvaiheen aikana (esimerkiksi jos tarkastus ei enää loppukoevaiheessa ole mahdollinen).
>
> *3.2.2 Koeponnistus*
>
> Painelaitteiden lopputarkastukseen on kuuluttava painekoe, joka tavallisesti tehdään nestepainekokeena vähintään 7.4 kohdassa säädetyllä paineella, jos tämä on tarkoituksenmukaista.
>
> Luokan I sarjavalmistetuille laitteille tämä koe voidaan tehdä tilastollisin perustein.
>
> Jos nestepainekoe on vahingollinen tai sitä ei voida suorittaa, voidaan toteuttaa muita hyväksyttyjä kokeita. Muiden kuin nestepainekokeen osalta on toteutettava täydentäviä toimenpiteitä kuten ainetta rikkomattomia tarkastuksia tai muita tehokkuudeltaan vastaavia toimenpiteitä ennen kokeiden suorittamista.
>
> *3.2.3 Varolaitteiden tarkastus*
>
> Laitekokonaisuuksien lopputarkastukseen kuuluu myös varolaitteiden tarkastus, jonka tarkoituksena on varmistua siitä, että 2.10 kohdassa tarkoitettuja vaatimuksia noudatetaan kaikilta osin. [4, s. 208–209.]

Koeponnistus suoritetaan jo aikaisemmassa valmistusvaiheessa, joten se jää loppukoestuksesta pois. Valmistettavissa laitteissa direktiivin tarkoittamia varolaitteita ovat varoventtiili, sekä lämmityskontaktoreilta ylipainetilanteessa ohjauksen katkaiseva painekytkin.

Painelaitedirektiivin moduuli D [4, s. 228–231] edellyttää, että laatujärjestelmä hyväksytetään ilmoitetulla laitoksella, sekä ilmoitettava kaikista laatujärjestelmään suunnitelluista

muutoksista. Lisäksi ilmoitetun laitoksen on tehtävä auditointeja, jotka johtavat laatujärjestelmän täydelliseen uudelleenarvioimiseen joka kolmas vuosi. Koska nykyistä laatujärjestelmää on säännöllisesti arvioitu ilmoitetun laitoksen toimesta, oletetaan, että se täyttää direktiivin vaatimukset ja voidaan pitää sitä minimivaatimuksena uutta järjestelmää kehitettäessä.

2.2 Sähkölaitteita koskevat vaatimukset

Kone- ja pienjännitedirektiivien vaatima suojaus sähkön aiheuttamilta vaaroilta on täytetty noudattamalla yhdenmukaistettua koneiden sähköjärjestelmiä koskevaa b-luokan standardia EN 60204-1 [5]. Tämän b-luokan standardin lisäksi on olemassa myös yhdenmukaistettu keittopatojen erityisvaatimukset määrittävä c-tyypin standardi EN 60335-2-47, jota käytetään yhdessä sähkölaitteiden turvallisuusvaatimukset määrittävän EN 60335-1 standardin kanssa. Molemmat standardit ovat yhdenmukaistettuja sekä kone- että pienjännitedirektiivin alaisuuteen. [6, s. 77; 7, s. 35.]

Laivoihin asennettavissa laitteissa tulee lisäksi noudattaa tilaajan valitseman luokituslaitoksen vaatimuksia. Usein vaatimukset pohjautuvat IEC-standardeihin sisältäen yleensä joitakin tiukennuksia [8, s. 51]. Luokituslaitokset voivat myös vaatia esimerkiksi testausraporttien liittämistä laitteen dokumentaatioon [9, s. 1170.]

Direktiiveissä ei suoraan vaadita laitteiden kappaletestausta, mutta niissä vaaditaan mahdollisten testiraporttien säilyttämistä [10, s. 71; 11, s. 370]. Kappaletestausta vaaditaan kuitenkin molemmissa soveltamisalaan kuuluvissa standardeissa. Koneiden sähköjärjestelmiä käsittelevän EN 60204-1:n mukaan testaus tulisi suorittaa tuote- eli c-tyypin standardin mukaan aina kun laitteelle on sellainen olemassa [12, s. 158]. Kuitenkin tuotestandardi EN 60335-1 sallii testien tekemisen toisilla menetelmillä, kunhan turvallisuustaso pysyy samana [13, s. 103.]

Standardeja verrattiin toisiinsa sekä suoritettavien testien laajuuden, että testausmenetelmien perusteella. Vertailussa huomioitiin vain jännitealueella 200 V–480 V toimivat peruseristetyt laitteet ja sen pohjalta määriteltiin sarja testejä, jotka varmistavat molempien standardien täyttymisen. Vaaditut testit ja niiden vastaavuus on esitetty taulukossa 1. Varustamoille suunnattujen laitteiden lisävaatimukset perustuvat Lloyd’s Register -

luokituslaitoksen yleisiin vaatimuksiin. Muut luokituslaitokset jätettiin huomioimatta, koska niiden vaatimuksia ei ollut saatavilla. Näin ollen Marine-vaatimuksia ei suoraan voi soveltaa, vaan ne on kysyttävä tilaajalta tapauskohtaisesti. Ajatus yhden varustamon vaatimuksien tarkastelun takana on tietoisuuden lisääminen mahdollisista erityisehdoista.

Taulukko 1. Soveltamisalaan kuuluvissa standardeissa määritellyt kappaletestit ja niiden vastaavuus.

SFS-EN 60204-1	**SFS-EN 60335-1**	**Lloyd's Register yleiset vaatimukset**
Todennetaan, että sähkölaitteisto on teknillisen dokumentaation mukainen		
Syötön automaattisella poiskytkennällä toteutetun suojauksen ehtojen todentaminen	Maadoitusyhteyden jatkuvuus	
Eristysresistanssimittaus*		Eristysvastusmittaus
Jännitekoe*	Sähkölujuustesti	Jännitekoe
Suojaus jäännösjännitteiltä*		
Toimintakokeet	Toiminnallinen testi	
* Todentamisen ei välttämättä tarvitse sisältää kyseistä kohtaa, ellei tuotestandardissa toisin vaadita.		

Taulukon 1 testit on lueteltu siinä järjestyksessä missä ne suositellaan tehtäväksi standardissa EN 60204-1 [12, s. 158]. Marine-laitteille tehtävä eristysvastusmittaus tulee kuitenkin suorittaa välittömästi jännitekokeen jälkeen [9, s. 1170]. Muilta osin testauksessa ei ole tarvetta poiketa taulukon järjestyksestä.

2.2.1 Maadoitusyhteyden jatkuvuus

Maadoitusyhteyden jatkuvuustesti varmistaa, että laitteen runkoon esimerkiksi eristyksen pettäessä kohdistuva jännite purkautuu turvallisesti suojamaapotentiaaliin. Standardien EN 60335-1 ja EN 60204 eroina maadoitusyhteyden jatkuvuustestissä on, että ensiksi mainitussa vaaditaan käyttämään korkeampaa, vähintään 10 A:n koestusvirtaa ja jännitelähteen kuormittamattomaksi jännitearvoksi on määritelty korkeintaan 12 V [13, s. 103]. Jälkimmäisessä virta on määritelty 0,2 A:n ja 10 A:n välille ja jännitteen kuormittamattomaksi arvoksi 24 V [12, s. 160]. Tuotestandardin korkeampi virta tuottaa testissä tarkemman tuloksen. Siinä testi on kuvattu seuraavasti:

Teholähteestä, jonka jännite kuormittamattomana on enintään 12 V (vaihtosähkö tai tasasähkö), syötetään vähintään 10 A suuruinen virta jokaisen **kosketeltavan maadoitetun metalliosan** ja:
— maadoitusliittimen välille
— muilla **suojausluokan I laitteilla**
• maadoituskoskettimen tai pistotulpan maadoituskoskettimen välille
• kojevastakkeen maadoituskoskettimen välille.
Jännitehäviö mitataan ja sen pohjalta lasketaan resistanssi, joka ei saa ylittää seuraavia arvoja
— **verkkoliitäntäjohdolla** varustetuilla laitteilla 0,2 Ω, tai 0,1 Ω plus **verkkoliitäntäjohdon** resistanssi
— muilla laitteilla 0,1 Ω.
HUOM. 1 Testiä jatketaan ainoastaan niin kauan kuin on tarpeellista jännitehäviön mittaamiseksi.
HUOM. 2 On kiinnitettävä huomiota, ettei mittauskoettimen ja testattavan metalliosan välinen kosketusresistanssi vaikuta testituloksiin. [13, s. 103.]

2.2.2 Sähkölujuus

Sähkölujuustestissä varmistetaan, että laitteen jännitteellisistä osista ei voi tapahtua läpilyöntiä runkoon, mikä altistaisi käyttäjän sähköiskulle. Se suoritetaan sovellettavalla jännitealueella samalla tavalla sekä tuote- että koneturvallisuusstandardissa. Standardeissa ainoa ero on läpilyönnin raja-arvo, joka tuotestandardissa on määritetty 30 mA:iin. Koneturvallisuusstandardissa raja-arvoa ei määritellä ollenkaan. Tuotestandardin mukaan mittaukset suoritetaan seuraavasti:

Laitteen eristykseen annetaan vaikuttaa 1 s ajan käytännöllisesti katsoen sinimuotoisen vaihtojännitteen, jonka taajuus on 50 Hz tai 60 Hz. Testijännitteen suuruus ja sen vaikutuskohdat esitetään taulukossa A.1.

Taulukko A.1 Testijännitteet

Kohdistuspisteet	*Testijännite V*		
	suojausluokan I laitteet ja suojausluokan II laitteet		*Suojausluokan III laitteet*
	Mitoitusjännite		
	≤ 150 V	*> 150 V*	
Jännitteisten osien ja jännitteisistä osista *• ainoastaan peruseristyksellä erotettujen kosketeltavien metalliosien välille*	*800*	*1000*	*400*
• kaksois- tai vahvistetulla eristyksellä erotettujen kosketeltavien metalliosien välille[a, b]	*2000*	*2500*	*-*
a <CENELECin poistama huomautus> *b tätä testiä ei tehdä osille, jotka ovat suojausluokan II rakennetta, mikäli testiä ei pidetä tarpeellisena.*			

HUOM. 1 On varmistettava, että sellaisilla laitteilla, joita käytetään testin aikana, testijännite kohdistuu oikeaan paikkaan, esimerkiksi releellä ohjattavat lämmityselementit.
Läpilyöntiä ei saa tapahtua. Läpilyönti katsotaan tapahtuneeksi, kun testipiirin virta ylittää 5 mA. Kuitenkin tämä raja-arvo korotetaan arvoon 30 mA sellaisilla laitteilla, joilla on suuret vuotovirrat.
HUOM. 2 Testipiirin sisältyy virran tunnistava laite, joka laukaisee, kun virta ylittää raja-arvon.
HUOM. 3 Suurjännitemuuntajan on kyettävä ylläpitämään tietty jännite virran raja-arvolla.
HUOM. 4 Vaihtojännitteen sijasta eristykseen voidaan kohdistaa myös tasajännite, jonka suuruus on 1,5 kertaa taulukossa esitetty jännite. Vaihtosähkö, jonka taajuus on enintään 5 Hz, katsotaan tasasähköksi. [13, s. 103–104.]

Vuotovirta voi testissä nousta huomattavaksi, jos testattavassa laitteessa on kapasitanssia jännitteisten osien ja suojamaan välillä. Näissä tapauksissa laite olisi syytä koestaa tasajännitteellä. Esimerkiksi EMC suotimet ja taajuusmuuttajat voivat aiheuttaa suuren vuotovirran, koska niissä osa häiriöstä johdetaan tyypillisesti Y-kondensaattorin kautta maahan.

Asiakas, joka asentaa laitteet laivaan, saattaa vaatia sähkölujuustestin suorittamista normaalia suuremmalla koestusjännitteellä. Lloyd's Register -luokituslaitoksen yleisissä vaatimuksissa [9, s. 1170] koestusjännite lasketaan kaavalla

$$2 \times U_n + 1000\,V. \qquad (1)$$

Kaavassa 1 U_n on laitteen nimellisjännite, ja sen tulee olla taajuudeltaan 25–100 Hz vaihtojännitettä. Testi aloitetaan jännitteellä, joka on 1/3 mitoitusjännitteestä, ja testausjännitettä nostetaan kohti huippuarvoa niin nopeasti, kuin on testin kannalta soveltuvaa. Huippuarvossa jännitettä pidetään yhden minuutin ajan, minkä jälkeen jännite lasketaan takaisin 1/3 osaan testausjännitteestä. Testi katsotaan läpäistyksi, jos odottamattomia purkauksia vuotovirrassa ei ilmene. Vuotovirralle ei ole määritetty tarkempaa raja-arvoa. [9, s. 1170.]

Laivoissa käytetään tyypillisesti jännitteitä väliltä 380 V–480 V [6, s. 64], joten testi on kyettävä suorittamaan jännitteillä väliltä 1000 V–1960 V. Tasajännitteen käyttö koestuksessa ei näissä tapauksissa ole sallittua [9, s. 1170]. Siitä ei kuitenkaan ole haittaa, koska laivoille ei yleensä haluta asentaa taajuusmuuttajan ja suotimen sisältäviä laitteita niiden aiheuttamien vuotovirtojen ja häiriöiden takia.

2.2.3 Eristysvastusmittaus

Eristysvastusmittaus on välttämätön vain laivoihin asennettaville laitteille, joille se suoritetaan välittömästi jännitekokeen jälkeen [9, s. 1170]. Testin suorittaminen on ajallisesti nopeaa, joten sen tekemättä jättämisestä muille laitteille ei ole hyötyä. Turvallisuutta voi pitää tärkeämpänä, kuin muutamien sekuntien säästöä ajassa. Koneturvallisuusstandardi SFS-EN 60204-1:n mukaan testi suoritetaan seuraavasti:

> **18.3 Eristysresistanssimittaus**
>
> Pääpiirien johtimien ja suojajohdinpiirin välisen eristysresistanssin on oltava vähintään 1 MΩ mitattuna 500 V tasajännitteellä. Mittaus voidaan tehdä koko asennuksen yksittäisille ryhmille.
>
> Poikkeus: Laitteiston tietyille osille, sisältäen esim. kiskostoja, laahauskisko- ja laahausjohtojärjestelmiä tai liukurenkaita, sallitaan pienempiä vähimmäisarvoja, joka on kuitenkin vähintään 50 kΩ.
>
> Jos koneen sähkölaitteistossa on ylijännitesuojia, jotka todennäköisesti toimivat mittauksen aikana, sallitaan joko:
>
> niiden irrottaminen, tai
>
> mittausjännitteen alentaminen ylijännitesuojan toiminta-arvoa pienemmäksi, mutta ei syöttöjännitteen (vaiheen ja nollan välistä) huippuarvoa pienemmäksi. [12, s. 164.]

Joissakin tapauksissa, esimerkiksi IT-jakelujärjestelmää käyttävillä laivoilla, voidaan laitteelta vaatia korkeampaa eristysresistanssia. Näissä tapauksissa käytetään raja-arvona yleisesti 100 MΩ. [8, s. 64.]

2.2.4 Toiminnallinen testi

Standardien edellyttämässä toiminnallisessa testissä laitteen toimintoja tarkastellaan vain turvallisuusnäkökulmasta. Tuotestandardin EN 60335-1 mukaan sellaiset toiminnot, jotka väärin kytkettynä tai säädettynä voivat aiheuttaa turvallisuusriskin, on testattava asianmukaisesti [13, s. 104]. Proveno 2G-padasta tehdyn riskien arvioinnin [5] mukaan tällaisia toimintoja ovat:

- ohjelmallinen lämmityksen katkaisu suurimmassa toimintalämpötilassa
- pakotettu lämmityksen katkaisu paineen hälytysrajalla
- varoventtiilin laukeaminen viimeistään paineastiahyväksynnän mukaisella suurimmalla käyttöpaineella

- sekoittimen suojaaminen turvaritilällä
- sekoitusnopeuden rajoitus, kahden käden käyttö ja jatkuvaa painallusta edellyttävä toiminta turvaritilän ollessa auki
- jatkuvaa painallusta edellyttävä kallistus.

3 Nykyinen koestusjärjestelmä

Työssä perehdyttiin käytössä olleeseen koestusjärjestelmään sen mukana toimitetun dokumentaation avulla. Kävi ilmi, että järjestelmän laitteisto täyttää kaikki voimassa olevat direktiivit, sekä on riittävän hyvin dokumentoitu ja tuettu, että sen hyödyntäminen on mahdollista jatkossakin. Tuella tarkoitetaan, että kaikille mittalaitteille on saatavilla ajuri uusille Windows-käyttöjärjestelmille ja että rikkoutuneiden laitteiden tilalle on saatavilla uusia.

Testausjärjestelmän ohjelmiston lähdekoodia käytettiin apuna järjestelmän toiminnan tutkimiseen. Toiminnasta haluttiin saada kokonaiskuva uuden järjestelmän vaatimusten minimilähtökohdaksi. Ohjelmistosta itsestään ei löytynyt mitään uudelleenkäytettävää osuutta.

3.1 Laitteisto

Mittaukset ja muut toiminnot toteutetaan taulukon 2 mukaisilla sähköturvallisuus-testereillä. Niiden laitteistoon kuuluu mittalaitteita, analogisia antureita, erikokoisia pistorasioita testattavan laitteen kytkemiseen sekä mittalaitteet pistorasiaan yhdistäviä kytkentälaitteita.

Taulukko 2. Järjestelmään kuuluvat laitteet ja niiden tehtävät.

Toiminto	**Laite**
Jännitekoe	Finero FST-110
Eristysresistanssin mittaus	Finero FST-120
Suojajohtimen jatkuvuuden mittaus	Finero FST-130
Käytönaikaisen vuotovirran mittaus	Finero FST-140
Virta- ja tehomittaukset	Elcontrol Energy Net VIP-Energy
Mittalaitteiden, käyttöjännitteen ja pistorasioiden kytkentä, analogiset tulot	Finero FST-150

Mittalaitteet ovat tietyn, tarkoin määritellyn testin suorittamista varten. Relekortilla puolestaan hallitaan sitä, mikä mittalaite ja pistorasia ovat kulloinkin kytkettyinä. Analogisista tuloista on käytetty vain paineanturi, jolla mitataan painelaitteen höyrynpainetta eri käyttötilanteissa. Järjestelmä sisältää myös mahdollisuuden kytkeä testattava laite pistorasian kautta verkkojännitteeseen toiminnallisten testien suorittamista varten.

Sekä energia-analysaattori että FST-sarjan laitteet sisältävät etäohjauksen, joka on testauksen automatisoinnissa välttämätön ominaisuus. Etäohjaus edellyttää, että laitteet on liitetty tietokoneeseen. FST-sarjan laitteet on liitetty väylällä toisiinsa ja niistä yhteen on asennettu RS232-, eli sarjaporttimoduuli. Sen kautta kaikki FST-sarjan laitteet ovat etäohjattavissa samalla sarjaporttiyhteydellä. [14, s.15.]

Energia-analysaattorin etäohjaus perustuu Modbus-protokollaan. Modbus-väylän liikenne on sarjamuotoista ja energia-analysaattorin tapauksessa väylän fyysinen kerros on RS485:n mukainen. Kommunikointi analysaattorin kanssa tapahtuu tietokoneen laajennuspaikkaan asennetun RS485-ohjaimen kautta, joka tietokoneen käyttöjärjestelmälle näkyy sarjaporttina. Modbus-protokollan ylemmät kerrokset on toteutettu energia-analysaattorin ajurissa. [15, s.22.]

4 Uusien ominaisuuksien määrittely

Uudessa järjestelmässä painotettiin prosessin standardisointia ja inhimillisten tekijöiden vaikutuksen poissulkemista lopputuloksesta. Muita tavoitteita olivat käytön helppous, sekä jäljitettävyys ja informaation kerääminen laadunhallintaa varten. Jäljitettävyydellä tarkoitetaan, että kaikki koestuksen aikana kerätty informaatio on myöhemmin saatavissa, eikä uudelleen suoritettu prosessi ylikirjoita edellisillä kerroilla saatuja tuloksia.

4.1 Käyttöoikeudet

Järjestelmään määriteltiin käyttäjätasot perustoimintoja, edistyneitä toimintoja ja ylläpitotoimintoja varten. Edistynyt, niin kutsuttu tehokäyttäjä, siltä varalta, että joitakin laitteita toimitetaan asiakkaan tilauksesta testattuna tiukemmilla raja-arvoilla kuin soveltamisalaan kuuluvat standardit edellyttävät ja arvoja tulisi muokata koestuksen yhteydessä.

Ominaisuutta ei kuitenkaan haluttu avata tavallisille käyttäjille, ettei arvoja muokattaisi luvatta. Tavallisilta käyttäjiltä haluttiin myös piilottaa sellaista tietoa, jota he eivät välttämättä tarvitse, minkä tarkoitus on myös hankaloittaa tahallista mittaustulosten vääristämistä.

Taulukko 3. Käyttäjätasojen oikeudet.

Käyttäjätaso	**Parametrien ja raja-arvojen näyttö**	**Parametrien ja raja-arvojen muokkaus**	**Asetusten ja järjestelmän konfigurointi**
Ylläpitäjä	Näkee kaikki parametrit	Saa muokata kaikkia arvoja	Sallittu
Tehokäyttäjä	Näkee kaikki parametrit	Kun muokkaus on erikseen sallittu	Estetty
Käyttäjä	Näkee parametrit, jotka on asetettu näkyväksi	Estetty	Estetty
Muu *	Estetty	Estetty	Estetty
* Käyttäjiltä, jotka eivät kuulu mihinkään ylläolevista ryhmistä, on estetty kaikki toiminnot.			

Käytännössä tehokäyttäjät-ryhmään on tarkoitettu kuuluvaksi laitteiden suunnittelijat. Ryhmään pääseminen edellyttää riittävän tietotaidon omaamista testattavista laitteista ja niitä koskevista määräyksistä siten, ettei parametreihin tule tehtyä muutoksia, jotka voisivat heikentää asiakkaan käyttökokemusta tai laitteen turvallisuutta. Raja-arvoista poikkeaminen voi myös vaatia dokumentaatiota, jonka laatiminen on suunnitteluosaston tehtävä.

4.2 Raportointi ja laadunhallinta

Kaikki järjestelmän keräämä informaatio päätettiin tallentaa tietokantaan, mihin jää jälki jokaisesta tehdystä toimenpiteestä, eikä vanhaa tietoa päällekirjoiteta. Kerätyn informaation perusteella voidaan mitata tuotantolinjan laatua ja tutkia minkälaisia laatupoikkeamia tuotteissa esiintyy. Se mahdollistaa myös laitteen läpikäymän testausprosessin ja siihen tuotantoprosessin aikana tehtyjen korjausten jäljitettävyyden.

Metos Oy Ab noudattaa laadunhallintajärjestelmiä käsittelevää ISO 9001 -standardia ja yrityksellä on siitä voimassa oleva kolmannen osapuolen sertifikaatti. Standardi käsittelee laatuun vaikuttamista organisaation johtamisen kannalta. Työssä huomioitiin standardin vaatimukset etsimällä siitä sellaisia kohtia, joiden täyttämistä voisi suoraan edesauttaa loppukoestuksen tietojärjestelmällä.

Koestusraportti

Koestusraportilla osoitetaan, että tuote on läpäissyt organisaation sille asettamat hyväksyttämiskriteerit ja se kelpaa luovutettavaksi asiakkaalle. Hyväksyttämiskriteerien täytetyksi näyttämisen lisäksi laadunhallintajärjestelmä [16, s. 26] vaatii, että raportissa tulee käydä ilmi henkilö, joka tuotteen on todennut hyväksytyksi. Hyväksyttämiskriteerit, jotka määriteltiin kirjattavaksi raporttiin, olivat

- kuvaus suoritetuista testeistä
- testeissä käytetyt parametrit ja raja-arvot
- mittaustulokset
- tieto testin läpäisystä.

Vaikka koestusraportin pystyy koostamaan tietokantaan tallennetusta datasta, päätettiin se lisäksi tallentaa tekstimuotoisena verkkolevylle arkistoitavaksi. Tämä mahdollistaa tuotannon jatkamisen tietokannan mahdollisten ongelmatilanteiden ja huoltokatkosten aikana, ja varmistaa tärkeimmän tiedon säilymisen, joka tässä tapauksessa on hyväksyttämiskriteerien täyttyminen. Tekstimuotoinen raportti luodaan jokaisen onnistuneen yrityksen jälkeen ja siihen tallennetaan vain lopulliset tulokset, koska järjestelmän käyttäminen ilman tietokantaa on poikkeuksellista, eikä siksi ole tarkoituksen mukaista kahdentaa kaikkia tietokantaan liittyviä toimintoja.

Digitalisoitu poikkeama- ja korjausraportti

Järjestelmään määriteltiin poikkeama- ja korjausraportit, jotka suunniteltiin noudattamaan laadunhallintajärjestelmän vaatimuksia poikkeaviin tuotoksiin liittyvän informaation käsittelystä. Laatujärjestelmän [16, s. 27] mukaan poikkeavista tuotoksista tulee dokumentoida poikkeaman kuvaus, korjaustoimenpiteet, sekä mahdolliset poikkeusluvat ja niiden myöntäjä, jos tuote toimitetaan organisaation hyväksyttämiskäytännöistä poiketen.

Digitaalisessa raportissa yhdistetään automaattinen tiedonkeruu ja käyttäjän kuvaus tapahtuneesta. Vikojen analysointia varten kehitettiin kaksiportainen luokittelu, jossa poikkeamalle valitaan kategoria ja sen mukaan määräytyvä alakategoria. Yläkategorioita voi

olla esimerkiksi materiaali- ja asennusvirheet, alakategorioita puolestaan väärin asennettu sekoittimen akseli tai vuotava letkuliitos. Luokittelun tarkoitus on auttaa tunnistamaan usein toistuvia laatupoikkeamia. Automaattisesti kerättävää informaatiota ovat

- kirjautumistiedot, eli käyttäjätunnus ja tietokoneen nimi
- testattavan laitteen tyyppi ja sarjanumero
- mittaustulokset.

Myös poikkeaman luokittelu ja vikakuvaus voidaan lisätä automaattisesti niissä tapauksissa, joissa ne kyetään tunnistamaan. Muussa tapauksessa käyttäjä syöttää tai täydentää puuttuvat tiedot.

Poikkeama- ja korjausraportit tehtiin samaan lomakepohjaan, jossa poikkeaman ilmetessä täytetään poikkeamalle osoitetut kentät ja toimenpiteiden jälkeen korjaukselle osoitetut kentät. Jotta poikkeama pystytään identifioimaan toimenpiteiden raportoimista varten, tulee aikaisemmin täytettyjen tietojen olla korjausraportissa näkyvillä, mutta niiden muokkaamisen tulee olla estetty.

First Pass Yield

Eräänlaisena keittolaitelinjan sisäisen laadun mittarina on jo aikaisemmin ollut FPY. FPY on suhteellinen osuus prosessin ensimmäisellä kerralla läpäisseistä tuotoksista kaikkiin prosessiin ensimmäistä kertaa syötettäviin tuotoksiin. FPY mitataan jokaiselle prosessille erikseen ja joskus myös osaprosesseille. Keittolaitelinjalla, jossa mittaustietoa kerätään vain koestuspisteellä, on saatu tulos koko linjan tulos. [17.]

Laatupoikkeamien analysointi

Laatupoikkeamien kertymä esitetään valitulla aikavälillä Pareto-analyysin tyyppisessä kaaviossa, johon ne on luokiteltu aikaisemmin mainitun luokitusjärjestelmän mukaan. Kaaviossa poikkeamat järjestetään esiintymismäärän mukaisessa laskevassa järjestyksessä ja siitä näkee heti mikä asia aiheuttaa eniten poikkeamia.

5 Koestusprosessin määrittely

Koestustapahtuma mallinnettiin prosessikaaviona käyttäjän näkökulmasta. Liitteessä 1 esitetyssä prosessimallissa määriteltiin mekanismit, joilla aikaisemmin suoritetun sekä meneillään olevan prosessin tulokset ja sen välitulokset vaikuttavat prosessin kulkuun. Se myös määrittää, mitkä toiminnot tapahtuvat manuaalisesti ja mitkä automaattisesti.

Käyttäjälle haluttiin antaa mahdollisuus hallita prosessin kulkua poikkeustilanteissa, joten malliin lisättiin mahdollisuus välittömään uudelleen yritykseen testin epäonnistuessa ja aikaisemmin suoritettujen testien ohittamiseen. Yritysten välillä tehtävät korjaustoimenpiteet voivat kuitenkin kohdistua laitteessa sellaisiin osiin, että aikaisempia testaustuloksia ei enää voi pitää voimassa olevina [13, s. 103]. Tästä syystä ylläpitäjälle annettiin mahdollisuus määritellä testien epäonnistuminen pakottamaan prosessi alkamaan alusta.

5.1 Alkuvalmistelu

Suoritettavien testien määrittely

Prosessi alkaa siitä, että käyttäjä syöttää järjestelmään testattavan laitteen tiedot. Tarvittavia tietoja ovat sarjanumero, tuoteperhe ja tuotteen konfiguroitavat ominaisuudet sekä siihen asennetut lisävarusteet. Syötettyjen tietojen perusteella järjestelmä generoi testaussekvenssin, joka sisältää kaikki laitteelle suoritettavat testit ja niissä sovellettavat parametrit ja raja-arvot.

Aikaisempien testausyritysten käsittely

Kaikki sekvenssiin kuuluvat testit käydään läpi ja niille etsitään edellisellä yrityksellä saatu tulos. Jos tällainen merkintä löytyy ja sen tulos on hylätty, esitetään käyttäjälle sama poikkeama raportti, joka esitettiin aikaisemmin vian ilmetessä, mutta tällä kertaa käyttäjän täytettävänä on enää kuvaus korjaustoimenpiteistä. On mahdollista, että täydennettäviä raportteja on edellisten yritysten jäljiltä useita. Testausta ei voi aloittaa, ennen kuin käyttäjä kuittaa kaikki raportit. Prosessi myöskään kulje eteenpäin sellaisesta testistä, jossa ilmennyttä vikaa ei ole korjattu.

Tapauksissa, joissa testi on aikaisemmin suoritettu hyväksytysti, eikä sen uudelleensuorittamista ole pakotettu, jätetään käyttäjälle mahdollisuus ohittaa testi. Ohitettaessa edellisellä kerralla saatu tulos jää voimaan. Näissä tilanteissa käyttäjän tehtäväksi jää arvioida, ovatko testausyritysten välillä suoritetut korjaustoimenpiteet voineet vaikuttaa laitteen sellaisiin osiin, joiden toimintaa tai turvallisuutta testissä mitataan.

5.2 Testaussekvenssin ajo

Testit ajetaan järjestelmän määräämässä järjestyksessä. Testin aikana käyttäjä suorittaa ohjeistuksen mukaiset toimenpiteet ja tarkastukset sekä odottaa automatisoitujen mittausten valmistumista.

Testistä saatu välitulos käsitellään välittömästi testin valmistuttua. Mittaustulokset, rajaarvot ja parametrit tallennetaan tietokantaan suodattamattomina. Hyväksytyn testin jälkeen siirrytään automaattisesti seuraavaan testiin ja hylkäyksestä haaraudutaan laatupoikkeamien käsittelyrutiiniin.

Keskeytys

Prosessin keskeyttäminen ajon aikana on aina mahdollista. Keskeytys pakottaa testin pysähtymään, mistä siirrytään testauslaitteiston alasajoon. Keskeyttäminen ei ole osa suunniteltua prosessin läpimenoa, joten se tulkitaan hylätyksi testaustulokseksi ja käynnistää laatupoikkeamien käsittelyrutiinin.

Poikkeamien käsittely

Epäonnistuneen testin jälkeen käyttäjälle annetaan täytettäväksi poikkeamaraportti. Raportin täyttämisen jälkeen vaihtoehdot ovat testausprosessin lopettaminen, seuraavaan testiin siirtyminen tai epäonnistuneen testin uudelleen yrittäminen.

Testausprosessin jatkaminen ja testin yrittäminen uudelleen voidaan testikohtaisesti estää ylläpitäjän toimesta, jos esimerkiksi katsotaan, ettei laitteen käyttäminen ole turvallista ennen korjausta tai että tulevat korjaustoimenpiteet joka tapauksessa mitätöisivät

sekvenssissä aikaisemmin suoritetut testit. Uudelleenyritys edellyttää myös, että poikkeamaraportin kenttä korjaustoimenpiteille on täytetty. Korjauksesta on täytettävä kuvaus, vaikka mitään toimenpiteitä ei olisi tehty, sillä eri tuloksen saaminen ilman toimenpiteitä voi kertoa mittalaitteiston epäluotettavuudesta.

5.3 Jälkiprosessointi

Kun testausosuus päättyy, eli joko viimeisen testin valmistuessa tai muusta syystä lopetettaessa, siirrytään jälkiprosessointiin. Jälkiprosessoinnissa testauslaitteisto ajetaan alas ja todetaan laitteen kelpoisuus. Kaikki testit läpäisseelle laitteelle luodaan koestusraportti, jolla osoitetaan hyväksyttämiskriteerien läpäisy.

6 Järjestelmän suunnittelu ja toteutus

Järjestelmän ohjelmistoratkaisut päätettiin toteuttaa alusta alkaen itse. Valmiit ratkaisut, kuten National Instrumentsin Test Stand, hylättiin kustannusten, rajoitusten, sekä työlään käyttöönoton takia. Lähdekoodin jääminen yrityksen omaan käyttöön helpottaa muutosten tekemistä, mikä pienentää riippuvuutta sidosryhmistä, kuten mittalaite- tai järjestelmätoimittajista ja mahdollistaa järjestelmän jatkuvan parantamisen. Kehittämällä oma järjestelmä voitiin käyttöliittymä ja ohjelmistotekniset ratkaisut myös räätälöidä prosessille ja testattaville laitteille soveltuviksi. Erityisesti tehtaan laaja tuotevalikoima ajoi tähän ratkaisuun.

Itse kehitetyn ja ylläpidetyn järjestelmän riskinä on kuitenkin osaamisen katoaminen yrityksestä henkilöstön vaihtumisen myötä. Vaihtoehtoa pidettiin silti parempana, kuin ulkopuolisen yrityksen toimittamaa järjestelmää, sillä järjestelmätoimittajan ajautuessa ongelmiin voi tuen saaminen olla vielä vaikeampaa. Riskejä pienennettiin hyvillä työmenetelmillä, kuten kommentoinnilla ja dokumentoinnilla. Tarvittavan ohjelmiston arvioitiin olevan lähdekoodin määrässä mitattuna niin pieni, että ylläpidon voisi tarvittaessa siirtää ulkopuoliselle taholle.

Ohjelmiston kehityksessä noudatettiin prototyyppimallia. Mallille tunnusomaista on, että kehittäminen aloitetaan käyttöliittymäsuunnittelusta ja toteutukseen lisätään vaiheittain

uusia ominaisuuksia, joita asiakas pääsee arvioimaan. Samalla asiakkaalta otetaan vastaan muutospyyntöjä ja ideoita uusista ominaisuuksista. Prototyyppimalli valittiin, koska sillä saatiin nopeasti aikaan demo, ja visiota lopputuotteen käyttökokemuksesta ja ominaisuuksista päästiin esittelemään tilaajalle, jolla projektin alussa ei vielä ollut näkemystä lopputuotteesta. [18, s.303–311.]

6.1 Ohjelmistoarkkitehtuuri

Laitteistoriippumattomuutta tavoiteltiin jakamalla ohjelmisto osiin. Osat olivat koestusprosessia hallinnoiva pääohjelma, ja määrittelemätön määrä aliohjelmia, jotka suorittavat tarkoin määrättyjä tarkastuksia tai mittauksia tietylle tuoteperheelle tai sen osalle. Pääohjelmaan oli konfiguroitavissa testit parametreineen ja raja-arvoineen, niiden suoritusjärjestys ja aliohjelmat, jotka testit testattavalle laitteelle toteuttavat.

Mittausten hajauttaminen erillisiin aliohjelmiin irrottaa kaiken laitteistoriippuvaisen koodin pois pääohjelmasta. Näin prosessista ja tiedonkeruusta huolehtivaa koodia ei tarvitse muokata, vaikka kaikki testit, mittalaitteet tai testattavat laitteet muuttuisivat. Mittalaitteiden ohjaukselle tehtiin vielä oma kirjasto, joka sisältää rajapinnan kaikille määrityksille, joita kappaletestaukseen kuuluvat testit tyypillisesti tarvitsevat. Mittalaitteiden vaihtuessa tarvitsee siis ainoastaan muokata tämä kirjasto kommunikoimaan uuden laitteiston kanssa.

Hajauttaminen mahdollistaa yksittäisen testin suunnittelun tarkoin määritetyille laitteille. Näin esimerkiksi kahdessa tuotteessa eri tekniikalla toteutettu sama ominaisuus voidaan testata eri tavalla. Pääohjelma huolehtii, että testi suoritetaan vain, kun testi on soveltuva testattavalle laitteelle. Samalla hajauttaminen mahdollistaa myös vaiheistetun käyttöönoton, jossa järjestelmää laajennetaan asteittain kattamaan eri tuoteperheitä ja niihin kuuluvien laitteiden eri versioita.

6.2 Ohjelmointikielet

Vaatimuksina ohjelmointikielen valintaan olivat laaja käyttäjäkunta, kattava komponenttikirjasto Windows-ympäristöä varten sekä helppokäyttöinen kehitysympäristö. Komponenttikirjastoon tuli sisältyä ainakin käyttöliittymäkomponentit, verkko- ja tietokantayhteydet, tiedostonhallinta, ulkoisten prosessien hallinta, sekä tuki monisäikeisyydelle. Käytännössä valinta tehtiin C++-kielen ja sille kehitetyn QT-kehitysympäristön sekä Microsoftin kehittämien C#-kielen ja Visual Studio -kehitysympäristön väliltä. Valinnassa päädyttiin C#-kieleen sen helposti omaksuttavan syntaksin, automaattisen roskienkeruun ja Visual Studion helpon käyttöönoton takia.

C# on Microsoftin kehittämä ohjelmointikieli, joka on suunniteltu C++:n pohjalta tavoitteena yhdistää C++:n tehokkuus yksinkertaisempaan syntaksiin ja korjata sen pahimpia ongelmia. Merkittävimpiä eroja ovat, että C# on vahvasti tyypitetty ja osoittimien käyttöä on rajoitettu, mikä estää kokonaan tietyntyyppisten ohjelmointivirheiden mahdollisuuden. Rajoitusten takia saman toiminnallisuuden saavuttaminen voi vaatia enemmän koodia, mutta tuotettu koodi on helpommin luettavaa ja yksinkertaisempaa, mikä pienentää virheiden mahdollisuutta. [18, s. 191.]

Visual Studio on Microsoftin tarjoama integroitu kehitysympäristö, joka tukee C#-kieltä sekä muutamia muita ohjelmointikieliä. Integroidun kehitysympäristön tehtävänä on koota yhteen kaikki ohjelmistokehitykseen tarvittavat työkalut aina tekstieditorista kääntäjään ja versionhallintaan. Visual Studio sisältää useita ohjelmointia helpottavia ominaisuuksia, tärkeimpänä Intellisense, joka varoittaa ohjelmointivirheistä reaaliaikaisesti lähdekoodia kirjoitettaessa. [19. s. 10.]

Testejä harkittiin aluksi toteutettavaksi graafisella ohjelmointikielellä LabVIEW-kehitysympäristössä. LabVIEW on yleisesti testaussovelluksissa käytetty alusta, joten sen käytöllä ajateltiin tarjota uusien testien kehittäjille helppo rajapinta järjestelmään. Lisäksi useat mittalaitevalmistajat tarjoavat laitteiden etäkäyttöä varten LabVIEW-ajurit, joista löytyi päivitetty versio myös olemassa olevalle laitteistolle.

LabVIEW kuitenkin hylättiin lyhyehkön perehtymisen jälkeen, koska kehitystyön todettiin olevan projektin aikatauluun liian hidasta ja tiedon virtaukseen perustuvan syntaksin

omaksuminen tuotti hankaluuksia perinteiseen ohjelmointiin tottuneelle. Mittalaitteiden hyvin dokumentoitu sarjaporttirajapinta ja .NET-viitekehyksen sarjaportin käsittelyyn tarkoitetut ominaisuudet mahdollistivat testien toteuttamisen C#:lla.

6.3 Työssä hyödynnetyt ohjelmointiparadigmat

Ohjelmistokehityksen tärkeimpänä työkaluna käytettiin olio-ohjelmointia. Siinä keskeistä on ohjelman jakaminen moduuleihin ja asioiden kuvaaminen joukolla muuttujia ja funktioita. Näitä muuttujien ja funktioiden kokoelmia kutsutaan olioiksi, ja ne usein mallintavat reaalimaailmassa olemassa olevia asioita, mutta joskus myös abstrakteja käsitteitä [19, s. 186–187]. Tässä alaluvussa selitetään ohjelmoinnin käsitteitä ja toimintamalleja siitä näkökulmasta, mitä niillä on työssä tavoiteltu ja siinä laajuudessa mikä ohjelmistokuvauksen ymmärtämisen kannalta on keskeisiä.

Olio-ohjelmointi

Olio-ohjelmoinnin pääkomponentit, *luokat* ja *instanssit* ovat vahvasti yhteydessä toisiinsa. Luokka on tietotyyppi, jossa määritellään, miten siihen kuuluvan olion tiedot esitetään ja miten niitä käsitellään. Se voi sisältää muuttujia ja metodeja. Muuttuja voi olla tyypiltään muun muassa luku, merkkijono tai toinen luokka. Metodi taas on funktio, jota voi tiettyjä poikkeuksia lukuun ottamatta kutsua vain oman luokkansa instanssin kautta. Luokkaan kuuluvilla muuttujilla ei yleensä voi olla arvoja, eikä luokalla yksinään tee mitään. [19, s. 187.]

Luokkaan kirjoitettujen määrittelyjen perusteella voidaan luoda instanssi. Instanssi on jonkin luokan olio, jolla on luokassa määritellyt muuttujat ja instanssille kuuluvana ne voivat nyt saada arvoja. Instanssi on siis aina jossakin tilassa. Saman luokan pohjalta voidaan luoda useita instansseja, jotka toiminnaltaan ovat identtisiä, mutta tilaltaan eli muuttujien arvoiltaan uniikkeja. [19, s.187.]

Perintä tarkoittaa olemassa olevan luokan, niin kutsutun yläluokan ominaisuuksien kopioimista uuteen luokkaan, jota kutsutaan alaluokaksi. Alaluokkaa voi olla eräänlainen jatke yläluokalle, jossa on uusia ominaisuuksia, se voi olla yläluokan erikoistapaus, jossa joitakin ominaisuuksia on rajoitettu tai muutettu, tai niiden yhdistelmä. [19, s. 194–195].

Vahvasti tyypitetyissä kielissä muuttujaan voi asettaa vain sellaisen arvon, joka on samaa tyyppiä, kuin muuttuja itse [18, s. 191]. Yläluokasta periytyvän alaluokan olio kuitenkin kuuluu sekä ala- että yläluokkiin, eli sen voi sijoittaa kumpaa tahansa tietotyyppiä olevaan muuttujaan. Näin tehtäessä puhutaan *monimuotoisuudesta*. [19, s. 196–197.]

Kun alaluokan olio on sijoitettuna yläluokan tietotyypin muuttujaan, on se edelleen tietotyypiltään alaluokkaa, mutta siitä saa käyttöönsä vain yläluokassa määritetyn toiminnallisuuden. Olion sisällä tapahtuvat toiminnot pysyvät siis sellaisina, kuin ne alaluokassa on määritelty. Näin monimuotoisuuden avulla kahta tai useampaa samankaltaisen luokan instanssia voi kohdella kuin ne olisivat samaa luokkaa. [19, s. 197.]

Tilanteessa, jossa yläluokasta johdetaan useita alaluokkia, mutta itse yläluokkaa sellaisenaan ei tarvita mihinkään, voidaan yläluokka määritellä *abstraktiksi*. Abstraktista luokasta ei voi luoda instanssia, mutta se voi sisältää mitä tahansa alaluokkiin periytyviä ominaisuuksia. Koska abstraktista luokasta ei voi luoda instanssia, ei sen ominaisuuksille ole pakko määritellä toteutusta. Yleensä abstraktissa luokassa määritellään toteutus sellaisille ominaisuuksille, jotka periytyvät alaluokkiin muuttumattomina. [19, s. 232–233.]

Ohjelmointirajapinta on erikoistapaus abstraktista luokasta. Siinä määritellään luokan ulospäin näkyvät ominaisuudet, mutta ei niiden toteutusta. Koska toteutusta ei määritellä, ei ohjelmointirajapinnasta myöskään voi luoda instanssia. Muuttujan tietotyyppi voi kuitenkin olla ohjelmointirajapinta, jolloin muuttujaan sijoitetusta oliosta saadaan käyttöön vain ohjelmointirajapinnassa määritelty toiminnallisuus. [19, s. 232–233.]

Monimuotoisuuden lisäksi ohjelmointirajapinnan käytöllä tavoitellaan myös *informaation piilottamista*. Informaatiota halutaan piilottaa esimerkiksi kun olio annetaan sellaisen funktion käsiteltäväksi, jonka ei haluta tekevän siihen muutoksia. Tarvittavaa funktiota varten voidaan kirjoittaa erityinen ohjelmontirajapinta, joka paljastaa oliosta vain funktion toiminnan kannalta välttämättömät ominaisuudet. Tällä tavalla kukaan ei vahingossakaan voi lisätä funktioon toiminnallisuutta, joka rikkoisi siihen syötetyn olion. Informaatiota voi piilottaa myös ylä- ja alaluokkien välillä määrittelemällä yläluokan ominaisuuksia sellaisiksi, että edes alaluokka ei näe niitä. [18, s. 74–75.]

Tapahtumat

Tapahtumat ovat ohjelman ajon aikana tapahtuvia ohjelmoijan määrittämiä asioita, joiden lauetessa ohjelman suoritus voi hypätä toiseen paikkaan käsittelemään tapahtumaa ilman, että tapahtuman laukaiseva funktio tietää mikä osa koodissa tapahtumaa käsittelee. Tapahtumaa kuuntelemassa voi olla useita käsittelijöitä, jotka kaikki suoritetaan sen lauetessa. Käsittelijät asetetaan ohjelmallisesti kuuntelemaan tapahtumaa ja kuuntelemisen voi myös lopettaa. [19, s. 377–378.]

Luokkaan kuuluvat tapahtumat ja niiden käsittelijät ovat aina instanssikohtaisia. Kun tapahtuma laukeaa instanssissa, tiedetään tarkalleen missä oliossa se tapahtui ja ketkä vastaavat sen käsittelystä. Tapahtumia käytetään usein käyttöliittymäkoodissa, jossa voi esimerkiksi olla useita painonappeja. Painonapit ovat saman luokan olioita, mutta jokaisella niistä on eri tehtävä. Jotakin nappia painettaessa suoritetaan juuri sen painamistapahtumaan liitetty koodi, vaikka nappi itse ei ole erityisasemastaan millään tavalla tietoinen.

Monisäikeisyys

Imperatiivisessa ohjelmoinnissa on yksi reitti, *säie*, jota pitkin ohjelma etenee suorittaen komentoja järjestyksessä yksi kerrallaan. Ajon aikana ohjelma voi kuitenkin käynnistää uusia säikeitä eli edetä yhtä aikaa kahta toisistaan riippumatonta reittiä pitkin, jolloin puhutaan monisäikeisyydestä. [20, s.4.]

Tietokoneessa voi olla useita ohjelmia samanaikaisesti käynnissä ja samalla sen on suoritettava käyttöjärjestelmän toimintoja. Kaikki ohjelmat ja toiminnot pyörivät omissa säikeissään. Käyttöjärjestelmä antaa suorittimen resurssit aina vuorotellen lyhyeksi ajaksi yhden säikeen käytettäväksi, jolloin vaikuttaa siltä kuin säikeitä suoritettaisiin yhtäaikaisesti. Monta suoritinta tai suoritinydintä sisältävässä tietokoneessa säikeitä on mahdollista suorittaa aidosti yhtä aikaa, mutta kaikki sovellukset eivät hyödy aidosta rinnakkaisuudesta ja aito rinnakkaisuus on vain yksi monisäikeisyydellä saavutettavista hyödyistä. [20, s. 7.]

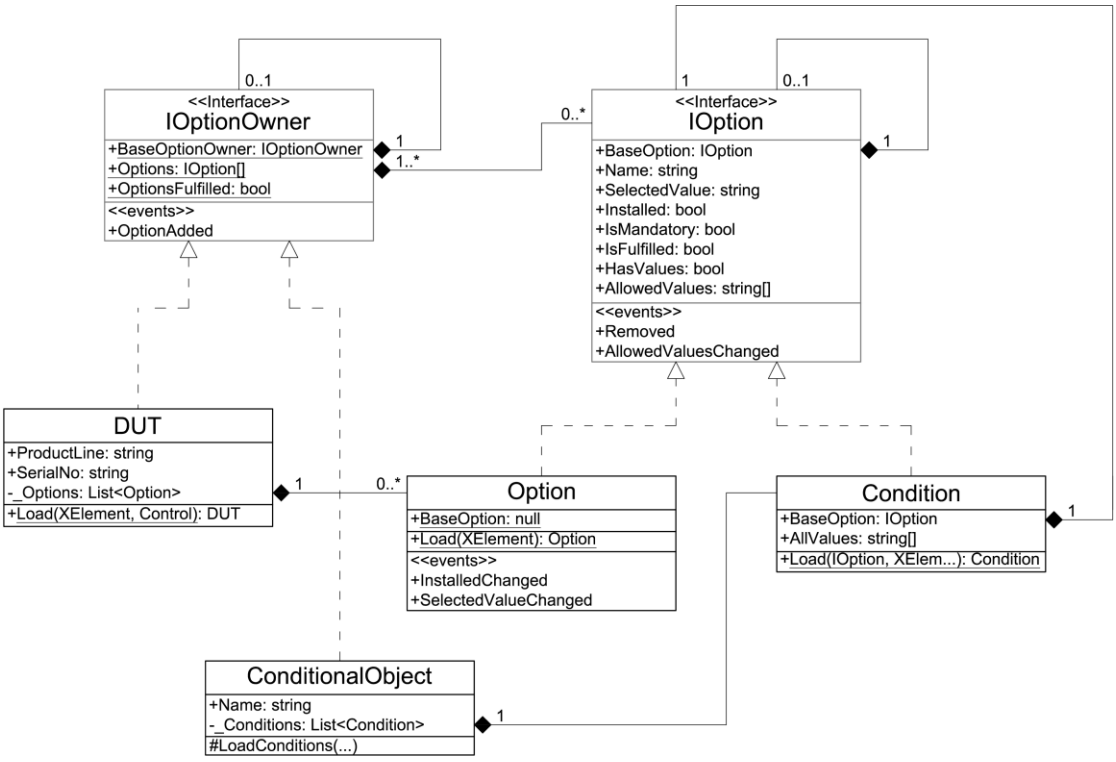
Ohjelmoinnissa voi tulla vastaan tilanteita, joissa suoritettavan ohjelman täytyy odottaa jonkin siitä itsestään riippumattoman toiminnon suorittamista loppuun. Odotuksen aikana ohjelma ei pääse eteenpäin ja tilanteen kestäessä pitkään se vaikuttaa niin sanotusti jäätyneen. Suorittamalla pitkään kestävä toiminto eri säikeessä kuin käyttöliittymä voidaan myös pitkään kestävän toiminnon aikana ottaa uusia komentoja vastaan. [21, s.248.]

6.4 Pääohjelman luokkasuunnittelu

Luokat pyrittiin suunnittelemaan hyvien ohjelmointikäytäntöjen mukaan siten, että jokainen niistä toteuttaisi vain yhden asian. Kehitystyössä noudatettu projektimalli ja vähäinen kokemus olio-ohjelmoinnista kuitenkin haittasivat periaatteiden noudattamista ja uusien paradigmojen omaksuminen on havaittavissa projektin myöhäisemmässä vaiheessa kirjoitetuista luokista. Tästä syystä muun muassa testattavaa laitetta kuvaava luokka oli lopulta vastuussa aivan liian suuresta määrästä toimintoja ja uudelleensuunnittelun tarpeessa.

6.4.1 Testattava laite, konfiguroitavat ominaisuudet ja lisävarusteet

Tuoteperheitä ja testattavaa laitetta kuvaamaan kehitettiin DUT-luokka. Kuten kuvioissa 1, 2 ja 3 esitetyistä riippuvuuksista voi havaita, on DUT-olion tehtävä hallita tuoteperheen konfiguroitavia ominaisuuksia, siihen saatavilla olevia lisävarusteita, tuotteille tehtäviä testejä, testaustuloksia ja testausprosessin tilaa.



Kuvio 1. Testattava laite ja sen konfiguroitavat ominaisuudet ja asennetut lisävarusteet esitetään DUT- ja Option–luokkien avulla. ConditionalObject- ja Condition-luokat mahdollistavat haarautumisen ja ehdollisuuden rajoittamalla ConditionalObject-luokan ominaisuudet perivän olion käyttöä erikseen määritettävillä konfiguraatiolla.

Konfiguroitavia ominaisuuksia ja lisävarusteita kuvataan Option-luokalla. Ominaisuus ja lisävaruste eroavat toisistaan siten, että ominaisuus on aina olemassa, mutta lisävarustetta ei ole pakko valita. Ominaisuudelle on myös aina oltava listattuna vähintään kaksi toisistaan poikkeavaa arvoa, mutta lisävarusteelle voi riittää vain kyllä ja ei. On kuitenkin olemassa myös sellaisia lisävarusteita, joista on valittavana erilaisia vaihtoehtoja. Esimerkki ominaisuudesta on keittopadan nettotilavuus, joka voi saada kahdeksan arvoa väliltä 40–400 l, monivalintaisesta lisävarusteesta puolestaan käsisuihku, joka voi olla tavallinen, kelautuva, ”heavy duty” -malli, tai puuttua kokonaan.

6.4.2 Varustelusta riippuva haarautuminen ja ehdollisuus

Jotta testauksessa voisi määritellä haarautumista laitteiden erilaisten variaatioiden perusteella, kehitettiin Condition- ja ConditionalObject-luokat. Condition-luokan olio on yksittäistä ominaisuutta tai lisävarustetta koskeva ehto, joka toteutuu, kun ominaisuus saa

ennalta määritettyjä arvoja. Se sisältää viittauksen siihen Option-olioon, jonka sallitut arvot se määrittää. Sallituiksi arvoiksi voidaan määritellä vain sellaisia arvoja, jotka rajoituksia koskeva ominaisuuskin voi saada. Lisävarusteen tapauksessa ehto voidaan määritellä toteutuvaksi myös siinä tapauksessa, että lisävarustetta ei ole asennettu.

ConditionalObject-luokka on abstrakti luokka, jonka olioihin voi ladata yksittäisiä ominaisuutta rajoittavia Condition-luokan ehtoja. ConditionalObject-luokan oliolla siis määritellään kaikki tekijät, jotka vaikuttavat varustelusta riippuvaan haarautumiseen. Se sisältää myös metodin, jolla kokeillaan, onko testattavan laitteen varustelu sellainen, että kaikki ehdot toteutuvat. Tätä tietoa käytetään, kun päätellään mitä testejä, raja-arvoja ja parametreja testattavalle laitteelle sovelletaan.

Sekä Condition- että Option-luokat toteuttavat IOption-ohjelmointirajapinnan. Kuviosta 1 on havaittavissa, miten tämän ohjelmointirajapinnan kautta sekä Option- että Condition-olio voi esiintyä toisen Condition-luokkaan kuuluvan olion viittauksena siihen ominaisuuteen, jonka sallittuja arvoja rajoitetaan. Näin muodostuu ketju, jonka lopusta löytyy varsinaista ominaisuutta kuvaava olio. Ketjun avulla estetään sellaisten määritysten tekeminen ketjun lopussa, jotka eivät välissä olevan rajoituksen takia missään tilanteessa voisi toteutua.

Jokainen ConditionalObject-olio viittaa siihen olioon, joka omistaa ne ominaisuudet ja lisävarusteet, joiden mukaan se voi luoda rajoituksia. Tätä tarkoitusta varten luotiin IOptionOwner-ohjelmointirajapinta, jonka toteuttavat sekä DUT- että ConditionalObject-luokat. Ohjelmointirajapinnan tehtävänä on estää sen kautta DUT-oliota käyttävältä ConditionalObject-olioilta pääsy DUT-olion muihin toimintoihin. Samalla se mahdollistaa ConditionalObject-luokan olioiden ketjuttamisen samaan tapaan kuin edellisessä kappaleessa Option- ja Condition-oliot IOption-ohjelmointirajapinnan kautta.

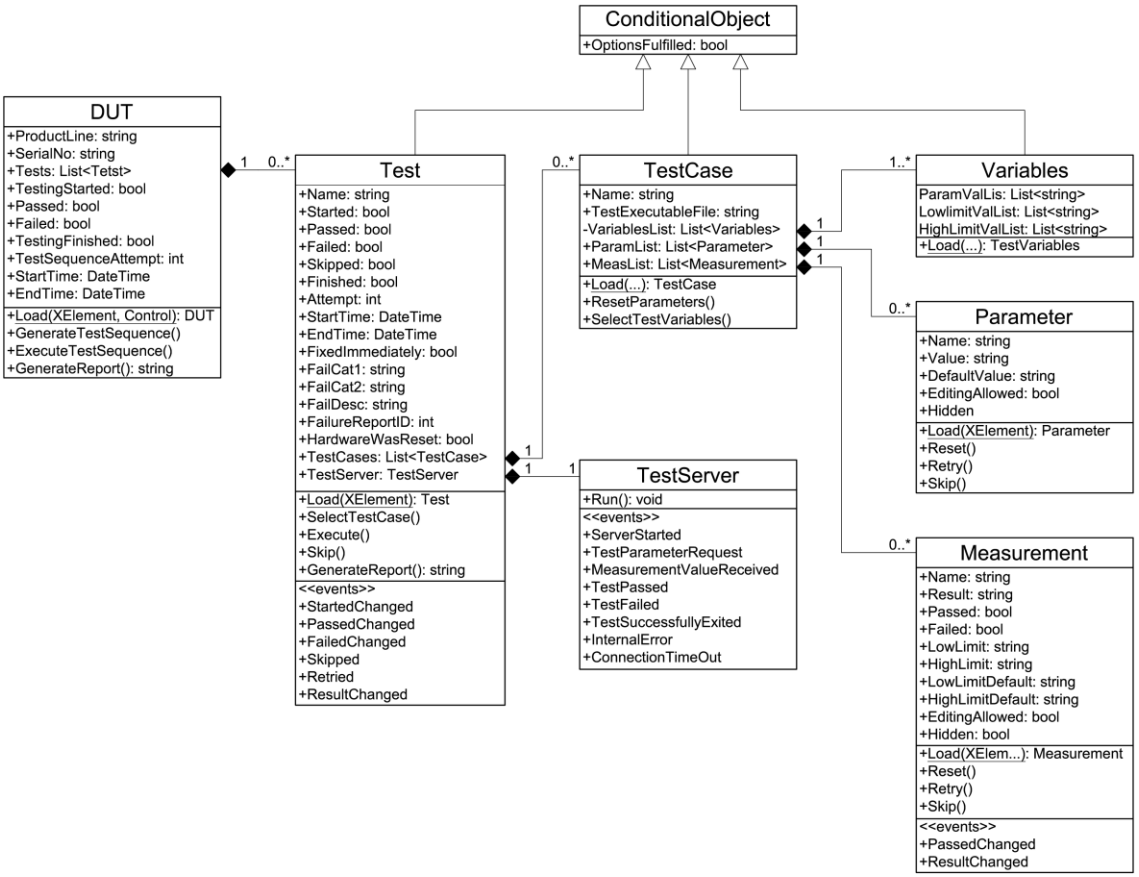
6.4.3 Testit, parametrit ja raja-arvot

Testit, parametrit ja raja-arvot vaihtelevat testattavan laitteen ominaisuuksien ja siihen asennettujen lisävarusteiden mukaan. Laitteelle suoritettavia testejä kuvataan Test-luokalla, erilaisia tapoja suorittaa tarvittava testi TestCase-luokalla ja eri tavalla konfigu-

roiduille laitteille sovellettavat parametrit ja raja-arvot tallennetaan Variables-luokan olioihin. Kuviossa 2 esitetään, miten luokkien Test-, TestCase- ja Variables-oliot voivat pitää sisällään määrittelyjä, jotka rajaavat ne koskemaan vain tietyllä tavalla konfiguroituja laitteita perimällä ConditionalObject-luokan ominaisuudet.

Laitteelle suoritettavia testejä kuvaava Test-luokka on väliporras, jonka tehtävänä on määritellä ne tilanteet, joissa kyseinen testi tulee suorittaa laitteelle. Esimerkiksi jäähdytystoiminto testataan vain niissä laitteissa, joihin lisävaruste on asennettu. Tämän lisäksi Test-luokka sisältää toimintoja testin ajamiseen, tuloksen määrittämiseen sekä raportointiin.

Testin varsinaiset ominaisuudet, kuten mitä parametreja ja mittauksia siihen kuuluu, ja polun, missä testin toteuttava ohjelma sijaitsee, määritellään TestCase-luokassa. Haarautumisen olisi voinut toteuttaa myös siten, että Test-luokka sisältäisi kaikki TestCase-luokan ominaisuudet, mutta toteutettu ratkaisu auttaa erottamaan erilaisten testaustapojen ja prosessiin kuuluvien testien välistä eroa. Sillä saadaan aikaiseksi myös virheilmoitus, jos jonkin laitteen vaatimaa testaustapaa ei vielä ole määritelty. Esimerkiksi jäähdytystestiä kuvaavalle Test-luokan oliolle voisi kuulua omat testaustavat eri tavalla toimiville jäähdytysjärjestelmille, joita ovat verkostovedellä, suljetun kierron jäävedellä sekä niiden yhdistelmällä tapahtuva jäähdytys.



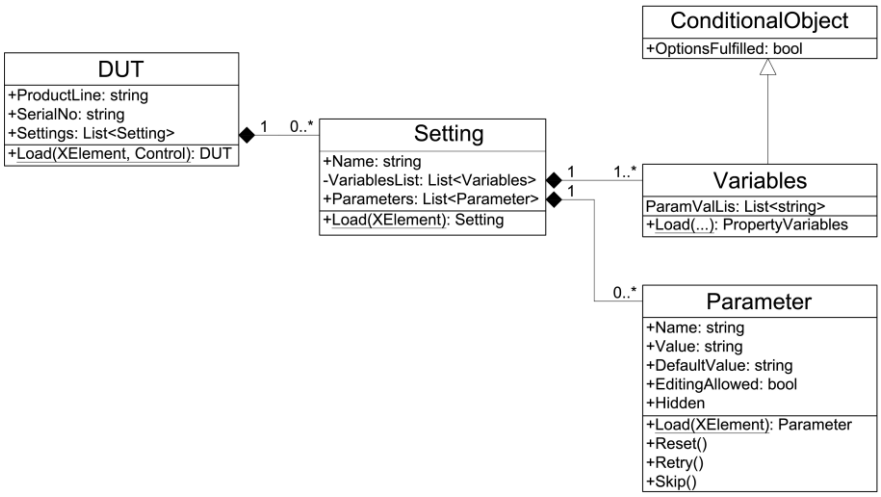
Kuvio 2. Testaussekvenssi koostuu ehdollisista testeistä (Test), jotka voivat haarautua yhteen tai useampaan erilaiseen testaustapaan (TestCase). Jokaisella testaustavalla on omat parametrinsa, sekä mittauksena. Parametrit ja mittausten raja-arvot voivat olla ominaisuuksista ja lisävarusteista riippuvaisia ja eri konfiguraatioille soveltuvia arvoja säilytetään Variables-olioissa.

Parametreja ja mittaustuloksia käsitellään Parameter- ja Measurement-luokkien avulla. Measurement-luokassa on sisäänrakennettu tarkastelu raja-arvojen sisällä pysymiseen mittaustuloksen hyväksyttävyyden varmistamiseksi. Numeeriset arvot, jotka parametreille ja mittaustulosten raja-arvoille annetaan, haetaan siitä Variables-luokan oliosta, jolle asetetut määritykset vastaavat testattavaa laitetta.

6.4.4 Tuotekohtaiset parametrit

Tuotekohtaisia parametreja käytetään välittämään sellaista informaatiota, jota useammat testit voivat tarvita. Määrittelemällä nämä parametrit erillään testikohtaisista parametreista varmistetaan, että ne eivät muutu testin vaihtuessa, vähennetään konfiguroitavien asetusten määrää ja varmistetaan että ne ovat kaikkien testien saatavilla.

Tuotekohtaisia parametreja kuvaava Parameter-luokka ja siihen asetettavia arvoja kuvaava Variables-luokka ovat toiminnaltaan hyvin lähellä testausmenetelmille määriteltyjä parametreja ja niille asetettavia arvoja, kuten kuviosta 3 on havaittavissa. Luokkahierarkiassa DUT- ja Parameter-luokkien välissä olevaa Setting-luokkaa käytetään vain kokoamaan yhteen tuotekohtaisia parametreja, joiden arvoja koskevat samat valintasäännöt.



Kuvio 3. Järjestelmä sisältää myös tuotekohtaisia parametreja, jotka ovat yhteisiä jokaiselle testille. Niiden toteutus on saman kaltainen, kuin testitapaukselle kuuluvien parametrien.

6.5 Pääohjelman käyttöliittymäsuunnittelu

Käyttöliittymäsuunnittelussa sovellettiin lean-ajattelutapaa ja hyvän käyttökokemuksen periaatteita. Tehokkuus ja helppokäyttöisyys ovat saavutettavissa tarkoin harkitulla ominaisuuksien määrittelyllä ja esillepanolla. Helppokäyttöisen ohjelman suunnitteluperiaate on, että todennäköisesti tapahtuvaan toimintoon liittyvät komponentit pidetään näkyvillä,

harvoin tapahtuvaan liittyvät piilotetaan valikon taakse ja toiminnon ollessa mahdoton, siihen liittyvät komponentit deaktivoidaan tai poistetaan kokonaan. [22.]

Tyypillisen käyttäjän kannalta pääohjelmalla on neljä keskeistä toimintoa, joita tarvitaan joka kerta kun prosessi toistuu. Niiden avulla käyttäjä

- määrittelee mitä laitetta testataan
- hallitsee testausprosessin kulkua
- näkee kokonaiskuvan prosessista, eli mitä on tehty, mitä ollaan tekemässä ja mitä on vielä tekemättä
- näkee osaprosessin tarkat tiedot, eli raja-arvot, mittaustulokset, tilan, läpäisyn ja mahdolliset virheilmoitukset.

Prosessi etenee suoraviivaisesti, ja seuraavassa vaiheessa tarvittavien käyttöliittymäkomponenttien sisältö määräytyy käyttäjän edellisessä vaiheessa tekemien valintojen mukaan. Tästä syystä seuraavan vaiheen komponentit luodaan vasta kun niiden ominaisuudet ovat tiedossa. Vastaavasti edelliseen vaiheeseen palaaminen ei enää ole mahdollista. Edelliseen vaiheeseen liittyvät komponentit päätettiin jättää näkyville, mutta deaktivoida, jotta käyttöliittymästä selviää, miten sen hetkiseen tilaan on päädytty.

Dynaamisesti muuttuva käyttöliittymä, jossa kaikki prosessiin sisältyvät toiminnot ovat näkyvillä, mutta vain muutama toiminto käytettävissä, antaa käyttäjälle selkeän kuvan siitä, miten prosessissa tulee edetä ja mitä se sisältää. Samalla estetään toimintojen aktivoituminen väärällä hetkellä.

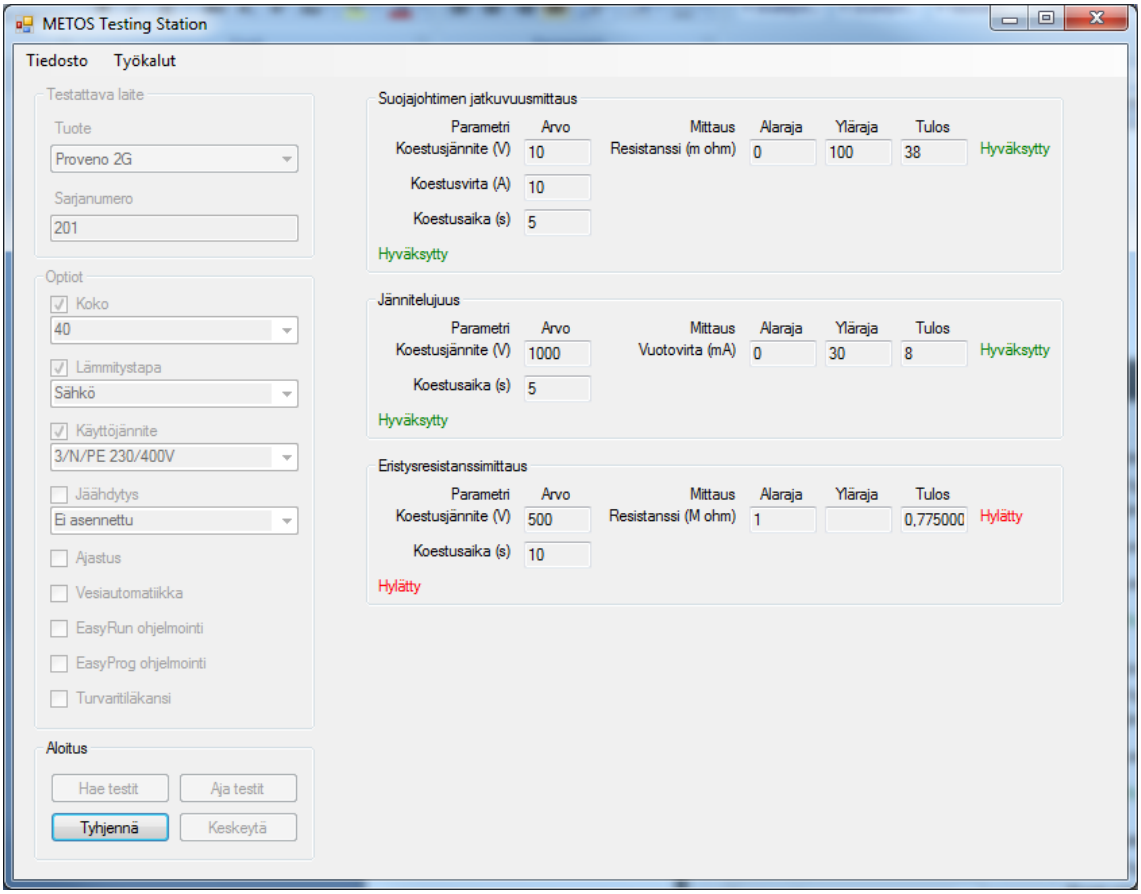
Ulkoasu

Käyttöliittymän toiminnalliset komponentit ryhmiteltiin siten, että prosessimallin mukaisessa käytössä niiden välillä siirrytään ylhäältä alas ja vasemmalta oikealle. Prosessin hallintaan liittyvät komponentit sijaitsevat vasemmassa laidassa ja loput tilasta on jätetty monitorointiin.

Kuvassa 1 näkyvän pääohjelman ikkunan vasemmassa ylänurkassa sijaitsee testattavan laitteen määrittely. Tuoteryhmä ja sarjanumero ovat ominaisuuksia, jotka kuuluvat

jokaiselle tuotteelle. Heti tuoteryhmän valitsemisen jälkeen generoidaan laitteen konfiguroitavien ominaisuuksien ja siihen asennettujen lisävarusteiden määrittelyyn tarvittavat komponentit. Vasemmassa alakulmassa sijaitsevat prosessin hallintaan liittyvät toiminnot ovat näkyvissä koko ajan.

Testattavalle laitteelle tehtävät testit on listattu suoritusjärjestyksessä. Testien yksityiskohtaiset tiedot on ryhmitelty kehyksiin. Tärkeimmät tiedot, kuten testin tila ja läpäisy, ovat korostettuna huomioväreillä, jotta järjestelmän tilan näkee nopeasti.



Kuva 1. Kuvakaappaus pääohjelman käyttöliittymästä suoritetun testausprosessin jälkeen.

Optimointi

Koska prosessissa on tarkoin määritelty käyttäjän järjestelmään syöttämä informaatio ja järjestys, jossa se syötetään, oli käyttöliittymä mahdollista optimoida vastaanottamaan se tehokkaasti. Liikkumista on helpotettu ja havainnollistettu aktivoimalla valmiiksi se

komponentti, jota käyttäjän oletetaan tarvitsevan seuraavaksi. Esimerkiksi tuotteen valitsemisen jälkeen sarjanumeron tekstikenttä aktivoituu ottamaan tekstiä vastaan ilman, että käyttäjä tarvitsee valita sitä.

Prosessissa eteenpäin vievät painonapit on määritetty reagoimaan enter-näppäimen painallukseen ja prosessin keskeyttävät vastaavasti esc-näppäimeen. Pikanäppäimiä sovelletaan testaussekvenssin generoimiseen ja asetusten tyhjentämiseen. Turvallisuussyistä ominaisuutta ei kuitenkaan käytetä prosessin käynnistämiseen, eikä vahinkopainallusten aiheuttamien tahattomien keskeytysten takia käynnissä olevan testauksen pysäyttämiseen.

Käyttöliittymän komponenttien välillä voi liikkua myös sarkainta painamalla. Komponenttien aktivoitumisjärjestys määriteltiin sen mukaan mitä käyttöliittymäkomponenttia oletettiin seuraavaksi tarvittavan.

6.6 Käyttäjien hallinta

Käyttäjien hallinta toteutettiin Windowsin AD (Active Direcory) -ominaisuudella. AD on DNS-järjestelmää hyödyntävä, verkon resurssien keskitettyyn hallintaan tarkoitettu työkalu. Se tarjoaa palveluita käyttäjien ja laitteiden identifiointiin ja käyttöoikeuksien hallintaan. [23, s. 1 - 2].

Vaihtoehtona oli kehittää oma kirjautumisjärjestelmä, jonka käyttäjätunnukset olisivat olleet tallennettuna verkkohakemistoon. AD valittiin toimeksiantajan pyynnöstä, koska yrityksen muutkin järjestelmät hyödyntävät sitä. AD tarjoaa myös paremman suojaustason ja siihen on olemassa valmiit kirjastot, joiden ansiosta se saatiin nopeasti implementoitua.

Järjestelmän asetuksiin lisättiin muuttujat eri käyttäjätasoja vastaavien AD-ryhmien nimille. Näin ollen yrityksen IT-osasto pystyy muuttamaan työntekijöiden käyttöoikeuksia lisäämällä heidät tiettyä käyttäjätasoa vastaavaan ryhmään. Suoritettaessa käyttöoikeuksia vaativia toimintoja järjestelmä kutsuu rutiinia, joka tarkistaa käyttäjän kuulumisen määriteltyihin ryhmiin ja palauttaa lueteltuna tyyppinä korkeimman tason, johon sillä hetkellä sisään kirjautunut käyttäjä kuuluu.

6.7 Rajapinnat

Järjestelmään tarvittiin rajapinta pääohjelman ja testien väliseen kommunikaatioon. Konfiguraatiota koskeva data puolestaan päätettiin säilyttää tiedostoihin tallennettuna ja tähän tarkoitukseen kehitettiin rajapinta olioiden tallentamiseksi tiedostoihin ja rekonstruktioimiseksi takaisin.

Molempiin tarkoituksiin kehitettiin XML-merkintäkieleen perustuva esitysmuoto. XML on W3C:n suosituksessa määritelty tekstimuotoinen formaatti rakenteellisen tiedon jäsentelyyn [24]. Se tarkoin määritelty, jotta sitä voi muodostaa ja lukea tietokoneella, mutta samalla riittävän selkolukuista ihmisen ymmärrettäväksi ja käsin tuotettavaksi. XML-muotoisen tiedon käsittelyyn on tarjolla useita työkaluja ja muun muassa .NET-viitekehys sisältää kirjaston tähän tarkoitukseen.

Prosessien välinen kommunikaatio

Kommunikaatio pääohjelman ja testien välillä toteutettiin TCP-yhteydellä localhost-osoitteen kautta. Kriteereinä protokollan valitsemiseen olivat luotettavuus, yksinkertaisuus ja useiden ohjelmointiympäristöjen tuki. TCP-yhteyksien käsittelyyn on olemassa valmiit kirjastot kaikissa merkittävissä ohjelmointikielissä, sekä LabVIEW-ohjelmointiympäristössä. Sen etuina pidettiin myös luotettavuutta, sillä TCP-yhteydellä lähetetyt paketit saapuvat perille aina oikeassa järjestyksessä ja kadonneet paketit lähetetään uudelleen. [25, s. 4].

Kommunikaatiota varten määriteltiin protokolla, johon lisättiin taulukon 4 mukaiset komennot. Komennot lähetetään XML-merkintäkieleen perustuvalla koodikielellä, jonka tarkka kuvaus on dokumentoitu liitteessä 2.

Pääohjelma toteuttaa kaikki protokollassa sille määritellyt tiedonlähetykseen ja vastaanotettavan tiedon käsittelyyn määritellyt toiminnot. Testin ei kuitenkaan ole pakko hyödyntää kaikkea tietoa, eikä edes toteuttaa kaikkia rajapinnan toimintoja. Taulukossa 4 listatuista komennoista kahdella tähdellä merkityt ovat sellaisia, joita ilman testi ei voi saada hyväksyttyä tulosta. Yhdellä tähdellä merkityt toiminnot eivät ole pakollisia, mutta ilman

niitä pääohjelma tulkitsee testin toimineen väärällä tavalla. Tuloksiin sillä ei kuitenkaan ole vaikutusta. Loput toiminnot ovat testin sisällä tapahtuvaa haarautumista varten.

Taulukko 4. Rajapinnan kautta jaettu informaatio ja sen suunta.

Data	**Suunta**
Pyyntö testausasetusten lähetykseen**	Testi → pääohjelma
Testattavan laitteen tiedot	Pääohjelma → testi
Testausparametrit ja raja-arvot	Pääohjelma → testi
Muut testaukseen liittyvät asetukset	Pääohjelma → testi
Mittaustulokset**	Testi → pääohjelma
Testin kokonaistulos**	Testi → pääohjelma
Vikakuvaus	Testi → pääohjelma
Testauslaitteiston alasajo suoritettu*	Testi → pääohjelma
* Jos tieto jää lähettämättä, tulkitsee järjestelmä testin toiminnan virheelliseksi ja varoittaa käyttäjää. ** Ilman kyseistä toimintoa testille ei voi saada hyväksyttyä tulosta.	

Sarjallistaminen

Objektin tallentamista tiedostoon kutsutaan sarjallistamiseksi ja sen tarkoitus on, että objekti voidaan myöhemmin rekonstruktoida aikaisempaan tilaan. Microsoftin .Net viitekehys sisältää valmiita toimintoja sarjallistamista ja rekonstruktointia varten [19, s. 710]. Työssä päädyttiin kuitenkin omaan toteutukseen siitä syystä, että objekteista halutaan tallentaa vain sellaista tietoa, joka koskee kyseessä olevaa tuotesarjaa yleisesti. Liitteessä 3 on dokumentoitu sarjallistamalla tallennettavan tiedon rakenne.

Sarjallistettavaa tietoa tarvitsevat luokat toteuttavat SaveAsXElement()-metodin tiedon tallentamista varten, sekä staattisen, uuden luokkaan kuuluvan olion palauttavan Load(XElement)-metodin. Yläluokkina käytettävät luokat toteuttavat sarjallistamisen muodostimella, joka saa parametrina sarjallistetut tiedot XML-elementin muodossa.

6.8 Testaustapojen toteutukset

Testit toteutettiin omina sovelluksinaan. Sovellukset noudattavat samaa kaavaa, jossa ensin pyydetään parametrit pääohjelmalta, suoritetaan testiin kuuluvat toimenpiteet ja lopuksi raportoidaan tulos takaisin pääohjelmaan.

Samanlaisena toistuvien ominaisuuksien uudelleenkäyttöä varten kehitettiin kirjasto, joka nimettiin Test_IPC:ksi. Kirjasto toteuttaa kaikkien rajapinnassa määriteltyjen komentojen lähettämisen sekä vastaanotettujen komentojen käsittelyn. Sen avulla testissä määritellään kaikki informaatio, jota testi toimiakseen tarvitsee, kuten raja-arvot, parametrit ja muut tiedot testattavasta laitteesta.

Ainoa asia, mitä Test_IPC-kirjastoa käytettäessä on pakko määritellä, jotta se toimisi oikein, on testin tukemien tuoteperheiden nimet. Muiden tietojen määrittely riippuu vain siitä, mitä testissä tarvitaan. Jos pääohjelma ei lähetä kaikkia testissä tarpeelliseksi määriteltyjä tietoja, ei kirjasto ikinä indikoi testauksen aloittamista mahdolliseksi.

Määriteltyjä arvoja voi myöhemmin kutsua niiden nimen perusteella ja hyödyntää testauksessa. Kirjastoon määritellyille mittauksille voi myös antaa mittaustuloksen, jolloin kirjasto hoitaa niiden palauttamisen pääohjelmaan.

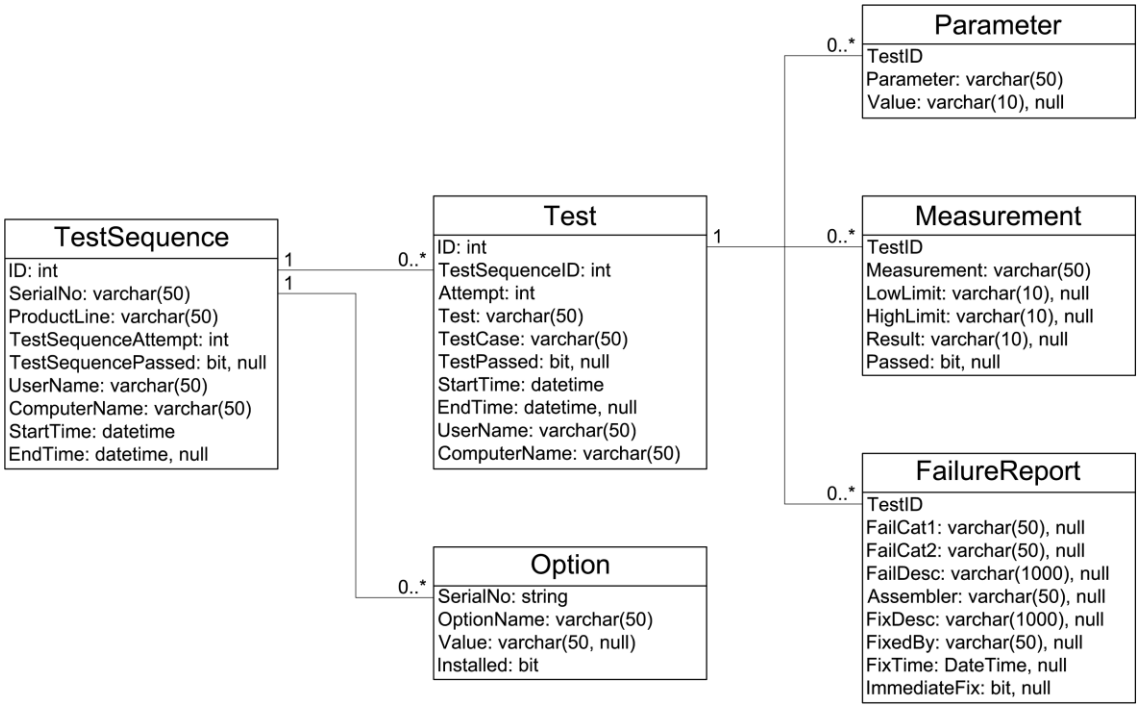
Kuva 2. Kuvakaappaus suojajohtimen jatkuvuustestistä, jonka aikana käyttäjä saa havainnolliset ohjeet mittauksen suorittamiseen.

Testien käyttökokemuksesta tehtiin mahdollisimman suoraviivainen poistamalla kaikki turhat vaiheet käyttäjän ja testin välisestä vuorovaikutuksesta. Esimerkiksi käyttäjältä pyydetään kuittausta vain, jos laitteisto pitää asettaa johonkin tilaan ennen testin aloittamista, eikä tilan asettamista voida automaattisesti havaita. Kuvan 2 suojajohtimen jatkuvuustestissä näytöllä olevat tekstit ja kuvat vaihtuvat sitä mukaa, kun käyttäjä suorittaa mittauksia. Viimeisen mittauksen jälkeen testi palauttaa tulokset ja sulkeutuu. Tulos näkyy vain pääohjelmassa, joka on suunniteltu näkyvän näytöllä testin kanssa samanaikaisesti.

6.9 Tietokantasuunnittelu

Koestuksesta kerättävät tiedot päätettiin varastoida relaatiotietokantaan. Vaihtoehtoja, kuten tietojen tallentamista tiedostoihin tai oliotietokantaan, ei liiemmin harkittu. Tiedostopohjaista ratkaisua ei haluttu luotettavuuden ja suorituskyvyn takia. Ongelmallista olisi ollut tiedostojen alati kasvava koko ja useammalta testauspisteeltä tulevat luku- ja kirjoituspyynnöt. Oliotietokannoista kehittäjällä taas ei ollut riittävää kokemusta niiden soveltuvuuden arviointiin. Relaatiotietokannan etuna oli myös, että sen tyyppisille tietokannoille on valmiita työkaluja automaattisten raporttien kehittämiseen.

Tietokantaan määriteltiin taulut kuvaamaan koestusprosessin vaiheita ja niihin liittyviä ominaisuuksia. Kuviossa 4 esitelty taulujen määrittely muistuttaa hieman järjestelmän luokkamäärittelyä. Tietokanta kuitenkin tukee vain sellaisten tietojen tallentamista, jotka koskevat suoritettua koestusprosessia, joten siitä puuttuu tiettyjä luokkia vastaavat taulut. Toisaalta tietokantaan lisätty FailureReport-taulu antaa aiheen pohtia, tulisiko luokkamäärittelystä löytyä vastaava luokka.



Kuvio 4. Tietokannan taulut ja niiden väliset relaatiot.

Kaikki taulujen väliset relaatiot ovat yksi moneen -tyyppisiä. Tällaisessa relaatiossa tiettyä objektia esittävä rivi jollakin tietokannan taululla voi olla omistajana useammalle objektille jollakin toisella taululla. Esimerkiksi testaussekvenssiin voi kuulua useita testejä, testiin useita mittauksia ja niin edelleen. Omistussuhde indikoidaan pääohjelmassa käytettyyn menettelyyn verrattuna päinvastaisesti siten, että kun testi kuuluu testaussekvenssiin, on testin taulussa kenttä sen testaussekvenssin ID:lle, jolle testi kuuluu.

Tietokannan tauluista TestSequence-taulu on hierarkiassa korkeimmalla. Sen tehtävä on pitää kirjaa testattujen laitteiden sarjanumerosta ja niiden kelpuutuksesta. Option-taulu kuvaa taas laitteen konfiguroitavaia ominaisuuksia ja siihen asennettuja lisävarusteita, ja yhdessä TestSequence-taulun kanssa niihin kerättyjen tietojen perusteella on mahdollista kertoa, minkälainen testattu laite oli.

Jokaisesta kerrasta, kun laitteen testaus aloitetaan alusta, luodaan uusi rivi TestSequence-tauluun. TestSequence-taulu ylläpitää laskuria, joka inkrementoituu aina kun samalla sarjanumerolla tunnettua laitetta yritetään testata uudelleen ja kertoo, montako kertaa laitetta on yritetty koestaa.

Jokainen TestSequence-taulun rivi toimii omistajana niille testeille, jotka sen aikana suoritettiin. Sellaiset testit, jotka jätettiin suorittamatta joko prosessin keskeytymisen tai aikaisemmalla kerralla saadun tuloksen hyödyntämisen takia, eivät sille kuulu, eikä niille edes luoda uutta riviä koestuksen aikana. Myös Test-taululla on laskuri, joka kertoo, montako kertaa samaa testiä on laitteelle yritetty suorittaa.

Measurement- ja Parameter-tauluille tallennetaan tiedon parametreista ja raja-arvoista, joilla testi on suoritettu, sekä siitä saaduista mittaustuloksista. Ne kuuluvat jollekin testille ja ovat sidottuina tiettyyn testauskertaan, koska on mahdollista, että parametreja olisi muutettu testauskertojen välillä.

Poikkeamaraportit sekä korjaukset tallennetaan FailureReport-taululle. Jokainen rivi sisältää sekä kuvauksen sekä viasta että korjaustoimenpiteistä. Myös poikkeamaraportti kuuluu testille, jotta saadaan tarkemmin missä tilanteessa poikkeama on havaittu. Valittu

relaatio mahdollistaisi myös useamman poikkeaman raportoimisen yhden testaussekvenssin aikana, vaikka järjestelmässä onkin päätetty estää prosessin eteneminen jokaisesta poikkeamasta.

7 Toimintakuvaus

7.1 Testattavien laitteiden määrittely

Järjestelmän käynnistyksessä muodostetaan tiedostoista lataamalla jokaista tuotesarjaa ja niihin kuuluvia testejä esittävät oliot. Sama toimenpide tehdään aina ennen uuden koestusprosessin aloittamista sen varmistamiseksi, että muistiin ei jää edellisestä ajosta sellaisia tietoja, jotka voisivat vääristää tuloksia.

Tuotesarjaa esittävä DUT-olio muuttuu yksilöä edustavaksi olioksi kun sille määritellään sarjanumero ja sen kaikille konfiguroiduille ominaisuuksile ja lisävarusteille on määritelty validi tila. Tämän jälkeen on mahdollista muodostaa sille testaussekvenssi. Sarjanumeron syöttämisen jälkeen järjestelmä etsii automaattisesti tietokantaan mahdollisesti aikaisemmin tallennetut tiedot laitteeseen asennetuista optioista ja tietojen löytyessä asettaa optioille samat arvot.

7.2 Testaussekvenssin generointi

Testaussekvenssiä muodostettaessa tarkistetaan jokaisesta tuoteryhmälle kuuluvasta testistä, täyttyvätkö sille asetetut ehdot. Jos ehdot täyttyvät, käydään testille kuuluvia testitapoja yksi kerrallaan läpi, kunnes testattavan laitteen optioita vastaava tapaus löytyy. Sama tehdään vielä testitapauksen parametriarvoille.

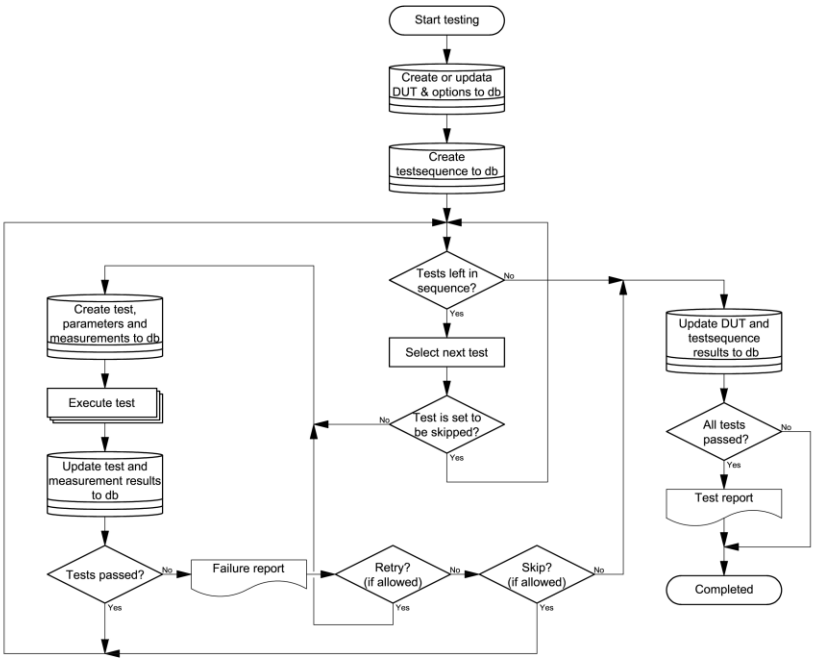
Jos tuotteelle ei ole määritelty yhtään testiä tai järjestelmä ei kykene löytämään joko testattavan laitteen konfiguraatioon soveltuvaa testitapausta tai sen parametriarvoja, ei laitteelle voi suorittaa asianmukaista testausta. Tässä tapauksessa eteneminen on kokonaan estetty ja virheellisestä konfiguraatiosta viestitään käyttäjälle virheilmoituksella.

Onnistuneen testaussekvenssin muodostamisen jälkeen haetaan tietokannasta edellisen samaa laitetta koskevan koestusyrityksen järjestysnumero. Vastaava järjestysnumero haetaan myös jokaiselle sekvenssiin kuuluvalle testille, minkä lisäksi niille haetaan tuoreimmat testaustulokset. Testien perusteella haetaan vielä jokainen poikkeamaraportti, jota ei ole merkitty käsitellyksi.

Poikkeamaraportit annetaan käyttäjälle täydennettäväksi, eikä poikkeaman aiheuttaneita testejä voi suorittaa ennen kuin raporttiin on kirjattu korjaustoimenpiteet. Tämän jälkeen kysytään vielä jokaisen kertaalleen hyväksytysti suoritetun testin ohittamisesta tai uudelleen suorittamisesta.

7.3 Testaus

Testaus etenee kuviossa 5 esitetyn kaavion mukaan. Testaussekvenssin ajo alkaa laitetta koskevien alkuarvojen tallentamisella tietokantaan, mistä siirrytään käsittelemään testit yksi kerrallaan. Ennen testin aloittamista tallennetaan sitä koskevat alkuarvot, ja suorittamisen jälkeen päivitetään tulokset. Testien loppuessa tai prosessin muutoin keskeytyessä, päivitetään aiemmin luotuihin testaussekvenssin tietoihin prosessin läpäisyä koskeva informaatio.



Kuvio 5. Testaussekvenssin suorittaminen.

Alkuarvojen tallennus

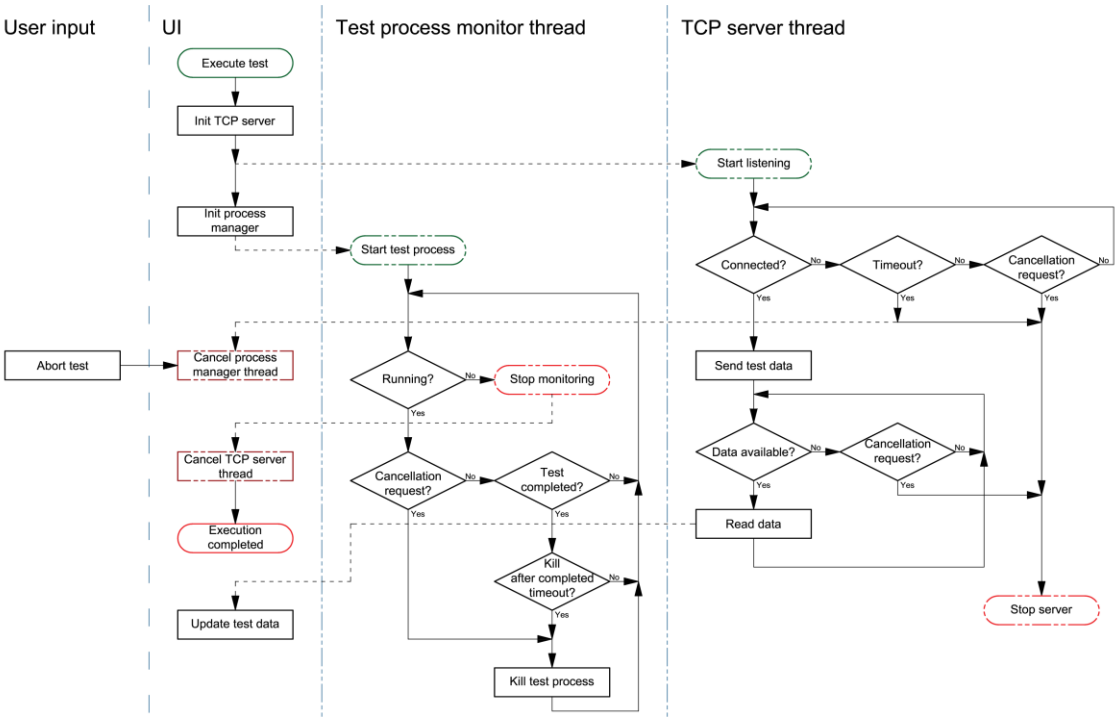
Ensimmäinen vaihe testauksessa on luoda tai päivittää tietokannan TestSequence-tauluun testaussekvenssiä koskeva rivi, sekä Option-tauluun testattavan laitteen konfiguraatio. Jokaista testiä ennen taas luodaan tietokannan Test-tauluun uusi suoritettavaa testiä koskeva rivi, sekä Parameter- ja Measurement-tauluihin rivit testiin kuuluville parametreille ja mittauksille raja-arvoineen.

Testin suorittaminen

Testin suorittamisen aikana pääohjelma toimii kolmessa säikeessä, jotka on varattu käyttöliittymälle, testiä suorittavan prosessin tilan monitorointiin sekä prosessien väliseen tietoliikenteeseen.

Pääohjelma toimii testin suorituksen aikana likimain kuviossa 6 esitetyn kaavion mukaisesti. Kaaviota on yksinkertaistettu siten, että siinä ei ole kuvattu TCP-palvelimen alustuksessa ja prosessin käynnistyksessä mahdollisesti tapahtuvien virheiden käsittelyä. Mallista karsituissa virhetilanteissa järjestelmän sisäinen virheidenkäsittely määrittelee

testille vikakuvauksen ja järjestelmä siirtyy prosessimallin mukaisesti tulosten tallennukseen ja poikkeamaraporttiin.



Kuvio 6. Yksinkertaistettu malli pääohjelman suorituksesta ja siihen liittyvien toimintojen jakautumisesta eri säikeisiin.

Heräte testin suorittamisen aloittamiseen saadaan edellisen testin valmistumisesta tai käyttäjältä kun kyseessä on sekvenssin ensimmäinen testi. Ensimmäiseksi järjestelmä alustaa TCP-palvelimen ja käynnistää TCP-yhteyttä monitoroivan säikeen, minkä jälkeen se alustaa testiprosessia monitoroivan säikeen.

Testiprosessia monitoroiva säie käynnistää testin erillisessä prosessissa ja kuvauksensa mukaisesti valvoo sen tilaa sekä tarvittaessa pakottaa sen pysähtymään. Monitoroimalla testiä ajavan prosessin tilaa eli sitä, onko prosessi vielä käynnissä, tiedetään, milloin testi on suoritettu loppuun ja voidaan siirtyä tulosten analysoimiseen. Prosessin keskeytystä käytetään, kun käyttöliittymästä saadaan keskeytyskäsky. Prosessi pakotetaan myös pysähtymään, kun se ei toimi oletetulla tavalla. Epänormaalina toimintana pidetään sitä, että

- testi ei pyri muodostamaan yhteyttä pääohjelmaan annetussa aikaikkunassa
- testi on raportoinut suorituksen päättymisestä, mutta prosessi on määritellyn aikarajan jälkeen yhä käynnissä.

Tulosten analysointi ja tallentaminen

Testi katsotaan läpäistyksi kun kaikki seuraavat ehdot täyttyvät:

- Kaikki määritellyt mittaustulokset on palautettu hyväksyttyinä.
- Mittaustulokset ovat raja-arvojen sisällä (kun raja-arvot on määritelty).
- Testin palauttama kokonaistulos on hyväksytty.

Mittaustuloksia jatkojalostetaan vain sen verran, että jokaisesta mittaustuloksesta tarkistetaan testin palauttama hyväksymistieto, sekä raja-arvojen sisällä pysyminen, joiden täytyy olla kunnossa hyväksytyn testaustuloksen saamiseksi. Pääohjelma sisältää siis ylimääräisen tarkastelun raja-arvoissa pysymiseen ja voi muuttaa läpäistynä palautetun testaustuloksen hylätyksi, mutta eivät toiseen suuntaan.

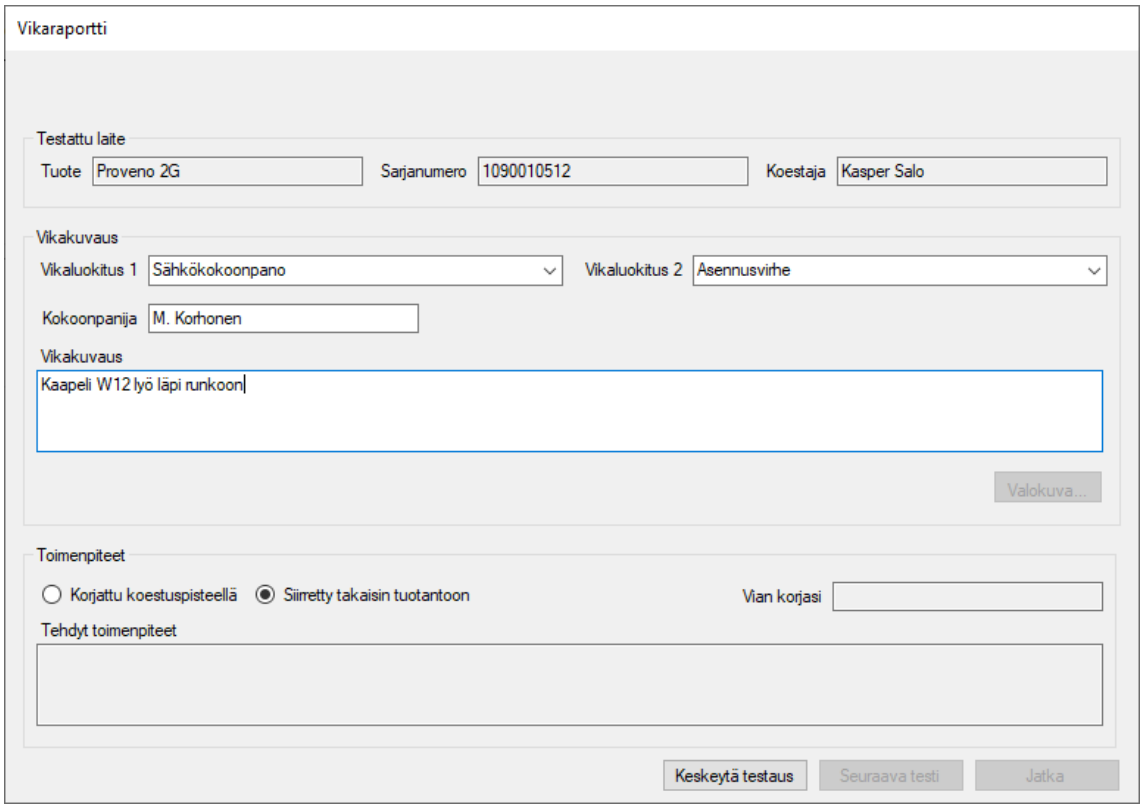
Kaikki testin suorittamisessa epänormaaliksi määritelty toiminta ei johda hylättyyn tulokseen. Hylkäys tapahtuu ainoastaan, kun palautettujen arvojen perusteella ei voi määritellä täyttikö testattava laite sille annetut kriteerit. Esimerkiksi testauslaitteiston alasajon epäonnistuminen ei aiheuta hylkäystä, mutta sen takia käyttäjälle annetaan varoitus tapahtuneesta ja kehotus resetoida laitteisto manuaalisesti.

Riveille, jotka luotiin valmiiksi tietokannan Test- ja Measurement-tauluihin, päivitetään testin suorittamisen jälkeen. Measurement-tauluun päivitetään testissä mitatut arvot sekä tieto niiden hyväksyttävyydestä, Test-tauluun puolestaan tieto koko testin läpäisystä ja päättymisajasta. Hylätty testaustulos aiheuttaa poikkeamaraporttiin siirtymisen.

Poikkeamien raportointi

Jokainen epäonnistunut testi vaatii poikkeamaraportin täydentämistä. Heti rutiiniin siirryttäessä luodaan FailureReport-tauluun valmiiksi uusi rivi tulevaa raporttia varten. Tä-

män jälkeen käyttäjälle esitetään kuvan 3 mukainen uusi ikkuna, automaattisesti kerättyjen tietojen täydentämistä ja korjaamista varten, jotka päivitetään käsittelyn jälkeen samaiselle riville. Poikkeamaraportti-ikkuna sisältää vaihtoehdot koestusprosessin mukaiseen haarautumiseen eli testin uudelleenyritykseen, ohittamiseen ja koestusprosessin keskeyttämiseen.



Kuva 3. Jännitekokeen epäonnistumisen seurauksena esitetty poikkeamaraportti.

Loppuraportti

Lopullinen tieto koko testausprosessin hyväksyttämisestä sekä laitteen kelpoisuudesta päivitetään TestSequence-tauluun vasta kun koko testausprosessi on saatu päätökseen. Testaussekvenssin rivi saa läpäisty merkinnän vain, jos kaikista testeistä on olemassa sellainen hyväksytty tulos, jota myöhemmät testit eivät ole mitätöineet.

Tietokantaan tallennettavien tietojen lisäksi luodaan koestuksen läpäisseille laitteille tekstimuotoinen koestusraportti verkkolevylle tallennettavaksi. Se sisältää jokaisesta testistä viimeisimmän tuloksen parametreineen, mittauksineen, sekä kellonaikoineen. Raportti sisältää myös tiedot testatun laitteen konfiguraatiosta.

8 Yhteenveto

Koska jo valmistettavien laitteiden suunnitteluvaiheessa on päätetty toteuttaa pienjännitedirektiivissä vaadittu käyttäjän suojaaminen sähkön vaaroilta noudattamalla standardia EN 60204, on samaisessa standardissa määritellyt testit pakko suorittaa. Sama koskee painelaitedirektiivin liitteen I kohtaa 3.2. Asiakkaat voivat joissakin tapauksissa, kuten aluksille asennettaviin laitteisiin, vaatia tiukennuksia standardin määrittämiin raja-arvoihin. Lisäksi laitteen tulee toimia asiakkaan odottamalla tavalla. Organisaatiolle asiakkaan vaatimusten täyttämisen lisäksi tärkeää on oman toiminnan kehittäminen. Toimintaa ja kehitystarpeita analysoimaan voidaan kehittää mittareita ja standardisoidut laatujärjestelmät, kuten ISO 9001, sisältävät sellaisia vaatimuksia, joiden täyttämistä voidaan edesauttaa loppukoestuksessa.

Työtä aloittaessa todettiin keittolaitelinjalla loppukoestukseen käytettävä laitteisto ominaisuuksiltaan riittäväksi täyttämään lainsäädännölliset vaatimukset. Samalla todettiin, että se on tuettu varaosien ja ajurien osalta. Näistä syistä päätettiin kehittää laitteistolle uusi ohjelmisto määriteltyjen vaatimusten pohjalta. Vaatimuksia uudelle ohjelmistolle olivat skaalautuvuus erityyppisten testattavien laitteiden kattamiseksi, laajennettavuus käyttämään kokonaan erilaista mittalaitteistoa, sekä inhimillisten tekijöiden vaikutuksen pienentäminen.

Vaatimukset täytettiin skaalautumisen osalta jakamalla ohjelmisto eri tehtäviä suorittaviin osiin. Testausprosessia hallitsemaan tehtiin pääohjelma, josta haaraudutaan tarpeen mukaan yhden testin suorittaviin aliohjelmiin. Uusien testien ohjelmointia pyrittiin nopeuttamaan luomalla niille kirjasto pääohjelman kanssa kommunikointia varten. Inhimillisten tekijöiden vaikutusta tuloksiin pienennettiin toteuttamalla testit siten, että ne sisältävät kuvalliset toimintaohjeet mittausten suorittamiseen. Automaattisten raporttien generoimista varten päätettiin kaikki järjestelmän keräämät tiedot tallentaa tietokantaan.

Työn aikana tilaajan näkemys tarkentui ja laajeni niin, että kaikkien ideoiden toteuttaminen työn puitteissa ei enää ollut mielekästä, eikä aikataulullisesti mahdollista. Niitä huomioitiin kuitenkin kehitysvaiheessa suunnittelemalla jotkin ominaisuudet siten, että järjestelmän jatkokehitys haluttuun suuntaan olisi mahdollista. Osa uusista ominaisuuksista on sellaisia, että järjestelmä jo luonnostaan on suunniteltu tukemaan niitä.

Puuttuvien laitteiden tuominen järjestelmään

Ajatuksena on ajaa uusi järjestelmä ylös vaiheittain siten, että aloitetaan valitun tuoteperheen yhdestä laiteversiosta ja jatketaan kehittämällä testit kaikille eri tavalla konfiguroiduille malleille. Lopulta on tarkoitus laajentaa järjestelmä siten, että se kattaa myös muut keittolaitelinjalla valmistettavat tuoteperheet, eikä mikään tekninen seikka estä ohjelmiston käyttöönottoa myös muilla tuotantolinjoilla.

Automatisoinnin kehittäminen uusilla mittalaitteilla ja antureilla

Samalla kun järjestelmään lisätään uusia testejä, on uudet testit mahdollista kehittää kokonaan uusille mittalaitteille. Projektin aikana pohdittiin sellaisten mittausten siirtämistä järjestelmään, jotka nyt tehdään käsin. Myös kokonaan uusien asioiden mittaamista spekuloitiin. Esimerkiksi ruokaa jäähdytettäessä padan höyryvaippaan juoksutetaan kylmää vettä, jonka virtausnopeus korreloi lähes suoraan jäähdytysnopeuden kanssa.

Kerätyn datan vieminen muihin järjestelmiin

Yrityksessä käytetään QlikView nimistä liiketoimintatiedon hallintaohjelmistoa. Ohjelmiston tehtävänä on kerätä tietoa yrityksen käyttämistä järjestelmistä ja sen avulla on mahdollista luoda kerätyn tiedon pohjalta automaattisia raportteja. Näin ollen luonteva ratkaisu olisi viedä myös loppukoestuksen automaattiset raportit samaan järjestelmään.

Suorituskykymittaukset

Koska valmistettavista laitteista on konfiguroitavia ominaisuuksia ja lisävarusteita vaihtamalla mahdollista luoda lukuisia erilaisia variaatioita, on kattavien suorituskykymittausten tekeminen kaikille kombinaatioille käytännössä mahdoton urakka. Tarkkaa tietoa laitteiden suorituskyvystä ei mikään määräys velvoita selvittämään, mutta monille asiakkaille se on erityisarvoisen tärkeää ja voisi olla myynnille merkittävä kilpailuetu.

Järjestelmään visioitiin sellaista mahdollisuutta, että tuoteryhmälle voisi määrittää jotakin ominaisuutta mittaavia testejä, jotka suoritettaisiin koestuksen jälkeen vain kerran määrätyn tyyppiselle laitteelle. Järjestelmälle siis kerrottaisiin mitkä valinnat vaikuttavat tähän ominaisuuteen ja vastaavan konfiguraation ilmestyessä testattavaksi, ehdotettaisiin suorituskykymittauksen tekemistä.

Uusintatesti huollon yhteydessä

Huollon yhteydessä laitteille on joka kerta syytä tehdä jonkinlainen tarkistus. Kehitetyn järjestelmän arvioitiin joiltakin osin soveltuvan myös tähän tarkoitukseen. Toteutusta varten pitäisi kehittää sellaiset testaustavat, jotka ohjeistaisivat huoltomiestä suorittamaan tarvittavat mittaukset siirrettävällä manuaalisesti käytettävällä mittalaitteella. Tällöin tiedot pitäisi tallentaa johonkin toiseen tietokantaan, sillä tehtaalla saatua hyväksyttyä koestusraporttia ei missään tilanteessa saa ylikirjoittaa tuotteen toimituksen jälkeen.

Toinen vaihtoehto verrata huoltotyön yhteydessä mitattuja arvoja koestusprosessissa saatuihin on lukea ne suoraan koestusraportista. Tämä vaihtoehto edellyttäisi koestusraportin helpon saatavuuden, mikä olisi hoidettavissa esimerkiksi liiketoimintojen hallintajärjestelmällä. Tämä lähestymistapa ei kuitenkaan mahdollistaisi kentällä mitattujen arvojen tallentamista, eikä niiden kehityksen seuraamista.

Dokumenttien liittäminen osaksi koestusraporttia

Vaikka järjestelmän tarkoituksena on kerätä ja tallentaa kaikki laitteen kelpoisuuden todentamiseen tarvittava tieto sähköisesti, kulkee tuotantolinjalla laitteen mukana sellaisia

dokumentteja, joiden täydellistä digitalisointia järjestelmä ei voi hoitaa. Tällaisia ovat esimerkiksi asiakasta varten räätälöityjen tai muuten hyvin harvinaisten tuoteversioiden piirustukset. Dokumenttien säilyttämisen ja saatavuuden parantamiseksi visioitiin mahdollisuutta niiden skannaamiseen ja liittämiseen osaksi koestusraporttia.

Lähteet

1 Metos Kitchen Intelligence on enemmän kuin älykäs keittiö. Verkkoaineisto. Metos Oy Ab. <https://www.metos.com/page.asp?pageid=2,1&languageid=FI&title=Metos%20yrityksen%C3%A4>. Luettu 6.11.2017.

2 Proveno 2G, Viking, Culino vaatimustenmukaisuusvakuutus. Metos Oy Ab. 2016. Yrityksen sisäinen asiakirja.

3 Sähkölaitteen valmistus, maahantuonti ja myynti. 2014. Verkkoaineisto. Tukes. <http://www.tukes.fi/Tiedostot/sahko_ja_hissit/esitteet_ja_oppaat/sahkolaitteiden_valmistus_maahantuonti_ja_myynti.pdf>. Luettu 29.7.2017.

4 Euroopan parlamentin ja neuvoston direktiivi 2014/68/EU painelaitteiden asettamista saataville markkinoilla koskevan jäsenvaltioiden lainsäädännön yhdenmukaistamisesta. Euroopan unionin virallinen lehti 27.6.2014, s. 164–259.

5 Keto, Rainer. 2010. Proveno 2G riskiarviointilomake. Yrityksen sisäinen dokumentti.

6 Koneista ja direktiivin 95/16/EY muuttamisesta annetun Euroopan parlamentin ja neuvoston direktiivin 2006/42/EY täytäntöönpanoon liittyvä komission tiedonanto. Euroopan unionin virallinen lehti 9.3.2018, s. 1–85.

7 Komission tiedonanto tietyllä jännitealueella toimivien sähkölaitteiden asettamista saataville markkinoilla koskevan jäsenvaltioiden lainsäädännön yhdenmukaistamisesta annetun Euroopan parlamentin ja neuvoston direktiivin 2014/35/EU täytäntöönpanosta. Euroopan unionin virallinen lehti 14.9.2018, s. 4–93.

8 Willman, Kari. 2015. METOS Oy Ab:n, Suomen tuotetehtaiden sekä Metos Marinen Sähkötyöohjeis-tus & tekninen perusopas. Rev. 4.0. Yrityksen sisäinen dokumentti.

9 General Information for the Rules and Regulations for the Classification of Ships. 2011. Lloyd’s Register.

10 Euroopan parlamentin ja neuvoston direktiivi 2006/42/EY koneista ja direktiivin 95/16/EY muuttamisesta. Euroopan unionin virallinen lehti 9.6.2006, s. 24–86.

11 Euroopan parlamentin ja neuvoston direktiivi 2014/35/EU tietyllä jännitealueella toimivien sähkölaitteiden asettamista saataville markkinoilla koskevan jäsenvaltioiden lainsäädännön yhdenmukaistamisesta. Euroopan unionin virallinen lehti 29.3.2014, s. 357–374.

12 SFS-EN 60204-1:2006. Koneturvallisuus. Koneiden sähkölaitteisto. Osa 1. Yleiset vaatimukset. Suomen standardoimisliitto.

13 SFS-EN 60335-1:2013. Kotitalouksiin ja vastaaviin käyttöihin tarkoitetut sähkölaitteet. Turvallisuus. Osa 1: yleiset vaatimukset. Suomen standardoimisliitto.

14 FST-110 10/100mA High Voltage Tester. Käyttöohe. Finero Oy. Verkkoaineisto. <http://finero.fi/tuki/FST//FST-110-High-Voltage-Tester.pdf>. Luettu 10.7.2017.

15 Vip Energy. Käyttöohje. Elcontrol Energy Net. Verkkoaineisto. <http://archive.elcontrol-energy.net/download/vipenergy_man_eng.pdf>. Luettu 14.7.2017.

16 SFS-EN ISO 9001: 2015. Laadunhallintajärjestelmät. Vaatimukset. Suomen standardoimisliitto.

17 Through put yield (TPY). Verkkoaineisto. Six Sigma Material <http://www.six-sigma-material.com/Throughput-Yield.html>. Luettu 31.5.2017

18 Bell, Douglas. 2015. Software Engineering for Students: A programming Approach. Fourth Edition. Essex: Addison-Wesley.

19 Watson ym. 2010. Beginning Visual C# 2010. Indianapolis: Wiley Publishing, Inc.

20 Albahari, Joseph. 2011. Threading in C#. Verkkoaineisto. <http://www.albahari.info/threading/threading.pdf>. Luettu 7.4.2019.

21 Britton, Chris. 2015. Designing the Tequirements: Building Applications that the User Wants and Needs. Boston: Addison-Wesley.

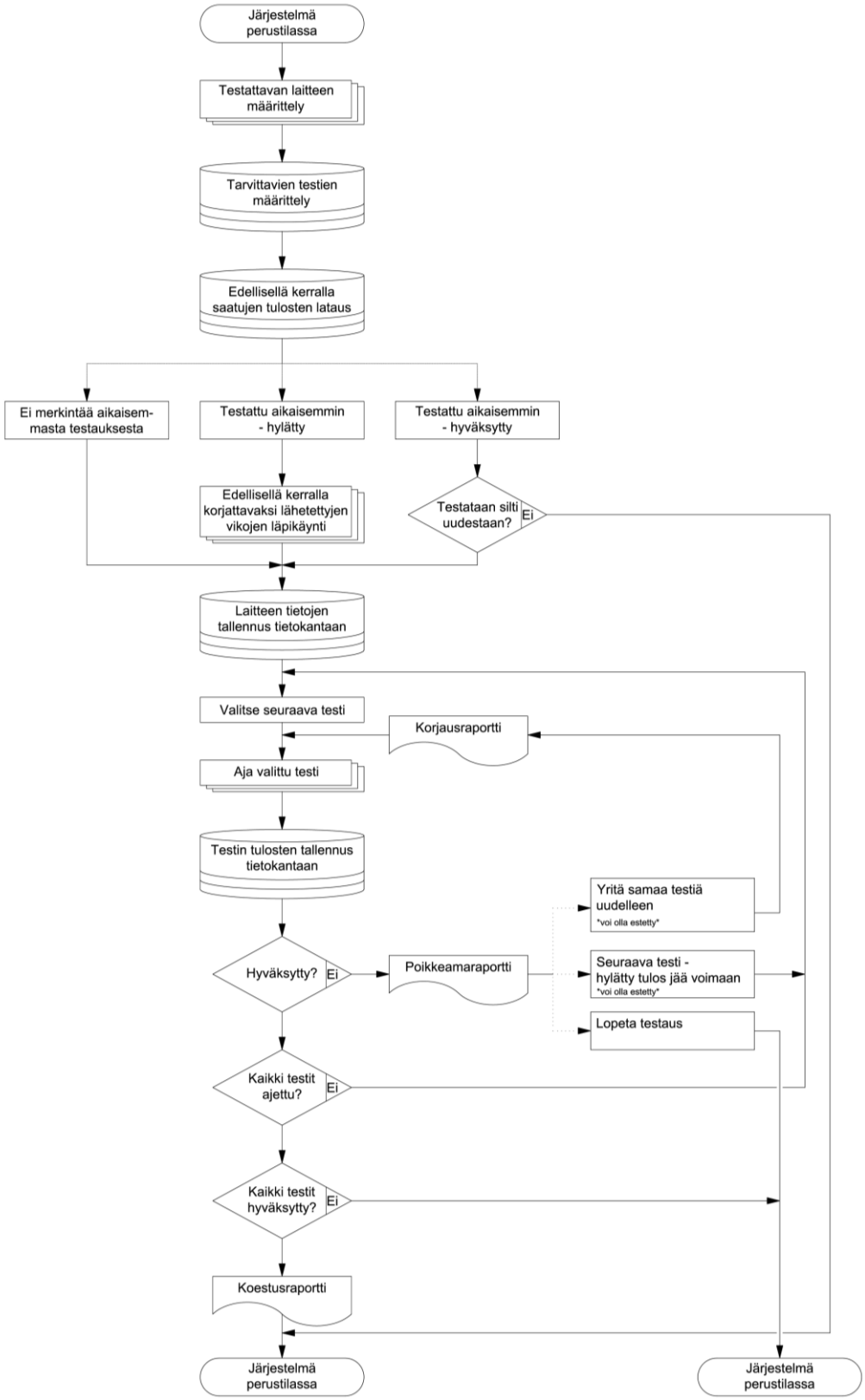
22 How to design a great user experience for desktop applications. Verkkoaineisto. MSDN. <https://msdn.microsoft.com/en-us/library/windows/desktop/dn742462(v=vs.85).aspx>. Luettu 20.6.2017.

23 Active Directory Fast Start: A Quick Start Guide for Active Directory. Smart Brain Training Solutions. 2014. Yhdysvallat: Createspace Independent Publishing Platform.

24 XML Essentials. 2015. Verkkoaineisto. W3C. <https://www.w3.org/standards/xml/core>. Luettu 6.11.2017.

25 RFC 793. Transmission Control Protocol. Protocol Specification. DARPA Internet program. Verkkoaineisto. <https://tools.ietf.org/html/rfc793#>. Luettu 7.10.2017.

Koestuksen prosessikaavio



Pääohjelman ja testien välinen rajapinta – määrittely ja esimerkkejä

Pääohjelma ja testi keskustelevat XML:ään perustuvalla merkintäkielellä. Komennot lähetetään aina saman juurielementin sisällä. Yksi juurielementti voi sisältää useita komentoja. Juurielementin nimi on TestCommand.

```
<TestCommand>
  <!-- Komennot lisätään tähän !-->
</TestCommand>
```

Komennot

Parametrikysely

Testi pyytää pääohjelmaa toimittamaan tiedot testattavasta laitteesta, siihen asennetuista optioista ja parametreista, joilla kyseinen testi suoritetaan.

```
<TestCommand>
  <RequestParameters/>
</TestCommand>
```

Testattava laite

Pääohjelma ilmoittaa testattavan laitteen mallin sekä konfiguroituja ominaisuuksia ja lisävarusteita koskevat valinnat.

```
<TestCommand>
  <DUT ProductLine="[Tuoteperhe, johon testsattava laite kuuluu]">
    <Option Name="[Nimi 1]" Installed="[T/F]">[Tarkennus]</Option>
    <Option Name="[Nimi n]" Installed="[T/F]">[Tarkennus]</Option>
  </DUT>
</TestCommand>
```

Yleiset asetukset

Pääohjelma ilmoittaa testille yleisiä testaukseen liittyviä asetuksia, jotka ovat samat koko testaussekvenssin ajan ja joita tyypillisesti useampi testi tarvitsee.

```
<TestCommand>
  <Setting Name="[Asetuksen nimi]">
    <Parameter Name="[Asetuksen ominaisuus 1]">[Arvo]</Parameter>
    <Parameter Name="[Asetuksen ominaisuus n]">[Arvo]</Parameter>
  </Setting>
</TestCommand>
```

Parametrit

Pääohjelma ilmoittaa testille suoritettavan testin parametrit.

```
<TestCommand>
  <Parameter Name="[Parametrin 1 nimi]">[Arvo]</Parameter>
  <Parameter Name="[Parametrin n nimi]">[Arvo]</Parameter>
</TestCommand>
```

Raja-arvot

Pääohjelma ilmoittaa raja-arvot testiin kuuluvia mittauksia varten

```
<TestCommand>
  <Measurement Name="[Mittauksen nimi]">
    <LowLimit>[alaraja-arvo]</LowLimit>
    <HighLimit>[yläraja-arvo]</HighLimit>
  </Measurement>
</TestCommand>
```

Mittaustulokset

Testi palauttaa mitatun arvon, sekä tiedon mitatun arvon kelvollisuudesta.

HUOM. Pelkkä raja-arvoissa pysyminen ei ole riittävä indikaatio onnistuneesta mittauksesta.

```
<TestCommand>
  <Measurement Name="[Mittauksen nimi]">
    <Result Passed="[T/F]">[Mittaustulos]</Result>
  </Measurement>
</TestCommand>
```

Testin tulos

Testi ilmaisee tiedon suoritetun testin läpäisystä. Sekä yksittäisten mittaustulosten, että koko testin tulos on palautettava.

```
<TestCommand>
  <OverallResult Passed="[T/F]" />
</TestCommand>
```

Vikakuvaus

Hylätyn testin tapauksessa voidaan palauttaa kuvaus viasta.

```
<TestCommand>
  <OverallResult Passed="[T/F]">
    <FailCat1>[Vikaluokitus, yläkategoria]</FailCat1>
    <FailCat2>[Vikaluokitus, alakategoria]</FailCat2>
    <FailDesc>[Tarkka vikakuvaus]</FailDesc>
  </OverallResult>
</TestCommand>
```

Testin alasajo

Testi ilmaisee, että jälkikäsittelytoimet, kuten laitteiston alasajo, ovat päättyneet. Tämän jälkeen testi voi sulkeutua aiheuttamatta odottamattoman sulkeutumisen aiheuttamaa virheilmoitusta.

```
<TestCommand>
  <ExitSuccess/>
</TestCommand>
```

Esimerkki rajapinnan toiminnasta

Saatuaan yhteyden pääohjelmassa pyörivään palvelimeen, lähettää testi parametrikyselyn. Parametrikysely on suoritettava, vaikka testi ei hyödyntäisi lähetettäviä tietoja.

```
<TestCommand>
  <RequestParameters/>
</TestCommand>
```

Pääohjelma vastaa lähettämällä tiedot tuotteesta, asetuksista sekä testin parametreistä ja raja-arvoista. Kaikki tiedot lähetetään yhden juurielementin sisällä.

```
<TestCommand>
  <DUT ProductLine="Proveno 2G">
    <Option Name="Koko" Installed="True">40</Option>
    <Option Name="Lämmitystapa" Installed="True">Sähkö</Option>
    <Option Name="Käyttöjännite" Installed="True">3/N/PE 400/230V</Option>
    <Option Name="Jäähdytys" Installed="True">C3</Option>
    <Option Name="Ajastus" Installed="False"></Option>
    <Option Name="Vesiautomatiikka" Installed="False"></Option>
    <Option Name="EasyRun ohjelmointi" Installed="False"></Option>
  </DUT>
  <Setting Name="Pistorasia">
    <Parameter Name="Virta (A)">32</Parameter>
    <Parameter Name="Vaiheet">3</Parameter>
  </Setting>
  <Parameter Name="Koestusjännite (V)">10</Parameter>
  <Parameter Name="Koestusvirta (A)">10</Parameter>
  <Parameter Name="Koestusaika (s)">5</Parameter>
  <Measurement Name="Resistanssi (m Ohm)">
    <LowLimit>0</LowLimit>
    <HighLimit>100</HighLimit>
  </Measurement>
</TestCommand>
```

Testi palauttaa mittaustulokset ja tiedon testin onnistumisesta. Tiedot yksittäisistä mittaustuloksista voidaan lähettää joko yksitellen tai kootusti lopullisen testaustuloksen kanssa, mutta lopullista tulosta ei saa lähettää ilman, että osatulokset ovat selvillä.

```
<TestCommand>
  <Measurement Name="Resistanssi (m Ohm)">
    <Result Passed="True">24</Result>
  </Measurement>
  <OverallResult Passed="true"/>
</TestCommand>
```

Testi palauttaa tiedon onnistuneesta suorituksesta ennen sulkeutumistaan.

```
<TestCommand>
  <ExitSuccess/>
</TestCommand>
```

Testausjärjestelmän konfiguraatiotiedostojen rakenne

Pääohjelma säilyttää testattaviin tuotteisiin liittyvät tiedot XML kielellä koodattuna tallennettuna tiedostoihin. Oletusarvoisesti tiedostot tallennetaan polkuun

[päohjelman työhakemisto]\parameters.

Tiedot tuoteryhmien konfiguroitavista ominaisuuksista ja saatavilla olevista lisävarusteista, testeistä ja osa testitapausten tiedoista tallennetaan tiedostoon Options.xml. Testausmenetelmästä tallentuu vain nimi, valintasäännöt, sekä viittaus toiseen tiedostoon, johon loput sen tiedoista tallentuvat.

Testausmenetelmän lisätiedot sisältävän teidoston nimi muodostetaan liittämällä testin ja testausohjelman nimet toisiinsa siten, että nimien väliin tulee alaviiva ja loppuun lisätään ”_parameters.xml”. Tiedosto sisältää sen aliohjelman polun, joka menetelmän toteuttaa, sekä siihen liittyvät parametrit ja mittaustulokset muuttuvine arvoineen ja valintasääntöineen.

Tiedostojen rakenne

Juurielementti

XML kielen määrittelyn mukaisesti kaikki tiedot on tallennettava juurielementtiin. Juurielementin nimi on Config.

```
<Config>
  <!-- Konfiguraatio tallennetaan tähän -->
</Config>
```

Tuoteryhmät, konfiguroitavat ominaisuudet ja lisävarusteet

Tuoteryhmät tallennetaan ProductLine elementillä. Sen ainoa attribuutti kuvaa tuoteperheen nimeä ja sen sisältämiä lapsielementtiä ovat konfiguroitavia ominaisuuksia ja liäsvarusteita, testejä ja tuoteryhmäkohtaisia määrittelyjä kuvaavat elementit Option, Setting ja Test.

```
<ProductLine Name="Proveno 2G">
  <Option><!--(...)--></Option>
  <Setting><!--(...)--></Setting>
  <Test><!--(...)--></Test>
</ProductLine>
```

Konfiguroitavat ominaisuudet ja lisävarusteet

Konfiguroitavat ominaisuudet ja lisävarusteet tallennetaan Option elementtiin, jonka attribuutit ovat ominaisuuden tai lisävarusteen nimi (Name) ja teito siitä, onko kyseessä vapaasti valittava lisävaruste, vai kaikissa laitteissa pakosti oleva konfiguroitava ominaisuus (IsMandatory). Jos optiosta on olemassa erilaisia versioita, tallennetaan niiden kuvaukset Value-elementteihin. Value-elementin IsDefault attribuutilla järjestelmä saadaan tarjoamaan kyseistä vaihtoehtoa automaattisesti kun optio merkitään asennetuksi. Oheisessa esimerkissä koko on määritelty jokaiselle laitteelle kuuluvaksi ominaisuudeksi ja luettolosta löytyy saatavilla olevat vaihtoehdot. Jäähdytys on määritelty lisävarusteeksi ja saatavilla olevista vaihtoehdoista C2 versio oletusvaihtoehdoksi. Vesiautomatiikalle ei ole määritelty erilaisia versioita, joten se voi olla vain asennettuna tai ei asennettuna.

```
<Option Name="Koko" IsMandatory="True">
  <Value>100</Value>
  <Value>200</Value>
  <Value>300</Value>
  <Value>400</Value>
</Option>
<Option Name="Jäähdytys" IsMandatory="False">
  <Value IsDefault = "True">C2</Value>
  <Value>C3</Value>
  <Value>C5</Value>
</Option>
<Option Name="Vesiautomatiikka" IsMandatory="False" />
```

Haarautuminen

Kun objekti, jota elementti esittää, koskee testattavaa laitetta vain tiettyjen optioiden ollessa asennettuja, ilmaistaan tämä lisäämällä kyseiseen elementtiin ehdot määrittäviä Condition elementtejä. Tämä tarkoittaa niitä objekteja, jotka periytyvät järjestelmän ConditionalObject luokasta. Ehto voidaan määrittää toteutumaan kun jokin seuraavista toteutuu:

- konfiguroitavasta ominaisuudesta tai liävarusteesta on asennettu jokin tietyistä versioista
- liäsvaruste on asennettu, mikä tahansa verio kelpaa (ilmaistaan määrittelemällä kaikki saatavilla olevat versiot sallituiksi)
- lisävaruste ei ole asennettu.

Condition elementissä mainitaan attribuuttina määriteltävän option nimi ja se, tuleeko sen olla asennettuna. Elementin sisällä on tekstinä optiolle sallittu arvo. Jos optiota ei ole olemassa eri versioita, voi elementti sisältää pelkät attribuutit. Objektille voi määrittää

myös useista optioista koostuvia ehtoja, jolloin kaikkien niistä on toteuduttava. Loogista tai-operaattoria ei eri optioita koskevien ehtojen välillä tueta.

Alla olevassa esimerkissä testi on määritelty tehtäväksi vain nettotilavuudeltaan 300- ja 400-litraisille padoille, joihin jäähdytysoptio on asennettu, mutta vesiautomatiikkaa ei. Se, että mikä tahansa jäähdytysvaihtoehdoista saa olla asennettuna ehtojen toteutumiseksi, ilmaistaan listaamalla kaikki optiolle sallitut arvot eri Condition elementeissä.

```
<Test Name="Example"
      AllowRetry="True" AllowSkip="False"
      AllowContinueOnFail="False">
  <Condition Option="Koko" Installed="True">300</Condition>
  <Condition Option="Koko" Installed="True">400</Condition>
  <Condition Option="Jäähdytys" Installed="True">C2</Condition>
  <Condition Option="Jäähdytys" Installed="True">C3</Condition>
  <Condition Option="Jäähdytys" Installed="True">C5</Condition>
  <Condition Option="Vesiautomatiikka" Installed="False" />
</Test>
```

Ehtoja käsin muokkaamalla suoraan tiedostoissa, on mahdollista luoda sellaisia konfigurointeja, jotka eivät voi toteutua, tai pahimmassa tapauksessa toteutuvat väärässä tilanteessa. Esimerkiksi jokaiselle laitteelle kuuluvan konfiguroitavan ominaisuuden määrittäminen kielletyksi ja ei asennettu –tilan salliminen samaan aikaan, kuin jokin saman option versioista voi olla asennettuna, aiheuttavat järjestelmässä toimintavirheen.
Oheisessa esimerkissä virheellisesti määriteltyjä ovat koko, koska se on kaikille laitteille kuuluva ominaisuus, ja jäähdytys, koska samaan aikaan sallittuna ovat C2 versio ja jäähdytyksen puuttuminen.

```
<Test Name="Jäähdytystesti" AllowRetry="True" AllowSkip="False"
      AllowContinueOnFail="False">
  <Condition Option="Jäähdytys" Installed="True">C2</Condition>
  <Condition Option="Jäähdytys" Installed="True">C3</Condition>
  <Condition Option="Jäähdytys" Installed="True">C5</Condition>
</Test>
```

Tuoteryhmäkohtaiset asetukset

Tuoteryhmäkohtaiset asetukset tallennetaan Setting elementillä. Asetus voi sisältää useita kenttiä, eli parametreja (Parameter). Parametrit voidaan määritellä saamaan erilaisia arvoja sen mukaan, mitä optioita laitteeseen on asennettu.
Erilaisin ehdoin sovellettavat arvot varastoidaan Variables elementissä, johon tallentuvat samalla ehdot kyseisten arvojen soveltamiseen. Variables elementissä parametrin arvot

sisältävät ParameterValue elementit ovat samassa järjestyksessä, kuin Setting elementin Parameter elementit.

Oheisessa esimerkissä määritellään asetukset asetuksien avulla laitteelle soveltuva pistorasia sekä testauslaitteiston mittalaitteiden sarjaportit. Siinä pistorasiaan vaikuttaa laitteen koko ja sarjaporttiliitäntään ei optioilla ole vaikutusta.

```
<Setting Name="Pistorasia">
  <Parameter>Virta (A)</Parameter>
  <Parameter>Vaiheet</Parameter>
  <Variables>
    <Condition Option="Koko" Installed="True">100</Condition>
    <ParameterValue>32</ParameterValue>
    <ParameterValue>3</ParameterValue>
  </Variables>
  <Variables>
    <Condition Option="Koko" Installed="True">200</Condition>
    <ParameterValue>63</ParameterValue>
    <ParameterValue>3</ParameterValue>
  </Variables>
  <Variables>
    <Condition Option="Koko" Installed="True">300</Condition>
    <ParameterValue>100</ParameterValue>
    <ParameterValue>3</ParameterValue>
</Variables>
  <Variables>
    <Condition Option="Koko" Installed="True">400</Condition>
    <ParameterValue>100</ParameterValue>
    <ParameterValue>3</ParameterValue>
  </Variables>
</Setting>
<Setting Name="Sarjaportti">
  <Parameter>Finero FST</Parameter>
  <Parameter>VIP Energy</Parameter>
  <Variables>
    <ParameterValue>COM1</ParameterValue>
    <ParameterValue>COM4</ParameterValue>
  </Variables>
</Setting>
```

Testit ja viittaus testausmenetelmään

Testit määritellään Test-elementissä, jonka attribuutteina ovat sen nimi (Name) ja epäonnistuneen vaikutusta prosessin kulkuun kuvaavat tiedot. Lupa yrittää testiä uudelleen ilman prosessin aloittamista alusta määritetään AllowRetry-attribuutilla. AllowSkip-attribuutti sallii aikaisemman hyväksytyn tuloksen hyödyntämisen prosessissa ja AllowContinueOnFail-attribuutti antaa luvan jatkaa testausprosessia, vaikka saatu tulos olisi hylätty. Test elementti voi myös sisältää ehtoja, joilla testi rajataan suoritettavaksi vain joil-

lekin laitteille. Testausmenetelmät on lueteltu TestCase-elementeissä. Ne sisältävät tapauksen nimen ja viittauksen toiseen tiedostoon, jossa testitapauksen tarkemmat tiedot ovat tallennettuina.

```
<Test Name="Suojajohtimen jatkuvuus"
    AllowRetry="True"
    AllowSkip="False"
    AllowContinueOnFail="False">
  <TestCase Name="Testi">
    C:\Koestus\Parameters\Suojajohtimen jatkuvuus_Testi.xml
  </TestCase>
</Test>
```

Testausmenetelmän tarkat tiedot

Myös testausmenetelmät sisältävässä tiedostossa data tallennetaan Config-juurielementtiin. Tiedosto voi sisältää useita testausmenetelmiä ja esimerkiksi automaattinen tiedostonnimeämiskäytäntö tallentaa eri tuoteperheelle konfiguroidut, mutta samalla tavoin nimetyt testi ja testitapausyhdistelmän testitapauksen samaan tiedostoon. Ne erotetaan toisistaan tallentamalla TestCase-objektin attribuutteihin nimen lisäksi se testi, jonka se toteuttaa sekä se tuoteperhe, jolle se kuuluu.

Testausmenetelmän parametrit määritellään muuten samalla tavoin, kuin tuoteryhmäkohtaisten, mutta niillä on useampia attribuutteja. AllowEdit-attribuutti antaa tavalliselle käyttäjälle luvan muokata parametrin arvoa ennen testausprosessin aloittamista ja Hidden-attribuutti estää parametrin näkymisen tavalliselle käyttäjälle. Testausmenetelmien määrittelyssä lisänä on vielä mittausten määrittely. Ne määritellään Measurement-elementillä ja raja-arvot puolestaan MeasurementLimits-elementin LowLimit- ja HighLimit-attribuuteilla.

```
<TestCase Name="testi" TestName="Suojajohtimen jatkuvuus" ProductLine="Proveno 2G">
  <Comments />
  <Executable>C:\Koestus\Tests\GroundBond\GroundBondTest.exe</Executable>
  <Parameter AllowEdit="false" Hidden="false">Koestusjännite (V)</Parameter>
  <Parameter AllowEdit="false" Hidden="false">Koestusvirta (A)</Parameter>
  <Parameter AllowEdit="false" Hidden="false">Koestusaika (s)</Parameter>
  <Measurement AllowEdit="false" Hidden="false">Resistanssi (m ohm)</Measurement>
  <Variables>
    <ParameterValue>6</ParameterValue>
    <ParameterValue>10</ParameterValue>
    <ParameterValue>5</ParameterValue>
    <MeasurementLimits LowLimit="0" HighLimit="100" />
  </Variables>
</TestCase>
```