Teemu Keurulainen

Laboratoriotestien seurantajärjestelmä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö
24.5.2016

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Teemu Keurulainen Laboratoriotestien seurantajärjestelmä 16 sivua 24.5.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	
Ohjaaja(t)	Lehtori Markku Inkinen Laboratorioesimies Henna Pietarinen

Työn tarkoituksena oli suunnitella testien seurantaohjelma Muratan validointilaboratoriolle ja toteuttaa demosovellus. Seurantaohjelma helpottaisi laboratorio-operaattoreiden työntekoa yksinkertaistamalla testien aloitusta ja lopetusta, sekä poistaisi ylimääräisiä työvaiheita. Ohjelma laskisi rasitustesteille lopetusajan annetun keston mukaan ja ilmoittaisi testin olevan valmis.

Ohjelma suunniteltiin mahdollisimman yksinkertaiseksi, jotta se on helppo ottaa käyttöön. Käyttöliittymän suunnittelussa käytettiin menetelmää nimeltä käytettävyystestaus, joka antaa käyttäjille mahdollisuuden vaikuttaa lopputulokseen. Käytettävyystestauksen avulla saatiin ohjelmasta sopiva kaikille.

Seurantaohjelma toteutettiin Java-ohjelmointikielellä ja Netbeans IDE -ympäristössä. Tärkeimpiä komponentteja koodissa oli swingworker -luokka, joka mahdollisti useamman ajastimen käytön ohjelmassa.

Ohjelmasta saatiin toteutettua toimiva demo-sovellus. Käyttöliittymä oli helppo käyttää ja kaikki tarvittavat ominaisuudet toimivat. Ohjelma laski testeille lopetusajan ja ilmoitti sen valmistumisesta.

Avainsanat	seurantajärjestelmä, Java, validointilaboratorio

Author(s) Title Number of Pages Date	Teemu Keurulainen Tracking Software for Laboratory Tests 16 pages May 24th 2016
Degree	Bachelor of Engineering
Degree Programme	Automation technology
Specialisation option	
Instructor(s)	Markku Inkinen, Senior Lecturer Henna Pietarinen, Manager

The purpose of this study was to design tracking software for Murata's validation laboratory and produce demo software of the design. The tracking software would make laboratory operators work easier by simplifying the start and ending of tests. It would also decrease the number of work phases. The software would calculate ending time for tests from start time and duration, and it would then indicate when the test is finished.

The software was designed to be as simple as possible, so that it would be easy to use. A method called usability testing was used in the design of the user interface to take the operators' opinions into account. By using usability testing the software would be suitable for everyone.

Tracking software was developed using Java programming language and Netbeans IDE environment. One of the most important components in the code was swingworker -class which enabled usage of multiple timer string.

As a result of the study, working demo software was developed. It worked with only one test running at a time, but provided a good starting point for more development. User interface was easy to operate and every required feature was functional. The software calculated the ending time for the test and informed when it was over.

Keywords	Tracking software, Java, Validation laboratory

Sisällys

1 Johdanto 1

1.1 Yleistä 1
1.2 Murata Electronics Oy 1

2 Määrittely 1

2.1 Validoinnin tarkoitus 2
2.2 Lähtötilanne 3
2.3 Tavoite 4
2.4 Käyttöympäristö 5

3 Seurantaohjelman Suunnittelu 6

3.1 Käyttöliittymä 6
3.2 Kehitysympäristö 8
3.2.1 Java 8
3.2.2 Netbeans IDE 9
3.3 Atlassian Jira 9
3.4 Ohjelman käytön suunnittelu 10

4 Demo-ohjelman toteutus 11

4.1 Käyttöliittymä 11
4.2 Testin aloitus ja lopetus 12
4.3 SwingWorker 13
4.4 Jira-yhteys 14

5 Lopputulos 14

6 Yhteenveto 15

6.1 Tulevaisuus 15
6.2 Laajennukset 16

Lähteet 17

Liitteet

Lyhenteet

MFI	Murata Finland eli Murata electronics Oy
MEMS	Mikroelektromekaaninen järjestelmä
JIRA	Atlassianin luoma tehtävienhallintaohjelma
JAVA	Sun Microsystemsin luoma olio-ohjelmointikieli.
SWING	Java-kielelle tarkoitettu kirjasto graafisen käyttöliittymän luontiin.
EDT	Event Dispacher Thread. Javan swing-tapahtumia käsittelevä säie.

1 Johdanto

1.1 Yleistä

Insinöörityössä suunnitellaan sovellus ja toteutetaan demo laboratoriotestien seurantaa varten. Työ tehdään Murata Electronics Oy:n validointilaboratoriolle. Seurantaohjelman tarkoituksena on helpottaa ja yksinkertaistaa työntekoa laboratoriossa yhdistämällä ja automatisoimalla työvaiheita, jotka vievät turhaa aikaa.

1.2 Murata Electronics Oy

Murata Electronics Oy eli MFI valmistaa ja kehittää piipohjaisia kiihtyvyys-, kallistus- ja kulmanopeusantureita. Antureissa käytetään Muratan ainutlaatuista 3D MEMS -teknologiaa.[1.]

Murata Electronics Oy perustettiin vuonna 1991 nimellä Vaisala Technologies Inc.,Oy. Yrityksen perustivat Vaisala Oy ja SITRA. 1995 yhtiö siirtyi autojen turvatyyny- ja ohjaushallintajärjestelmiä valmistavan BREED Technologiesin omistukseen. Yritykselle rakennutettiin uusi tuotantolaitos 1998, jossa nykyinen Murata Electronics toimii edelleen. Vuonna 2012 yritys vaihtoi nimensä Murata Electronics Oy:ksi emoyhtiö Muratan mukaan. [1.]

Murata Manufacturing on maailmanlaajuinen markkinajohtaja elektronisten komponenttien tuotannossa ja myynnissä. Muratan innovaatioita löytää muun muassa autoista ja matkapuhelimista.[1.]

2 Määrittely

Työn alussa selvitetään mitkä ovat lähtökohdat ja mihin lopputulokseen halutaan päästä. Alussa on selvitettävä mitä työvaihetta seurantajärjestelmä muuttaa ja miten se vaikuttaa työntekoon.

2.1 Validoinnin tarkoitus

Validoinnilla tarkoitetaan kohteen toimivuuden tarkastamista tiettyjen kriteerien mukaan. Kohde voi olla mm. prosessi, laite, komponentti tai menetelmä. Validointi suoritetaan, jotta voidaan varmistaa ja todistaa kohteen toistettava ja tasainen toiminta. [2.]

Validointi kuuluu osaksi laadunvarmistusta. Laatua ei pysty varmistamaan pelkästään lopputuotteen perusteella, joten se liitetään jo valmistusprosessiin. Validoinnilla varmistetaan prosessivaiheiden ja tuotteiden toimivuus jo hyvissä ajoin ennen tuotannon käynnistämistä. [2.]

Muratan validointilaboratoriossa testataan uusien tuotteiden tai tuotantomenetelmien toimivuus. Yleisimpiä kriteereitä validoinnissa ovat tuotteen omat spesifikaatiot tai teollisuuden standardit kuten JEDEC ja IPC [3.]. Laboratorioon kuuluu erilaisia testilaitteita, joilla simuloidaan komponenttien toimintaa niiden käyttöympäristössä. Komponentteja rasitetaan erilaisissa olosuhteissa, jotta voidaan tutkia pitkän aikavälin vaikutusta. Esimerkiksi mekaanista rasitusta ajan kuluessa testataan vaihtelemalla lämpötilaa -50 ja +150 -asteen välillä.

Insinöörityössä keskitytään laboratorion rasitustesteihin, jotka suoritetaan uuneilla ja kosteuskaapeilla. Uunit ja kosteuskaapit eivät itsessään sisällä elektroniikkaa, joka mahdollistaisi seurannan suoraan laitteesta. Muita komponenttien rasitukseen tarkoitettuja testejä joita validointilaboratorio suorittaa ovat esimerkiksi lämpötilasyklaus ja painetestaus.

2.2 Lähtötilanne

TESTIERÄN SEURANTAKORTTI

PROJEKTI: SCA10x0 NUMERO: 96079

TESTAAJA(T): TEHI PUH: 749

JIRA: QAVAL-947 POC/DV/PV/muu:

TUOTETYYPPI: SCA10x0 KPL: 60

☒ TESTI, PROFIILI (ympyröi) ja ARVIOITU KESTO: 1000h

TC	-50°C … +150°C, 10 min dwell / muu	
HTSL	+125°C / +150°C / muu	muu profiili, mikä testilaite
UHAST	+130°C/85%RH / muu	BAKE04
HTOL	+125°C / +150°C / muu	
HAST	+130°C/85%RH / muu	BIAS: 5,5V
THB	+85°C/85%RH / muu	

muu, mikä

☐ tai … ESIKÄSITTELY, PROFIILI ympyröi esikäsittelyvaihtoehto ylivirtaa suorittettu osuus

MSL3 (24h +125°C, 40h 60°C/60%RH, 3x ERSA 96 PROFIILI)
MSL2 (TC 5 sykliä, 24h +125°C, 168h 85°C/60%RH, 3x ERSA 96)
muu, mikä

ALOITUS PVM JA AIKA	LOPETUS PVM JA AIKA	SYKLIT/TUNNIT	
1.1.2010 18:00	8.1.2010 18:00	168h	YHT
9.1.2010 18:00	13.2.2010 10:00	832h	1000h

MUUTA/
HUOMIOITAVAA

Kuva 1. Laboratoriotestien eräseurantakortti.

Laboratorion rasitustestejä seurataan mm. eräseurantakortilla (kuva1). Korttiin merkitään testin alussa kellonaika, jolloin osat laitetaan testiin, testin kesto tunneissa ja päivämäärä ja aika, jolloin testi on valmis. Eräseurantakortti kiinnitetään sen uunin tai kosteuskaapin oveen, jonne testiosat on laitettu rasituksiin. Testien valmistumisaikaa seurataan tarkistamalla eräseurantakortit kaappien ovista. Testejä on käynnissä parhaimmillaan kymmeniä, joissa kaikissa on omat rasitukset. Eräseurantakortteja on uunien ja kosteuskaappien ovissa monia, joten niiden seuraaminen on hankalaa ja vie aikaa.

Eräseurantakortti täytetään käsin, joten virheiden mahdollisuus on suuri. Työntekijä saattaa laskea testiajan väärin, mikä vaikuttaa testin kulkuun ja tuloksiin. Osat saattavat jäädä rasitukseen liian pitkäksi aikaa, koska työntekijä ei huomaa eräseurantakorttia tai muistaa ajan väärin. Eräseurantakortti voi myös unohtua, hävitä tai pudota testikaapin ovesta.

Testin aloitus- ja lopetusajat merkitään myös laboratoriossa käytössä olevaan tehtävänhallintaohjelma Jiraan (kts. sivu 9), joten niitä voi seurata omalta tietokoneelta ilman

laboratorioon menemistä. Jiralla hallitaan kaikkia laboratorion käynnissä olevia testejä, joten oikeiden testien löytäminen voi olla hankalaa ja vaatii usein monen tehtävän avaamista samaan aikaan. Samassa tehtävässä saattaa olla monta eri osapopulaatiota, jotka ovat eri testeissä. Näiden erottelu vie aikaa ja saattaa johtaa osien sekoittumiseen.

2.3 Tavoite

Työn tavoitteena on tehdä toimiva demo-sovellus ohjelmasta, joka seuraa testejä ja ilmoittaa niiden valmistumisesta. Sovelluksen avulla vähennetään operaattorin ylimääräistä työtä.

Laboratoriotestien seurantaohjelma kerää kaikki käynnissä olevat testit samalle näytölle. Ohjelma laskee automaattisesti lopetusajan annetun aloitusajan ja keston perusteella ja ilmoittaa, kun testi on lähellä valmistumista. Käyttöliittymässä on oma paneeli testin tiedoille ja taulukko, jossa käynnistetyt testit näkyvät omina riveinään. Testien seurantaohjelma tulee aluksi käyttöön vain paisto- ja kosteustesteille, koska ne vaativat eniten tarkkailua.

Tavoitteena on vähentää aikaa, joka kuluu testien läpi käymiseen yksitellen. Valmistuvat testit voi nähdä samasta paikasta yhdellä kertaa, mikä vähentää työntekijän liikkumaa matkaa päivässä, sekä testien ulkoa muistamista. Työntekijä pystyy näin ajoittamaan työtehtävänsä paremmin, mikä sallii keskittyneemmän ja huolellisemman työnteon. Testiosat saadaan aina oikeaan aikaan pois rasituksesta, koska lopetusajat ovat jatkuvasti kaikkien nähtävillä ilman erillistä tehtävien selaamista. Rasitusten jälkeisten testien toteutus helpottuu, koska ohjelmasta pystyy kerralla näkemään tulevalla viikolla valmistuvat rasitukset. Näin koko projekti pysyy aikataulussa ja testivuo saadaan loppuun sovitusti. Myös testiosien järjestely uuneihin helpottuu, kun testit näkyvät listana. Ohjelmasta näkee nopeasti uunien ja kosteuskaappien käyttöasteen, mikä helpottaa uusien testiosien sijoittamista.

Parhaimmassa mahdollisessa tilanteessa seurantaohjelman kautta kiertää kaikki tarvittava tieto rasituksista. Ohjelma lisää automaattisesti kommentit Jiraan, jolloin koko projektin seurannasta tulee helpompaa ja yksinkertaisempaa. Ohjelma aikatauluttaa työntekijöiden päivän valmiiksi rasitusten osalta, jolloin muut työt on helpompi jakaa pitkin

päivää. Selkeämpi aikataulu ja tehtävien järjestely lisää tehokkuutta ja huolellisuutta niitä vaativissa töissä.

2.4 Käyttöympäristö

Laboratorioon kuuluu kaksi eri aluetta, itse laboratorio sekä toimistotila. Laboratorion puolella sijaitsevat kaikki laitteet sekä tarvikkeet. Myös uunit ja kosteuskaapit, joihin seurantaohjelma liittyy, sijaitsevat laboratoriossa. Toimiston puolella jokaisella työtekijällä on oma työpiste, jossa on tietokone. Työpisteellä kirjataan ylös kaikki testit ja rasitukset, joita tehdään, sekä tarkastellaan tuloksia.

Seurantaohjelma tulee saada näkyville molemmissa tiloissa, jotta edestakainen kulku minimoidaan ja virheiden määrä pienenee. Operaattorin on helpompi seurata ja lisätä testejä ohjelmaan omalla työpisteellään. Laboratorion puolella testejä pitää pystyä seuraamaan muun työn ohessa.

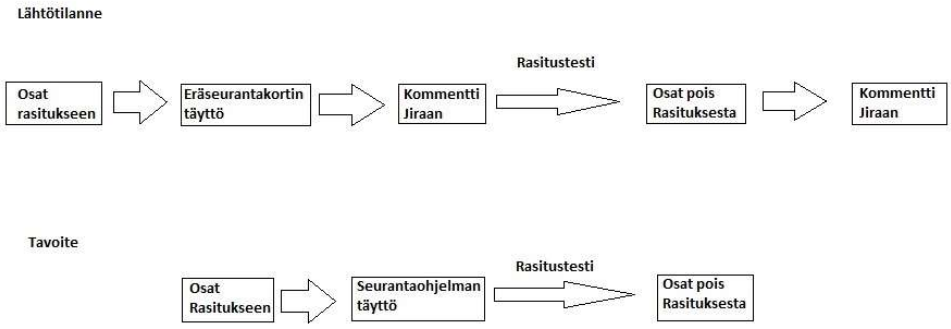
Kuva 2. Validointilaboratorion uunien sijainti.

Laboratoriossa uunit on sijoitettu vierekkäin yhteen paikkaan (kuva 2). Seurantaohjelman käynnissä olevat testit tulisi saada näkyviin tällä alueella, jotta operaattori voi nopeasti ja vaivatta nähdä seuraavaksi valmistuvat testit. Ohjelma tulee näkyviin näytölle uunien läheisyyteen, mutta sen ei tarvitse olla muokattavissa. Kosteuskaapit ovat sa-

malla tavalla vierekkäin, mutta eivät uunien läheisyydessä. Oma näyttö näillekin kaapeille voisi olla mahdollinen, mutta koska laboratorio ei ole iso alue, voidaan yksi yhteinen näyttö sijoittaa keskeiselle paikalle.

3 Seurantaohjelman suunnittelu

Ohjelman suunnittelussa tulee huomioida sen käytettävyys. Ohjelman tulee olla tarpeeksi helppokäyttöinen ja yksinkertainen, jotta sen käyttö ei aiheuta lisätyötä ja sekaannusta. Tavoitteena on saada yhdellä kertaa kaikki tarvittavat tiedot päivitettyä. Samojen tietojen kirjoittaminen moneen eri paikkaan lisää riskiä, että kommentit menevät sekaisin (kuva 3).



Kuva 3. Lähtötilanne ja tavoite testin kululle.

Ohjelman käytettävyyttä voidaan parantaa lisäämällä tarvittavat tiedot alasvetovalikoilla. Kirjoittamisen vähentäminen pienentää virheiden määrää, koska valittavissa on vain käynnissä olevat testit tai laboratoriossa olevat laitteet. Käyttöliittymä ja ohjelma saadaan käyttäjäystävälliseksi teettämällä käytettävyystestaus operaattoreilla.

3.1 Käyttöliittymä

Käyttöliittymä on isossa osassa ohjelmassa, joten sen ulkoasun ja käytettävyyden tulee olla yksinkertainen ja helppokäyttöinen. Käyttöliittymän käytettävyys saadaan parhaaksi mahdolliseksi huomioimalla käyttäjien mielipiteet ja toiveet. Käyttöliittymän lopullinen

muoto muodostuu kaikista käyttäjien esittämistä ideoista. Laboratorion operaattoreilta ja insinööreiltä saaduista ajatuksista ja ehdotuksista saadaan luotua kaikille sopiva käyttöliittymä.

Käyttöliittymän täytyy näyttää yksinkertaiselta ja siitä pitää löytyä kaikki tarvittavat komponentit testin aloittamiseen. Testin aloituksen tulee olla helppoa, jotta ohjelmaa kannattaa käyttää. Insinööreiltä ja operaattoreilta saatujen ehdotusten ja ideoiden mukaan saatiin suunniteltua helposti käytettävä ja yksinkertainen paneeli testin aloittamista varten.

Käytettävyystestaus

Käytettävyystestauksella saadaan tuotua esille käyttäjän kannalta ongelmalliset tilanteet ennen varsinaisen ohjelman valmistumista. Käytettävyystestin tekijä pystyy kertomaan jo testiä tehdessään, mikä on vaikeaa tai epäkäytännöllistä, jolloin ohjelmasta saadaan mahdollisimman nopeasti toimiva ja käyttäjäystävällinen. [4.]

Seurantaohjelmaa vasten käytettävyystestaukseen valittiin kaksi laboratorio-operaattoria ja kaksi validointi-insinööriä. Testaus suoritettiin ensin paperiversiolla ja sen jälkeen tietokoneversiona, joka muistutti jo lopullista käyttöliittymää.

Ensimmäinen malli käyttöliittymästä piirrettiin paperille, jotta siitä saatiin selvä kuva ja jotta sitä oli helpompi muokata. Ensimmäistä versiota esiteltiin operaattoreille ja laboratorioinsinööreille, ja heille annettiin tehtäväksi testin käynnistäminen. Ensimmäinen muutosehdotus oli testin sijainnin valinta. Suunnitelmassa oli lueteltu kaikki mahdolliset kaapit erikseen omina valintapainikkeinaan, mikä näytti sekavalta. Toiseen paperiversioon oli muutettu kaikki ajatukset, jotka käyttäjiltä saatiin. Käytettävyystestaus uusittiin samoilla testihenkilöillä sekä yhdellä uudella operaattorilla. Paperi annettiin operaattorille, jonka tehtävänä oli selostaa, miten aloittaisi testin kyseisen mallisella käyttöliittymällä. Testin tulosten perustella koodattiin tietokoneelle lopullista ohjelmaa vastaava käyttöliittymä ja testattiin käytettävyys vielä kerran. Testiryhmää laajennettiin jälleen, jotta kaikki validointitiimin jäsenet voisivat kommentoida ohjelmaa ennen varsinaista versiota.

Käytettävyystestauksen perusteella käyttöliittymästä saatiin kaikille sopiva ja helppokäyttöinen. Saadun palautteen perusteella oli helppo jatkaa ohjelman suunnittelua ja toteutusta.

3.2 Kehitysympäristö

Kehitysympäristöksi valittiin Netbeans IDE ja ohjelmointikieleksi Java. Muratalla käytetään Java-kieltä monessa eri yhteydessä, joten valinta seurantaohjelman ohjelmointikieleksi oli selkeä. Muratan IT-osasto sekä ohjelmistokehitystiimi pystyvät tarvittaessa laajentamaan seurantaohjelmaa ja auttamaan demo-sovelluksen tekemisessä.

3.2.1 Java

Java on Sun Microsystemsin kehittämä laaja teknologiaperhe ja ohjelmistoalusta, johon kuuluu muun muassa oliopohjainen ohjelmointikieli. Java on julkaistu vapaana ohjelmistona GNU GPL -lisenssillä vuonna 2006. [5.]

Java-ohjelmointikielen on kehittänyt Bill Joy ja James Gosling 1990-luvun alussa. Ensimmäinen Java Development Kit (JDK) julkaistiin 1995. Laitteistoriippumattomuuden ja helposti omaksuttavan kieliopin takia Java saavutti suuren suosion 1990-luvun lopussa. Javaa markkinoitiin aluksi web-käyttöön, mutta sen suosio kasvoi palvelinkäytössä, dynaamisten www-sivujen luonnissa, raskaissa palvelinsovelluksissa ja kännyköissä. [5.]

Muihin kieliin verrattuna Java sisältää runsaasti ominaisuuksia, kuten graafisen käyttöliittymäkirjaston. Javan lähdekoodia ei käännetä suoraan konekielelle, vaan tavukoodiksi, joka suoritetaan virtuaalikoneessa (Java Virtual Machine). Virtuaalikoneessa ajon takia Java-ohjelmat ovat tavanomaisia konekieliohjelmia turvallisempia, mutta myös hieman hitaampia. [5.]

3.2.2 Netbeans IDE

Netbeans IDE on ilmainen, avoimen lähdekoodin integroitu ohjelmointiympäristö, joka tukee kaikkia Javan sovellustyyppejä. Se on alustasta riippumaton, eli se toimii kaikilla tunnetuilla käyttöjärjestelmillä, kuten Microsoft Windows, Mac OS X, Linux ja Solaris. Netbeans yksinkertaistaa Java Swing -sovellusten kehitystä. Kaikki ohjelmointiympäristön toiminnot tulevat moduuleista, jotka ovat heti käytettävissä. [6; 7.]

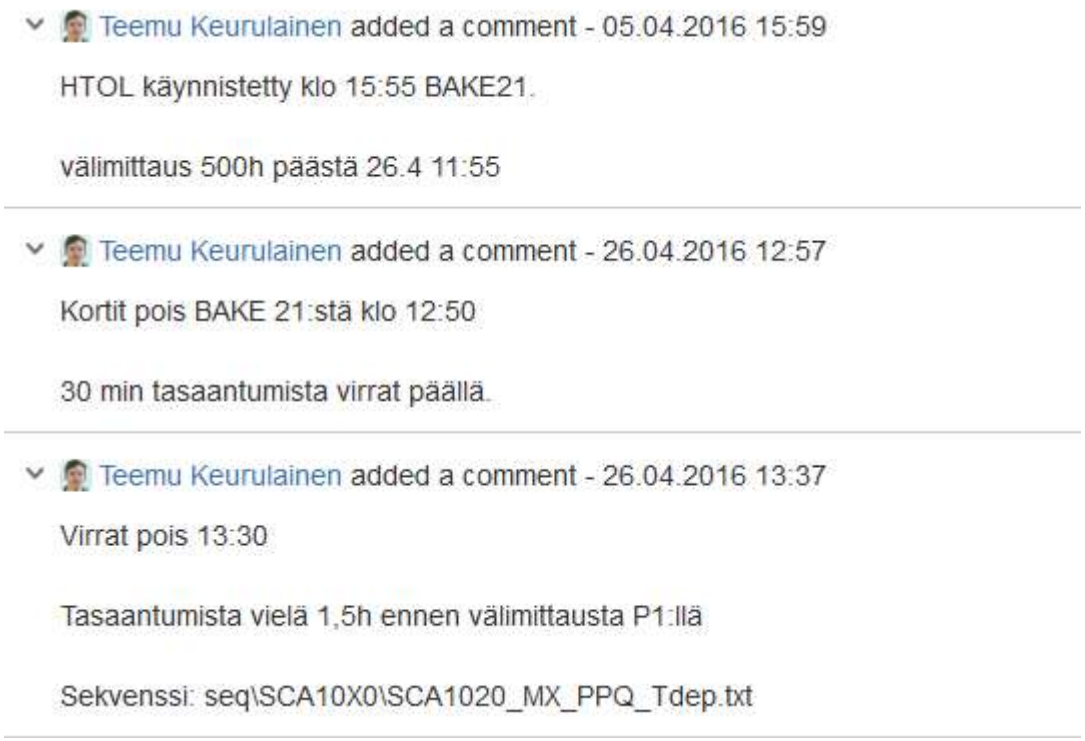
Netbeans syntyi Prahan yliopiston opiskelijaprojektista vuonna 1996 ja myytiin Sun Microsystemsille 1999. Vuonna 2010 Oracle hankki Sun Microsystemsin ja sen mukana Netbeans -ohjelmointiympäristön. [7.]

3.3 Atlassian Jira

Atlassian on australialainen vuonna 2002 perustettu ohjelmistoyritys, jonka tekemä tehtävänhallintaohjelma Jira on. Jiraa käytetään maailmanlaajuisesti projektien hallintaan ja työseurantaan. [8; 9.]

Murata Electronics Oy:n validointilaboratorio käyttää Jiraa projektien seurannassa sekä työtehtävien järjestelyssä. Kaikki projekteihin liittyvät työt luodaan Jiraan omana tehtävänään, jotta jäljitettävyys ja projektin työjärjestys säilyvät. Jokainen uusi työvaihe kirjataan tehtävään, josta tekijä lukee ohjeistuksen ja kommentoi työn edistymistä. Tehtävän luoja saa kaiken tarvittavan tiedon tehdyistä työvaiheista ja edistyksestä (kuva 4).

Seurantaohjelman yhdistäminen Jiraan helpottaa testien seurantaa ja jäljitettävyyttä, jos operaattori ei ole omalla työpisteellään. Jokaiselta testilaitteelta pääsee kirjautumaan Jiraan, jonne myös kirjataan käynnistetyt testit. Jira-yhteys luodaan muodostamalla custom-kentät Jiran taskeihin, joka tunnistaa avainsanat ja kerää tarvittavat tiedot, kuten aloitusajan, lopetusajan ja keston seurantaohjelmasta ja päivittää sen kommenttina Jiraan.



Kuva 4. Paistotestin käynnistys ja seuraus Jirassa.

Jira-yhteyttä varten käytetään Jiran REST API -toimintoa. REST API on rajapinta, jonka kautta ohjelma kommunikoi Jiran kanssa ja päivittää haluttuja tietoja.

3.4 Ohjelman käytön suunnittelu

Käyttö tulee suunnitella mahdollisimman yksinkertaiseksi, jotta operaattorit käyttävät uutta ohjelmaa. Laboratorion testejä seurataan jo Jira-kommenttien avulla, joten ohjelman tulee olla vaivaton käyttää Jiran rinnalla. Jos käyttö on liian monimutkaista, vanhasta tavasta luopuminen on vaikeaa.

Ulkoasun tulee olla selkeä ja hyvin jaoteltu. Testien aloittaminen tehdään alasvetovalikoilla ja numerokentillä, jotta kirjoitusvirheet ja näppäilyvirheet saadaan minimoitua. Ohjelmaan syötetään vain tarvittavat tiedot, jotta käyttäjä voi yhdellä vilkaisulla selvittää tehtävän tilan. Muut oleelliset tiedot voidaan kirjoittaa Jiraan muistiin.

Ohjelmaan merkitään ensimmäisenä tehtävän Jira-tunnus. Koodi yhdistää rasitettavat osat oikeaan testisuunnitelmaan ja jatko-ohjeisiin. Jira-tunnus toimii myös osien tun-

nuksena erottamaan eri testien osat. Ohjelmaan syötetään myös osien sijainti, eli missä uunissa/kosteuskaapissa osat ovat ja mikä testi on kyseessä. Testi valitaan klikkaamalla oikean testin kohdalla olevaa checkboxia, ja testikaappi valitaan tämän jälkeen alasvetovalikosta.

Kun testin yleiset tarvittavat tiedot on kirjoitettu, valitaan testin aloitusaika ja kesto. Kesto valitaan spinner-komponentilla ja aloitusaika kirjoitetaan tekstikenttään. Ohjelma laskee lopetusajan ja sijoittaa sen omaan kenttäänsä testiriville. Ennen testin lopullista aloittamista on mahdollista lisätä vielä kommentti, joka näkyy vain Jirassa. Esimerkiksi osien lukumäärän tai mahdolliset erikoistestit voi kirjoittaa siihen.

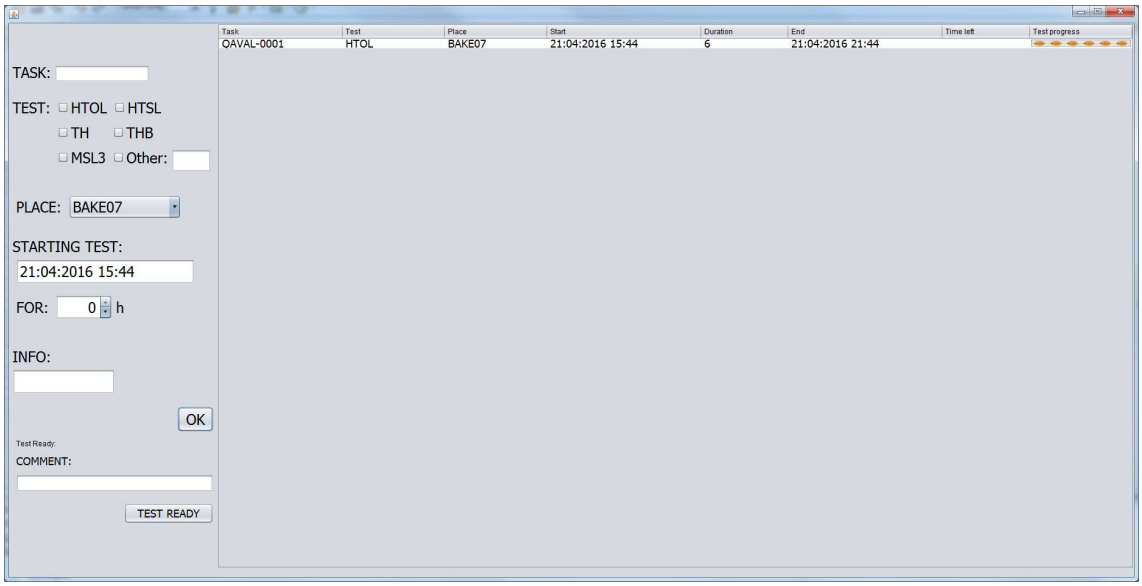
Testi aloitetaan Ok-napilla, kun kaikki tiedot on täytetty. Testi siirtyy omaksi rivikseen taulukkoon, joka listaa kaikki käynnissä olevat testit. Jäljellä oleva aika näkyy tunneittain, jotta se on helppo tarkistaa nopealla vilkaisulla. Ajan ollessa loppumassa, kyseisen testin rivin väri muuttuu, jotta operaattori tietää kyseisen testin vaativan työtä tietyn ajan päästä.

4 Demo-ohjelman toteutus

Ohjelmasta tehdään proof of concept -tyylinen demo, jotta voidaan varmistaa, että ohjelma voi toimia laboratorioympäristössä.

4.1 Käyttöliittymä

Ohjelman teko on helpoin aloittaa käyttöliittymästä. Kun käyttöliittymä on saanut selkeän muodon, voi toimintojen koodaamisen aloittaa. Käytettävyystestauksessa käytetty malli käyttöliittymästä toimi hyvänä pohjana varsinaiseen ohjelmaan.

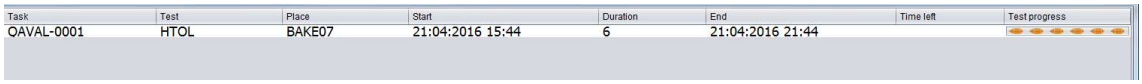


Kuva 5. Seurantaohjelman käyttöliittymä.

Käyttöliittymä sisältää Java Swing -komponentteja, joita ovat muun muassa alasvetovalikot ja tekstikentät (kuva 5). Lista käynnissä olevista testeistä toteutettiin JTable-komponentilla, jotta kaikki rivit näkyvät samalla tavalla ja samankokoisina näytöllä. Käyttöliittymän kaikki komponentit otettiin suoraan valmiista kirjastoista.

4.2 Testin aloitus ja lopetus

Ohjelmointi aloitettiin yhdistämällä käyttöliittymään kirjoitettu data yhdeksi riviksi taulukkoon, jossa testit listataan (kuva 6). Jokaiselle komponentille annettiin muuttujanimi, jotta tunnistetaan, missä järjestyksessä tiedot päivittyvät. Kaikista taulukkoon halutuista tiedoista tehtiin metodi, joka kerää tiedot yhteen ja päivittää ne. Käyttöliittymään nimettiin myös kaikki testit ja testilaitteet, joita validointilaboratoriossa käytetään.



Kuva 6. Testi seurantaohjelmassa.

Testiä aloittaessa painetaan OK-painiketta, jolloin tiedot kaikista valinnoista päivittyvät taulukkoon omalle riville. Task -sarakkeeseen kirjoitetaan tunnus, jolla testiä seurataan Jirassa. Test-kohtaan valitaan mihin testiin osat menevät. Kuvan neljä esimerkissä

testiksi on valittu HTOL (high temperature operating life). Uuni tai kosteuskaappi, johon osat laitetaan, valitaan alasvetovalikolla kohtaa place. Validointilaboratoriossa kaapit on nimetty BAKE tai Kosteus, sekä numero. Testin kesto määräytyy annetun testisuunnitelman mukaan ja se valitaan kohtaan duration. Ohjelma laskee annetusta kellonajasta ja kestosta lopetusajan Time left -sarakkeeseen ja näyttää jäljellä olevan ajan tunteina tai minuutteina.

Testi lopetetaan valitsemalla haluttu rivi taulukosta ja painamalla Test ready -painiketta (Kuva 5). Ohjelma kysyy varmistuksen lopettamisesta, jottei operaattori vahingossa poista väärää testiä. Ennen testin lopetusta operaattorin tulee lisätä kommentti käyttöliittymän kommenttikenttään. Testiä ei saa lopetettua ennen kuin kommentti on kirjoitettu (Kuva 7). Tällä varmistetaan, että testin lopetuksesta jää merkintä Jiraan.

```
private void cancelTest()
{
    DefaultTableModel model=(DefaultTableModel) testChart.getModel();
     String[] choices = {"Yes","No"};
     int response = JOptionPane.showOptionDialog(this,"Are you sure to end test?", "Test ready",JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
         if(response == 0 && !commentField.getText().trim().equals("")){
             model.removeRow(testChart.getSelectedRow());
             commentField.setText("");
         }
         else if(response == 0 && commentField.getText().trim().equals("")) {
             JOptionPane.showMessageDialog(this,"Write a Comment");
         }
         else{

         }
         worker.cancel(true);
```

Kuva 7. Koodi testin lopetuksesta

4.3 SwingWorker

SwingWorker on abstrakti luokka Java-kielessä, jonka avulla voi suorittaa pitkiä tehtäviä ohjelman taustalla. SwingWorker-luokka pystyy suorittamaan monta säiettä samaan aikaan ilman, että muu ohjelma pysähtyy (kuva 8). EDT, joka suorittaa kaikki Swing-komponentit, pystyy suorittamaan vain yhden komennon kerralla. Verrattuna normaaliin ajastimeen, joka ajettaisiin EDT:ssä, Swingworker-luokka suorittaa ajastuksen oman luokan sisällä ja päivittää EDT:tä vain sopivin väliajoin. Tällöin EDT ei pysähdy ajastuksen ajaksi, vaan jatkaa normaalia toimintaansa, päivittäen swing-komponentteja. [10.]

```
186     private void startWorker()
187     {
188       int time;
189       timeLeft= (int) durationSelector.getValue()*60;
190
191       worker = new SwingWorker<Integer,Integer>() {
192           @Override
193           protected Integer doInBackground() throws Exception {
194               for(int i = timeLeft;i>=0;i--){
195                   System.out.println("running"+i);
196                 Thread.sleep(100);
197                 publish(i);
198                testChart.setValueAt(i, 0, 6);
199             }
200                return null;
201         }
202           @Override
203           protected void process(List<Integer> data) {
204
205         }
206
207           @Override
208           protected void done()
209           {
210              testChart.setBackground(Color.red);
211           }
212       };
213        worker.execute();
```

Kuva 8. Swingworker-koodi.

Testien ajastus on ohjelman tärkein osa. Ohjelman tulee seurata monta eri testiä samaan aikaan ja jokaisella on oltava oma ajastin, joka käy eri aikaa. Jos ohjelmassa käytettäisiin vain yhtä säiettä, se suorittaisi vain yhden ajastimen kerrallaan, eikä käyttöliittymä toimisi ajastuksen aikana. Ohjelmaan pitää pystyä lisäämään uusia testejä vanhojen ollessa käynnissä, joten ajastimien tulee pyöriä ohjelman taustalla.

4.4 Jira-yhteys

Jira-yhteyden toteutukseen tarvitaan Muratan IT-osaston apua, koska järjestelmää hallinnoidaan sieltä. Tavoitteena on saada jokaisesta aloitetusta testistä kommentti vastaavaan Jira-tehtävään, jotta seuranta säilyy dokumentoituna. Myös testin lopetuksesta jää kommentti.

5 Lopputulos

Seurantaohjelmasta saatiin valmiiksi toimiva demo, joka voi seurata yhtä testiä kerralla. Testin aikalaskuri näyttää jäljellä olevan ajan ja tarvittavat tiedot testattavien osien jäljit-

tämiseen. Ajan kuluttua loppuun testirivi muuttuu punaiseksi ilmoittaen testin päättymisestä.

Operaattoreilla teetetyn käytettävyyskokeilun perusteella käyttöliittymä on riittävän helppokäyttöinen, jolloin virheiden määrä laskee ohjelmaa käytettäessä. Kaikki koehenkilöistä osasivat käynnistää ja lopettaa testin ilman erillisiä ohjeita ja ymmärsivät testirivin merkinnät. Täytettävien tietojen nimet vastaavat laboratoriossa aikaisemmin käytettyjä termejä, joten testien tilan tunnistaa nopeasti.

Käyttöliittymä on selkeä ja sisältää kohdat kaikelle tarvittavalle tiedolle. Testin tunniste ja aloitusaika syötetään tekstikenttään, muut tiedot valitaan joko alasvetovalikosta tai valitsemalla oikea vaihtoehto klikkaamalla. Kellonaika päivittyy automaattisesti omaan kenttäänsä, mutta sen saa halutessaan muutettua, jos testin on aloittanut aikaisemmin.

6 Yhteenveto

Seurantaohjelmasta saatiin valmiiksi toimiva demo-ohjelma, joka näyttää yhden testin. Käyttöliittymä on selkeä ja testin valinta on yksinkertaista ja nopeaa. Ohjelmaan pystyy päivittämään kaikki samat tiedot, mitä laboratorion testeihin päivitetään tällä hetkellä. Käytettävyystestien ja valmiin demon perusteella laboratorion toimintaa saadaan yksinkertaistettua tämän tyyppisellä ohjelmalla. Operaattorien työ helpottuu, koska muistaminen ja ylimääräinen aikojen tarkistaminen vähenee.

6.1 Tulevaisuus

Koska ohjelma on vain demo, toimiva kokonaisuus on seuraava askel eteenpäin. Ohjelman tulee pystyä seuraamaan kymmeniä testejä samaan aikaan ja päivitettävä Jiraa jokaisen testin kohdalla. Seurantaohjelma tulee saada näkyville keskeisille paikoille tai monelle eri työpisteelle, jotta käyttäjät näkevät sen ilman vaivaa.

6.2 Laajennukset

Perusohjelmaa voi laajentaa seuraamaan kaikkia laboratorion laitteita. Laitteet, jotka sisältävät oman elektroniikkansa, voi liittää osaksi ohjelmaa, jotta tehokkuus myös niiden käytössä lisääntyy. Nämä laitteet eivät välttämättä vaadi käyttöliittymää, vaan vain näytön kertomaan arvioidun keston. Jos kaikki laitteet liitetään osaksi ohjelmaa, saadaan laboratorion toimintaa vielä sujuvammaksi. Operaattorit saavat organisoitua työnsä helpommin, koska tietävät, milloin seuraava laite vaatii työtä.

Lähteet

1 Murata Electronics Oy Intranet. Verkkodokumentti. http://intra.murata.fi/VTI/Pages/default.aspx. Luettu 2.5.2016.

2 Zeus Tech. Validointi. Verkkodokumentti. http://www.zeus.fi/validointi.html. Luettu 17.5.2016.

3 Murata Electronics Oy intranet. Asiakas- ja standardivaatimukset. verkkodokumentti. http://intra.murata.fi/Tyotilat/Tukijapalvelut/Laatu/Laatu-ja-ymparistojarjestelmat/Asiakas-%20ja%20standardivaatimukset/Pages/default.aspx. Luettu 19.5.2016.

4 Avania Oy. 2009. Käytettävyystestaus pähkinänkuoressa. Verkkodokumentti. http://www.avania.fi/kaytettavyystestaus-pahkinankuoressa. Luettu 25.4.2016.

5 Java (programming language). Verkkodokumentti. https://en.wikipedia.org/wiki/Java_(programming_language). Luettu 5.8.2015.

6 Netbeans IDE. Verkkodokumentti. https://netbeans.org/features/index.html. Luettu 5.8.2015.

7 Netbeans IDE. Verkkodokumentti. https://en.wikipedia.org/wiki/NetBeans. Luettu 5.8.2015.

8 Atlassian Jira. Verkkodokumentti. https://www.atlassian.com/software/jira. Luettu 6.8.2015.

9 Jira (Ohjelmisto). Verkkodokumentti. https://fi.wikipedia.org/wiki/Jira_(ohjelmisto). Luettu 6.8.2015.

10 Albert Attard. 2013. Verkkoartikkeli. http://www.javacreed.com/swing-worker-example/. Luettu 12.12.2015