Tampereen ammattikorkeakoulu

Sähkö- ja instrumentointi-suunnittelutyökalun kehitys

Samu Lahti

OPINNÄYTETYÖ
Maaliskuu 2020

Sähkö- ja automaatiotekniikan tutkinto-ohjelma
Automaatiotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Sähkö- ja automaatiotekniikan tutkinto-ohjelma
Automaatiotekniikka

LAHTI, SAMU:
Sähkö- ja instrumentointisuunnittelutyökalun kehitys

Opinnäytetyö 61 sivua, joista liitteitä 3 sivua
Maaliskuu 2020

Opinnäytetyön tavoitteena oli luoda AFRY Finland Oy:n käyttöön suunnittelutyökalu, jolla pystytään helpottamaan prosessisähkö- ja instrumentointisuunnittelua erityisesti pienemmissä projekteissa. Konsultointi- ja suunnittelutoimistoilla on käytössään nykypäivänä lukuisia erilaisia ohjelmistoja sekä tapoja suunnittelutyön helpottamiseksi. Kyseiset suunnitteluohjelmistot ovat yleisesti suunnittelijoiden itse laatimia, taulukkolaskentaohjelmistoihin perustuvia vaikeaselkoisia luetteloita, jotka on tallennettu monen eri tekijän toimesta useaan eri paikkaan.

Ennen opinnäytetyön aloittamista tutkittiin AFRY Finland Oy:n käytössä olevaa ProElina-ohjelmistoa, jota yritys on käyttänyt suunnittelussaan jo monia vuosia. Tämän työkalun pohjalta pyrittiin luomaan yritykselle sekä sen työntekijöille mahdollisimman tutuilla toiminnoilla varustettu uudistettu työkalu.

Työkalu toteutettiin yrityksen työntekijöiden aiemmin laatiman, mutta suunnittelultaan keskeneräiseksi jääneen sovelluksen pohjalta. Työkalun ohjelmoiminen sisälsi kokonaan uuden käyttöliittymän suunnittelun, luetteloiden ja piirikaavioiden generoimisen sekä komponenttien automaattisen valinnan. Erillisenä ominaisuutena pyrittiin ohjelmoimaan asiakasyritysten lähtötietoluetteloista valmis tuontiominaisuus. Varsinaisessa työkalun ohjelmoinnissa käytettiin taulukkolaskentaohjelmisto Excelin VBA-ohjelmointikieltä.

Työkalun toteuttamisen ohella laadittiin erillinen käyttöohje, jonka avulla uudet työntekijät pystyisivät mahdollisimman helposti aloittamaan suunnittelun työkalua käyttäen. Opinnäytetyön lopputuloksena työkalu jaettiin yrityksen suunnittelijoiden yleiseen käyttöön testaamista varten. Sitä pyritään myös jatkuvasti kehittämään sekä levittämään laajemmin yrityksen suunnittelijoiden käyttöön. Opinnäytetyöprosessin aikana työkaluun saatiin sisällytettyä monia eri toimintoja, joiden pohjalta sen kehittäminen entistäkin monipuolisemmaksi on vaivatonta. Koska opinnäytetyön tekemiseen ja suunnittelutyökalun kehittämiseen liittyi salassa pidettäviä tietoja, opinnäytetyöstä on poistettu kaikki sellaiset seikat, jotka voisivat vaarantaa edellä mainitun luottamuksellisuuden.

Asiasanat: automaatio, suunnittelu, prosessisähkö, instrumentointi

ABSTRACT

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

LAHTI, SAMU:
Electrical and Instrumentation Design Tool Development

Bachelor's thesis 61 pages, appendices 3 pages
March 2020

The subject of this thesis was to create a designing tool for AFRY Finland Oy, which would help its employees in process electrification and instrumentation designing especially in smaller projects. Nowadays, consulting and engineering offices have different kinds of applications to help engineers in their work, especially in process electrification and instrumentation engineering. These applications are usually made by designers' themselves and they are based on different kind of spreadsheets which are usually located on designers own hard drives.

Before starting the thesis, company's current design tool, ProElina, was examined. The ProElina design tool has been in use for many years and it was used in this thesis to create a familiar user interface for employees.

The tool developed in this study was based and created on a pre-existing tool which was created by the employees of the company. The development goals included an entirely new user interface, user report and circuit diagram generating option and automatic selection of components. Furthermore, an attempt was made to add an import feature from customer consumer reports. The actual programming of the tool was done using VBA programming language in Excel.

In addition to the tool, a separate manual was created for new employees to help begin using the tool. As a result of this thesis, the tool was distributed to the employees of the company for testing purposes. The tool is also continuously developed and distributed to the designers in the company. During this study, many different functions were implemented to the tool, which makes it easy to develop it into an even more capable tool in the future. All confidential material was omitted from the public version of this thesis.

Key words: automation, engineering, process electrification, instrumentation

SISÄLLYS

1 JOHDANTO 5
2 SUUNNITTELU 6
2.1 Instrumentointisuunnittelu 7
2.2 Prosessisähkösuunnittelu 8
3 NYKYISET SUUNNITTELUTYÖKALUT 12
3.1 ProElina-suunnittelutyökalu 12
3.2 ALMA®-suunnittelutyökalu 16
4 OPINNÄYTETYÖN VAATIMUKSET JA TAVOITTEET 18
5 SUUNNITTELUTYÖKALU ARCO 19
5.1 Visual Basic for Applications -sovellus 19
5.2 Käyttöliittymä ja rakenne 22
5.3 Dokumenttien hallinta 31
5.4 Prosessisähköistys 37
5.5 Instrumentointi 46
5.6 Piirikaavioiden generointi 53
6 SUUNNITTELUTYÖ AIEMMIN JA TULEVAISUUDESSA 54
6.1 Suunnittelutyö aiemmin 54
6.2 Suunnittelutyö tulevaisuudessa 54
7 JOHTOPÄÄTÖKSET 56
LÄHTEET 58
LIITTEET 59
Liite 1. Tavoitteet 59

1 JOHDANTO

Useissa suunnittelu- ja konsultointiyrityksissä on vielä tänäkin päivänä teknologian nopeasta kehityksestä huolimatta käytössään erittäin laaja valikoima vanhentuneita ohjelmistoja tai muita työntekijöiden hajanaisia ratkaisuja suunnittelun helpottamiseksi. Nykypäivänä nämä ohjelmistot eivät kuitenkaan tarjoa enää tarvittavia ominaisuuksia tai ne ovat uusille työntekijöille vaikeasti omaksuttavia. Usein ne ovat myös heikosti skaalautuvia, jolloin varsinkin pienemmille projekteille ne ovat erityisen hankalia ottaa käyttöön. Markkinoilla on myös monia erilaisia kaupallisia ohjelmistoja, jotka eivät kuitenkaan houkuttele yrityksiä hankkimaan niihin lisenssejä. Kaupalliset ohjelmistot ovat usein erittäin huonosti sulautuvia aina kulloisenkin yrityksen käyttöön ja niiden lisenssit ovat kalliita.

Tämän opinnäytetyön aihe tuli AFRY Finland Oy:n tarpeesta kehittää työkalu, joka olisi työntekijöille helppo omaksua sekä käytettävyydeltään pienemmille projekteille sopiva. Työn tavoitteena on toteuttaa työkalu, joka on pääasiassa prosessisähkö- ja instrumentointisuunnittelijoille luotu, suunnittelua helpottava ohjelmisto. Yleisesti prosessisähkö- ja instrumentointisuunnittelijoiden työ sisältää suunnittelu- ja konsultointiyrityksissä urakoitsijaa varten tarvittavien dokumenttien luomisen asiakkaiden lähtötietoluetteloiden perusteella. Käytännössä tämä tarkoittaa erilaisten luetteloiden sekä piiri- ja periaatekaavioiden luomisen. Tällaisia luetteloita ovat esimerkiksi kaapeli-, turvakytkin-, ristikytkentä- sekä keskuslähtöluettelot.

Aluksi luodaan katsaus yrityksen käytössä olevaan ProElina-suunnittelutyökaluun, joka on ollut yrityksen käytössä jo kymmeniä vuosia. Lisäksi tutustutaan lyhyesti Suomessa yleisesti käytössä olevaan ALMA-järjestelmään. Näiden pohjalta pyritään luomaan työntekijöille tuttu suunnitteluympäristö myös uudelle työkalulle. Itse työkalu toteutetaan käyttäen taulukkolaskentaohjelmisto Excelin VBA-ohjelmointia. Työkalun ohjelmoinnin ohella luodaan myös uusille käyttäjille selkeä käyttöohje, jonka avulla käyttäjät pääsisivät mahdollisimman nopeasti käyttämään työkalua tehokkaasti.

2 SUUNNITTELU

Suunnittelu- ja konsultointiyrityksissä suunnittelijoiden roolit pitävät sisällään lukuisia erilaisia tehtäviä. Tässä opinnäytetyössä keskitytään erityisesti sähkö- ja automaatiosuunnittelijan tehtäviin. Suuremmissa yrityksissä on yleensä todella laaja kattaus eri alan ammattilaisia ja näin ollen myös eri suunnittelijoita, kuten prosessi-, mekaniikka- ja putkistosuunnittelijoita. Näiden suunnittelijoiden tehtäviä ei kuitenkaan käydä läpi tässä työssä. Sähkö- ja automaatiosuunnittelijalla on yleisesti projektien kohteina monenlaisia kokonaisuuksia. Projektina voi olla esimerkiksi yksittäisen sähkö- ja automaatiokeskuksen suunnittelu laitteelle, muutostyö jo olemassa olevaan prosessiin tai laajempi, kokonaisen voimalaitoksen sähköistyksen sekä automaation suunnittelu. Tässä työssä paneudutaan erityisesti voimalaitoksien tai vastaavien kokonaisuuksien prosessisähkö- ja instrumentointisuunnitteluun.

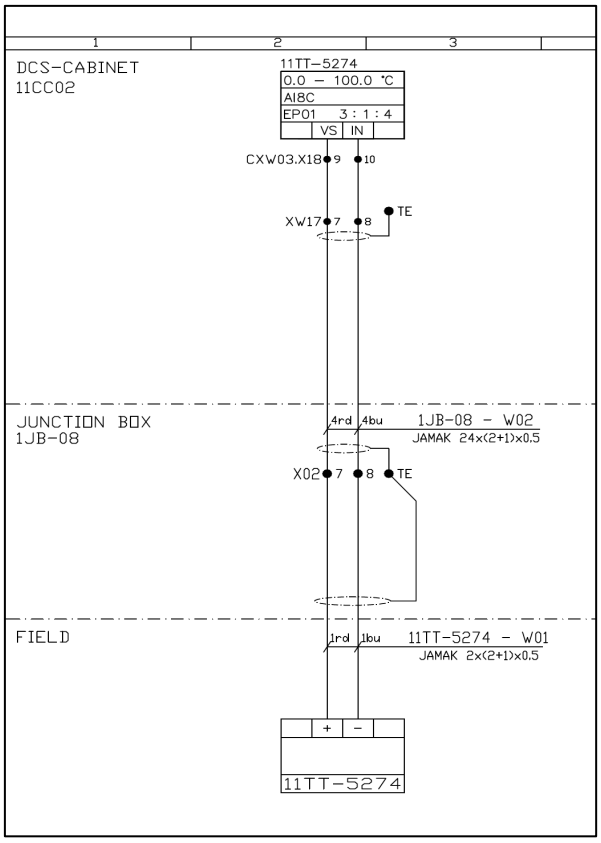
Sähkö- ja automaatiosuunnittelijan tehtävät sisältävät varsinaisen suunnittelun lisäksi myös lukuisia muita käytännön asioita. Tällaisia ovat esimerkiksi asiakkaan kanssa palaverien pitäminen, joissa sovitaan erilaisista muutoksista ja muista alkavan tai keskeneräisen projektin suuntaviivoista. Suunnittelijan näkökulmasta tärkeimmät materiaalit ovat projektin alussa saatavat lähtötiedot. Näissä lähtötiedoissa pyritään määrittelemään suunnittelijalle kaikki tarpeellinen tieto suunnittelun aloittamista varten. Sähköistyksen sekä instrumentoinnin lähtötiedot ovat yleensä taulukkolaskentaohjelmilla laadittuja erilaisia luetteloita, joissa mainitaan jokaisen sähköpiirin eli laitteen sähköiset ominaisuudet. Tällaisia ovat prosessisähköistyksessä esimerkiksi pumppuun tai puhaltimeen kytketyn moottorin teho, jännite ja virta. Instrumentoinnissa lähtötietoina laitteelle, kuten esimerkiksi lämpötilamittaukselle, annetaan yleisesti laitteen tyyppi ja sen vaatimat I/O:t eli liitynnät automaatiojärjestelmään.

Suuremmissa projekteissa suunnittelijoiden tehtäviä on yleensä jaettu eri henkilöille. Tällöin prosessisähkösuunnittelija on siis yleensä eri henkilö kuin instrumentointisuunnittelija. Pienemmissä projekteissa nämä voivat kuitenkin tilanteesta riippuen olla myös sama henkilö.

2.1 Instrumentointisuunnittelu

Instrumentointisuunnittelussa suunnittelijan tehtävänä on luoda asiakkaalta saaduista lähtötiedoista loppuasiakkaan vaatimia dokumentteja. Näihin dokumentteihin voidaan yleisesti ajatella kuuluvan vähintään kaapeliluettelot, piirikaaviot, kilpiluettelot sekä kenttäkoteloiden layout- ja piirikaaviokuvat. Näiden dokumenttien avulla pystytään laskelmoimaan muun muassa tarvittavien kaapelien määrät. Näillä laskemilla puolestaan asiakas tai urakoitsija laskee tarvittavat resurssit sekä tekee tarjouksen työstä. Lopullisilla dokumenteilla urakoitsija pystyy näin ollen toteuttamaan varsinaisen työn.

Instrumentoinnissa tärkeitä suunnittelun kohteita ovat kaapelien tyypit sekä kotelot ja laitteet, joihin kaapelit kytketään. Tarkoituksena on saada kerättyä automaatiojärjestelmään prosessista kaikki tarpeellinen tieto, jotta prosessia voidaan mitata ja ohjata mahdollisimman turvallisesti sekä kustannustehokkaasti. Kuvassa 1 on esitetty tyypillinen piirikaavio lämpötilamittauksen piiristä.



KUVA 1. Lämpötilamittauksen piirikaavio

Kuten kuvan 1 lämpötilamittauksen piirikaaviosta huomataan, on piirikaavion tarkoituksena esittää kaikki liittimet ja kaapelit tunnuksineen automaatiojärjestelmästä kenttälaitteelle eli varsinaiselle instrumentille saakka. Samanlaisia piirikaavioita voi isommissa voimalaitoksissa olla jopa satoja, jolloin piirikaavioiden generoimisista pohjakuvien perusteella on suunnittelijalle suuri hyöty. Suunnittelijan ei siis ole järkevää lähteä piirtämään jokaiselle piirille erikseen kuvaa, vaan hän voi hyödyntää erilaisia työkaluja. Tällaisista työkaluista kerrotaan tarkemmin luvussa 3.

2.2 Prosessisähkösuunnittelu

Prosessisähkösuunnittelussa on hyvin samanlaiset vaatimukset kuin instrumentointisuunnittelussakin. Usein voidaan kuitenkin olettaa, että prosessisähköistyksessä suunnittelun laajuus on instrumentointia laajempi ja tämä johtuu pääasiassa siitä, että prosessisähköistyksessä mukana on sekä sähkökeskuksien että yleisesti ottaen voimalaitoksen sähköverkon suunnittelu. Tämä sisältää instrumentointia laajemman kaapelien poikkipinta-alojen määrittelyn ja muita tarkasti laskettavia mitoituksia sekä esimerkiksi sähkökeskusten välisten lukitusten määrittelyn.

Itse sähkökeskuksiin tulee valita tarkasti eri lähdöille, joista sähkölaitteita syötetään, tarvittavat komponentit sekä liitynnät voimalaitoksen automaatiojärjestelmään. Suunnittelussa tulee myös ottaa huomioon prosessin mahdolliset laajennukset erilaisten varalähtöjen avulla. Prosessisähköistyksessä pidetään siis tärkeänä komponenttien, kaapelien ja muiden laitteiden mitoitusta. Mitoitusta tukemaan luodaan yleensä erilaisia taulukoita, joista suunnittelija voi tarkastaa esimerkiksi tietylle moottorin koolle tarvittavat komponentit, laitteet ja kaapelit. Erilaisten suunnittelua helpottavien työkalujen suurena hyötynä itse suunnittelijalle on näin ollen automaattinen komponenttien sekä kaapelien valinta tiettyjen lähtötietojen avulla. Tällaiset taulukot on kuitenkin erityisen tärkeää päivittää jokaiseen projektiin uudelleen, koska ne ovat eri tekijöistä, kuten esimerkiksi oikosulkuvirroista riippuvaisia.

Kuvassa 2 on esitetty tyypillinen kojevalintataulukko sähkökeskuksille. Taulukosta huomataan, että jokaiselle yleiselle moottorin koolle on määritetty vaadittavat komponentit. Kaapelien koot ovat kuitenkin tästä taulukosta jätetty pois, mutta ne on mahdollista esittää vastaavanlaisessa taulukossa.

Moottorikäyttöjen kojevalintataulukko, 400 V, 80 kA

Huom! Mitoitus riittävä seuraavalle tehoportaalle 200 kW asti.

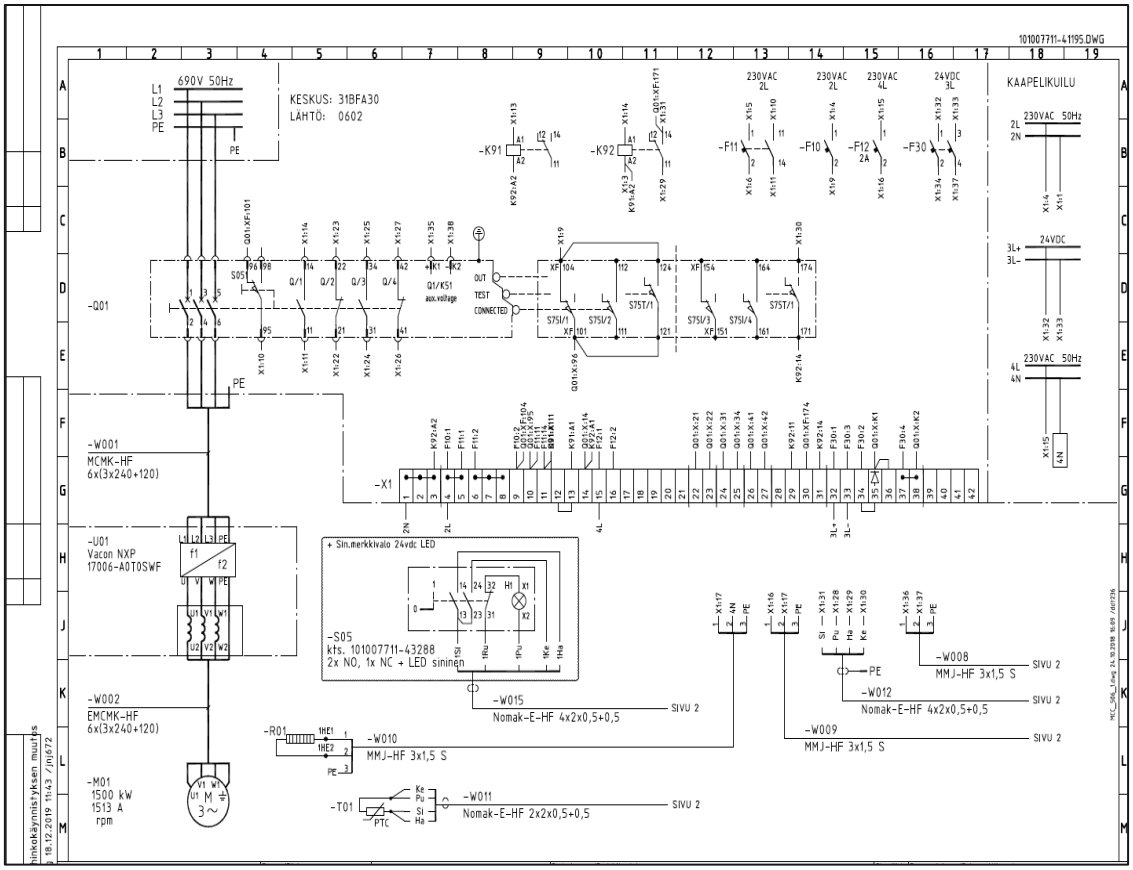
Moottorit ABB					Keskuksen kojeet							
P	In / [A] 400V 50Hz / Moottorin kierr. [r/min]				Kontak- tori-	Lämpörele	Asettelualue / Moottorin kierr. [r/min]				Kytkin-	Sulake OFA_
[kW]	750	1000	1500	3000	tyyppi 2)	Tyyppi	750	1000	1500	3000	varoke 2)	[A]
0,09	0,53	-	-	-	A9	TA25TU	0,4-0,63	-	-	-	OS 32D12	2aM
0,12	0,63	0,59	-	-			0,4-0,63	0,4-0,63	-	-		2aM
0,18	0,9	0,75	0,72	-			0,63-1,0	0,63-1,0	0,63-1,0	-		2aM
0,25	1,18	0,92	0,83	0,7			1,0-1,4	0,63-1,0	0,63-1,0	0,63-1,0		2aM
0,37	1,6	1,25	1,12	0,93			1,3-1,8	1,0-1,4	1,0-1,4	0,63-1,0		2aM
0,55	2,4	1,78	1,45	1,33			1,7-2,4	1,3-1,8	1,3-1,8	1,0-1,4		2aM
0,75	2,7	2,4	1,9	1,7			2,2-3,1	1,7-2,4	1,7-2,4	1,3-1,8		4aM
1,1	3,35	3,3	2,55	2,4			2,8-4,0	2,8-4,0	2,2-3,1	2,2-3,1		4aM
1,5	4,5	4,1	3,4	3,3			3,5-5,0	2,8-4,0	2,8-4,0	2,8-4,0		6aM
2,2	5,9	5,4	4,8	4,5			4,5-6,5	4,5-6,5	3,5-5,0	3,5-5,0		10aM
3,0	7,8	6,9	6,5	6			6,0-8,5	6,0-8,5	6,0-8,5	4,5-6,5		10aM
4,0	10	8,7	8,6	7,4	A12		7,5-11	7,5-11	7,5-11	6,0-8,5		16aM
5,5	13,4	11,9	11,1	10,5	A16		13-19	10-14	10-14	7,5-11		16aM
7,5	18,1	15,4	14,8	13,9	A26		13-19	13-19	13-19	13-19		20aM
11	25	23	22	20	A30		18-25	18-25	18-25	18-25		32aM
15	29	31	29	27	A40	TA75DU	22-32	22-32	22-32	22-32	OS 63 D12	40aM
18,5	36	36	37	33	A50		29-42	29-42	29-42	29-42		50aM
22	45	43	42	40	A63		36-52	36-52	36-52	29-42		63aM
30	60	59	56	53	A75	TA80-/ -110DU	45-63	45-63	45-63	45-63	OS 125 D12	80aM
37	74	69	68	64	A95		65-90	65-90	65-90	60-80		100aM
45	90	82	83	79	A110		65-90	65-90	65-90	65-90		125aM
55	104	101	98	95	A145	E200DU	60-200				OESA 250D3PL	160aM
75	140	140	135	131	A185							200aM
90	167	163	158	152	A210	E320DU	100-320					200aM
110	202	199	193	194	A260						OESA 400D3PL	250aM
132	250	238	232	228	A300							315aM
160	305	280	282	269	AF400	E500DU	150-500				OESA 630D3PL	355aM
200	395	355	349	334	AF460							500aM
250	470	450	430	410								630aM
315	605	565	545	510	AF580	E800DU	250-800				OESA 800D3PL	800aM
355	680	635	610	580	AF750							800aM

KUVA 2. ABB kojevalintataulukko (ABB 2006, 4)

Sähkökeskuksien komponentit ovat usein verrattain helposti valittavissa komponenttien valmistajien, kuten ABB, Siemens ja Schneider Electric, valikoimista, kun taas kaapelien poikkipinta-ala joudutaan laskemaan erikseen eri olosuhteista riippuen. Taulukoissa on usein annettu jokaiselle eri moottorin teholle vaadittava kaapelikoko, mutta suunnittelijan tulee tästä huolimatta osata ottaa huomioon esimerkiksi kaapelin pituuden aiheuttama jännitehäviö sekä mahdolliset normaalista

poikkeavat kaapelin asennustavat. Teollisuudessa kaapelit valitaan usein myös kokoa normaalia suuremmaksi, jotta kaapeli olisi soveltuva myös seuraavalle mahdolliselle moottorin koolle moottorin vaihtuessa suurempaan kokoon.

Prosessisähköistyksessä, kuten instrumentoinnissakin, käytetään yleisesti piirikaavioiden luomiseksi pohjakuvia. Pohjakuvat ovat tarkemmin määriteltynä AutoCAD tai vastaavalla ohjelmistolla piirrettyjä kuvapohjia, joissa on varsinaisen piirin lisäksi erilaisia tekstejä. Tekstissä saattaa olla esimerkiksi $- tai #-merkki ennen tiettyä hakusanaa. Tämän hakusanan avulla puolestaan ohjelmisto etsii esimerkiksi Excel-tiedostosta arvon, jolla kyseinen hakusana korvataan. Tällöin jokaista piirikaavioita ei tarvitse tehdä erikseen, vaan voidaan hyödyntää erilaisia työkaluja. Kuvassa 3 on esitetty taajuusmuuttajalla varustetun moottoripiirin piirikaavio.



KUVA 3. Taajuusmuuttajan piirikaavio

Piirikaaviot prosessisähköistyksessä saattavat olla instrumentointia monimutkaisempia ja ne jakautuvatkin yleensä kahdelle tai kolmelle sivulle, joista esimerkiksi ensimmäisellä on päävirtapiiri, toisella ohjauspiiri ja kolmannella signaalien kytentä. Lisäksi prosessisähköistyksessä on paljon erilaisia viittauksia, kuten esimerkiksi riviliitinten viittauksia. Näitä viittauksia pystyvät nykypäivänä jo tietokoneavusteiset suunnitteluohjelmistotkin tekemään automaattisesti erityisen hyvin, eikä niitä näin ollen yleisesti generoida erillisillä työkaluilla.

3 NYKYISET SUUNNITTELUTYÖKALUT

Suunnittelutyökaluilla voidaan yleisesti tarkoittaa erilaisia kaupallisia ohjelmistoja kuten AutoCAD-ohjelmistoa. Tässä työssä suunnittelutyökaluilla tarkoitetaan erilaisia ohjelmistoja, jotka on laadittu erityisesti projektikohtaista hallintaa varten. Tällaisella ohjelmistolla on tarkoitus hallinnoida yksittäisen projektin dataa ja mahdollistaa suunnittelijoille mahdollisimman helppo projektin ja sen dokumentoinnin hallinnointi. Kaupallisia työkaluja on olemassa monenlaisia ja tässä työssä tullaan käsittelemään niistä kahta erilaista.

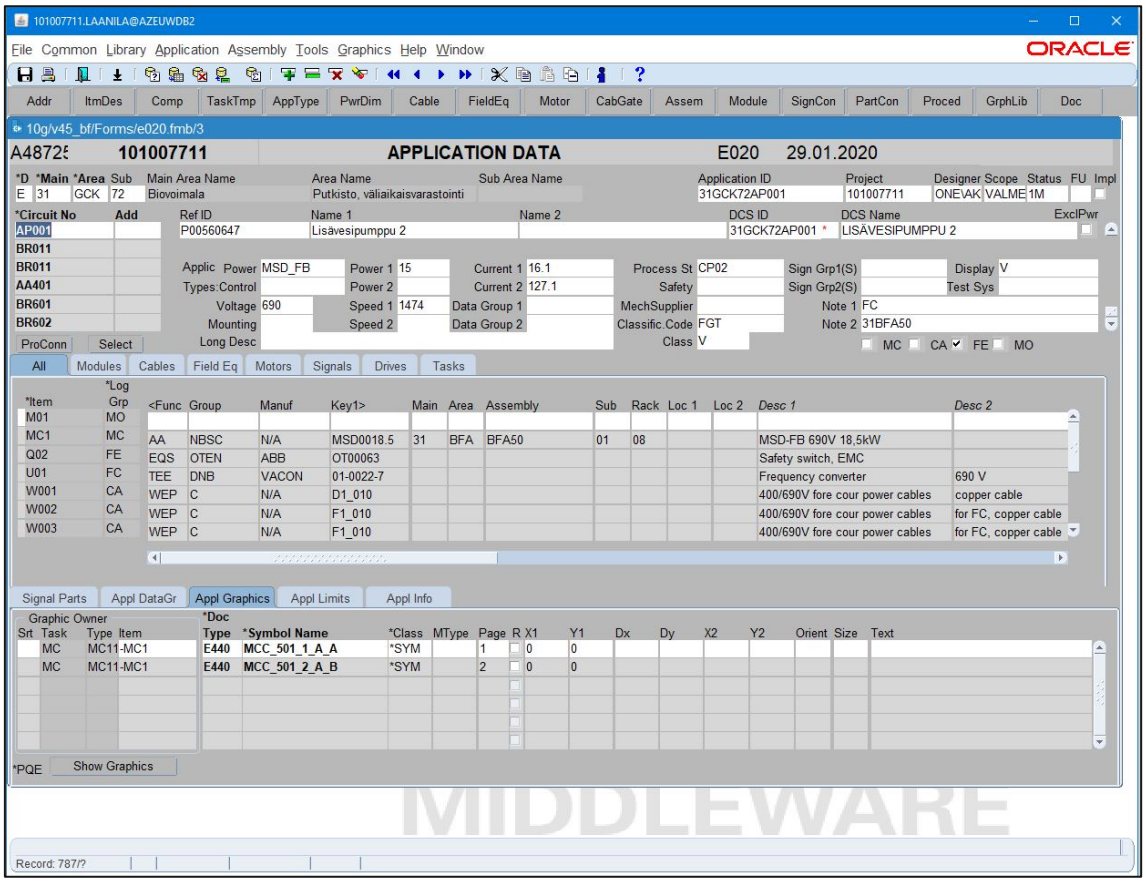
3.1 ProElina-suunnittelutyökalu

ProElina-työkalu on kehitetty Pöyry-nimisessä yrityksessä (nykyinen AFRY Finland Oy) ja se ei ole kaupallistettu ohjelmisto. ProElina mahdollistaa lukuisten eri dokumenttien tuottamisen niin prosessisuunnittelussa kuin myös sähkö- ja instrumentointisuunnittelussakin. Se on ollut yrityksen työntekijöiden käytössä kymmeniä vuosia ja sille on myös yritetty etsiä korvaajaa, mutta tuloksetta.

ProElina-suunnittelutyökalulla saadaan poistettua ongelma, jossa monet eri suunnittelijat laativat dokumentteja omilla tietokoneillaan muiden ulottumattomissa. Tämä lisää huomattavasti suunnittelun aikaisia virheitä sekä turhaa hämmennystä. Tällöin suunnittelijat saattavat tehdä myös saman työn moneen kertaan ja näin ollen suunnittelu vaatii huomattavasti enemmän suunnittelijoiden välistä yhteistyötä sekä koordinointia, kun taas ProElina -työkalulla näitä ongelmia ei samassa laajuudessa kohdata.

Kuvassa 4 on esitetty ProElinan yleisnäkymä. Tässä näkymässä tapahtuu prosessisähköistyksen kaikki perussuunnittelu. Vasemmassa ylänurkassa nähdään haettuna kaikki olemassa olevat piirit. Näitä pystytään työkalulla myös erikseen hakemaan painamalla ensin F7-näppäintä, joka aloittaa haun ja sen jälkeen syöttämällä tiettyyn kenttään haluttu arvo. Tämän jälkeen itse haku tehdään painamalla F8-näppäintä. Muut kentät ovat piirille asetettuja arvoja, kuten moottorin teho, pyörimisnopeus ja määritellyt kaapelit. Työkalun keskiosassa nähdään eri

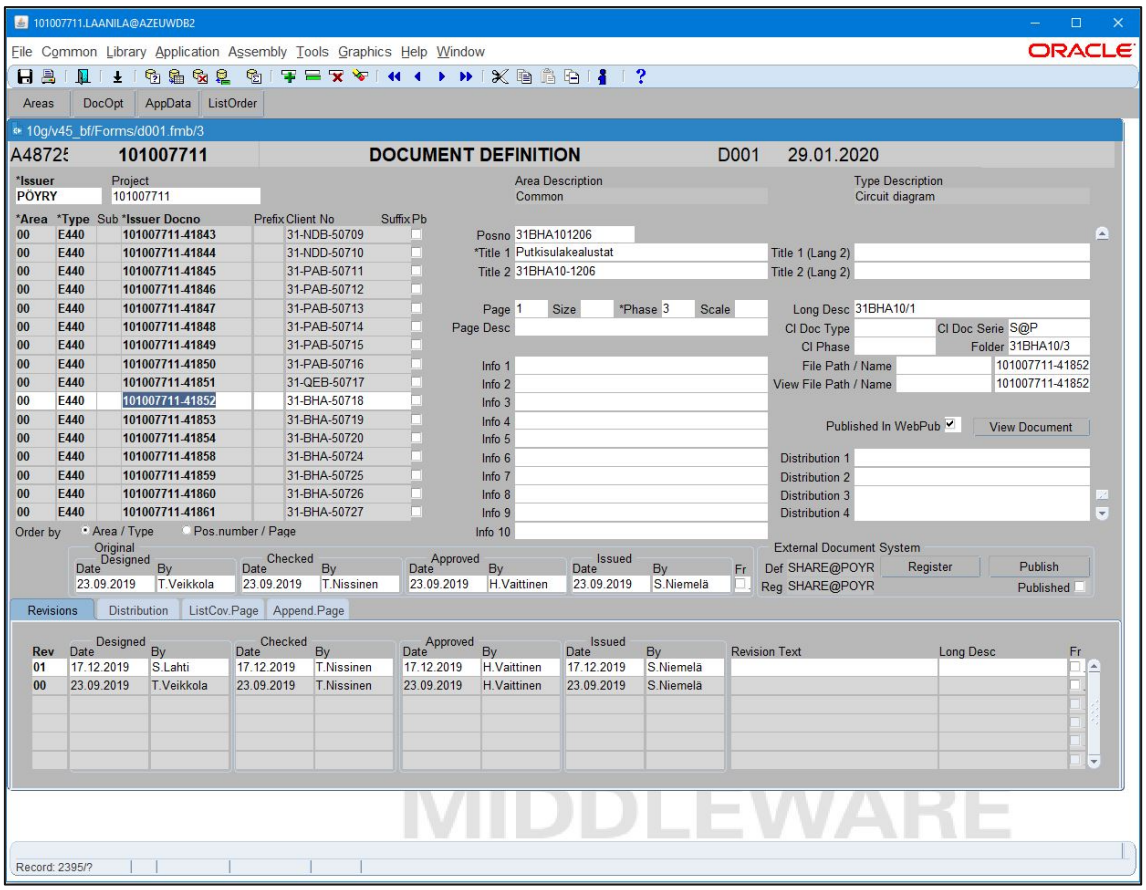
välilehdet muun muassa kaapeleille, moottoreille sekä signaaleille. Näillä välilehdillä suunnittelija pystyy hallitsemaan tarkemmin ja helpommin aina kutakin osa-aluetta. Kuvan alaosassa on esitetty osio, jossa pystytään määrittämään tarkempia piirikaavioiden generoimiseen liittyviä tietoja. Tällaisia ovat esimerkiksi grafiikat tai pohjakuvat.



KUVA 4. ProElina-yleisnäkymä

ProElina sisältää lisäksi monenlaisia välisivuja eri tarkoituksiin. Sillä pystytään esimerkiksi valitsemaan sivu, jossa suunnittelija pääsee tarkemmin käsittelemään keskuksien lähtöjä. Lisäksi työkalussa on oma osio komponenttien määrittelylle. Sillä pystytään esimerkiksi määrittelemään yksittäinen kaapeli tarkasti niin, että sille asetetaan kaapelin halkaisija, johtimien värit tai tunnukset ja valmistaja. Näiden avulla voidaan muun muassa signaalin kytkentää määritellessä saada suoraan johtimien värit ja numerot näkyville.

Kuvassa 5 on esitetty ProElina-työkalun dokumenttien hallintaan tarkoitettu sivu. Tällä sivulla voidaan luoda ja hakea sekä muokata erilaisia dokumentteja. ProElina käyttää toimiakseen E-koodeja, joilla määritellään, minkä tyyppinen dokumentti on kyseessä. Esimerkiksi E440-koodi kuvaa piirikavioiden tyyppiä. Koska ProElinalla pystytään olemaan suorassa yhteydessä Share-dokumentinhallintajärjestelmään, on näillä koodeilla suuri merkitys. Tämä siitä syystä, että dokumenttienhallintaohjelmisto Sharessa dokumentit tallentuvat juuri näiden koodien avulla omiin kansioihinsa.

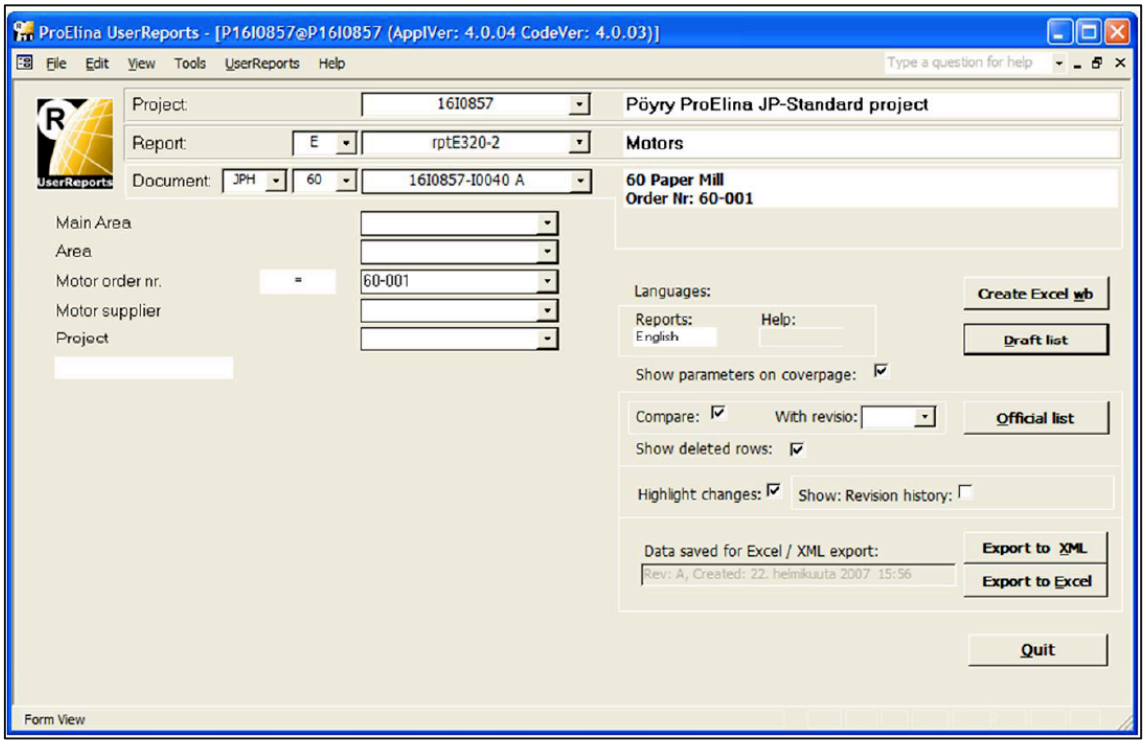


KUVA 5. ProElina-dokumenttisivu

ProElinaan on myös yhdistetty erillinen Microsoft Access -pohjainen luetteloiden generoimiseen tarkoitettu UserReports-työkalu. Se mahdollistaa erilaisten luetteloiden kuten dokumentti-, kaapeli- ja turvakytkinluetteloiden generoimisen. Näiden luetteloiden generoiminen säästää suunnittelijalta pääsääntöisesti runsaasti aikaa, koska jokaista yksittäistä muutosta ei tarvitse muuttaa monesta paikasta. Luetteloita generoitaessa käyttäjällä on myös mahdollisuus suorittaa vertailu ai-

kaisempiin versioihin dokumenteista, jolloin työkalu käy läpi edellisen revision luettelosta ja etsii eroavaisuudet. Tämän jälkeen työkalu pystyy värjäämään eri väreillä muuttuneet kohdat luetteloista, kuten poistuneen rivin tai tietyllä rivillä muuttuneen tiedon (ProElina, Sähkökoulutusmateriaali 2008, s. 24). Tällöin suunnittelijan tekemät virheet mahdollisessa käsin tehdyssä vertailussa vähentyvät.

Kuvassa 6 on esitetty UserReports-työkalun yleisnäkymä. Kuvassa ylälaidassa on varsinaisen generoitavan dokumentin tiedot. Dokumentin valinnan alapuolella ovat puolestaan generoimisen aloittavat painikkeet sekä vertailuun liittyvät valinnat. Vasemmassa laidassa on valitun dokumentin yleiset tiedot.



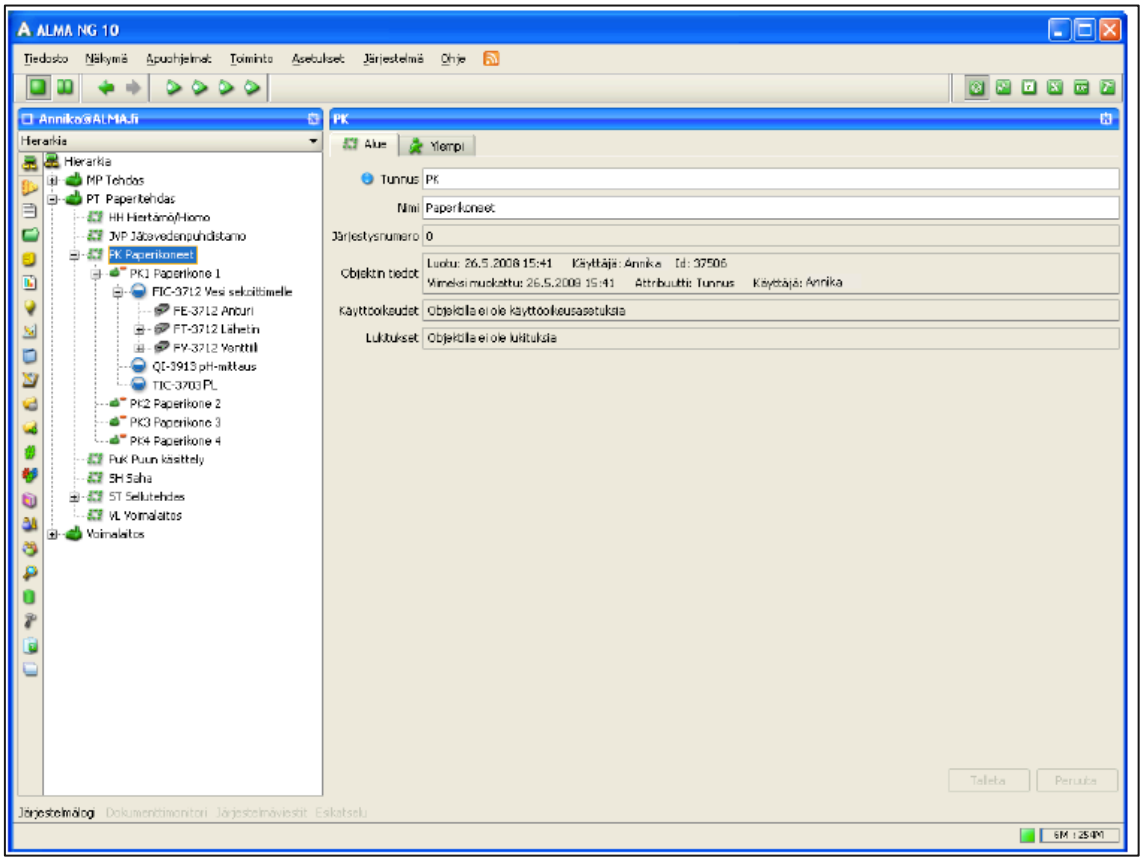
KUVA 6. UserReports-yleisnäkymä

Työkalulla saadaan tulostettua luettelot niin Excel-, XML- kuin PDF-muodossakin. Näihin muotoihin tulostettaessa saadaan vanhaan versioon verrattaessa luettelo, jossa muuttuneet tiedot ovat värjättynä ja identifioituna. Tämä puolestaan helpottaa huomattavasti muutoksien löytämistä. Mikäli muutoksia ei haluta näkyviin, voidaan ajaa myös niin sanottu väliaikainen luettelo.

3.2 ALMA®-suunnittelutyökalu

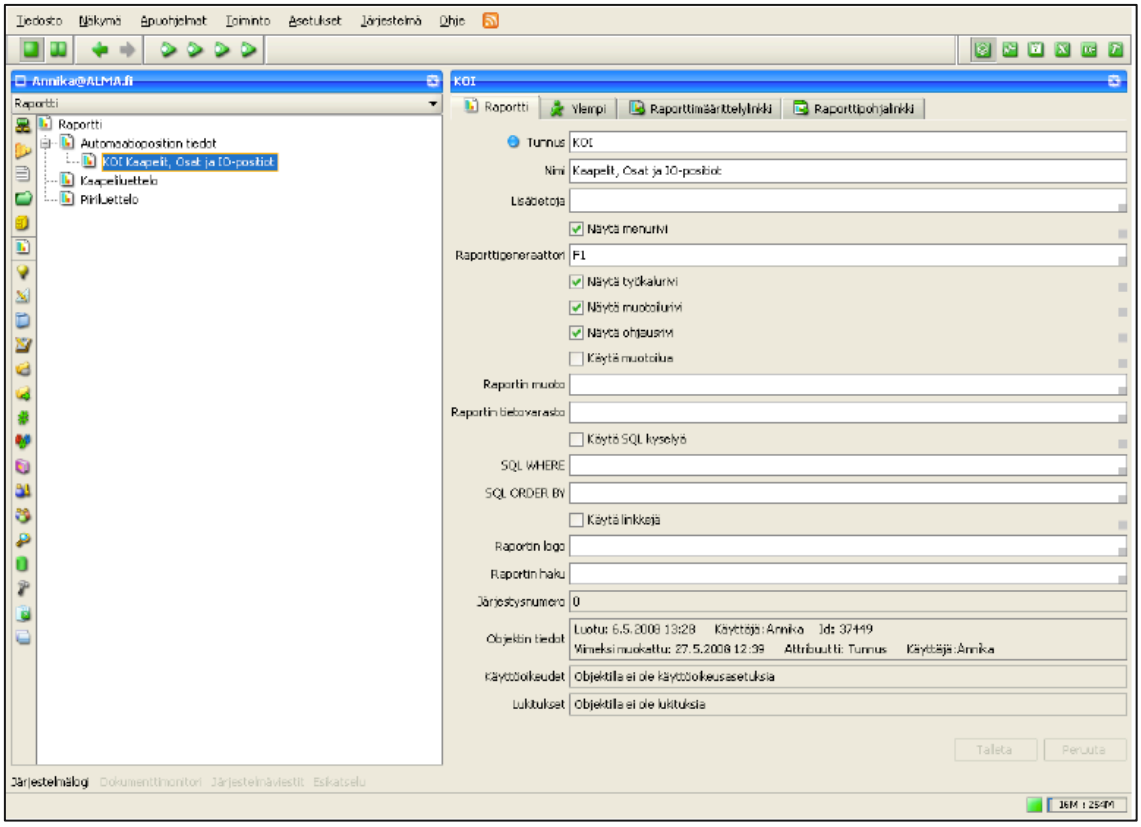
ALMA® on suomalaisen Alma Consulting Oy -yrityksen kehittämä työkalu suunnittelun, teknisen tiedon ja tapahtumien sekä kunnossapidon hallintaan (Alma, Tieto on kaikki kaikessa). ALMA® on kaupallinen hallintajärjestelmä, joka on käytössä jo monilla yrityksillä. ALMA® yhdistää monia eri ominaisuuksia kuten kunnossapidon, suunnittelun ja projektinhallinnan. Sillä pystytään siis käytännössä hoitamaan tehdasympäristöissä kaikki kunnossapidon hallinnasta ja suunnittelusta raportointiin. Tässä opinnäytetyössä tutustutaan ainoastaan ALMA®:n suunnitteluosuuteen.

Kuvassa 7 on esitetty ALMA®-työkalun yksi yleisnäkymistä. Kuvan vasemmassa reunassa nähdään työkalulle tyypillinen puuhierarkia. Tätä hierarkiaa on mahdollista muokata kunkin asiakkaan tottumuksien mukaan. Se voidaan muodostaa esimerkiksi niin, että pääkansio on paperitehdas ja tämän kansion alla omat kansiot jokaiselle paperikoneelle.



KUVA 7. ALMA®-yleisnäkymä (Alma käyttöohje, 42)

Kuvassa 8 on esitetty Almalla laadittavien raporttien käsittelyn ikkuna. Projektin raportit kootaan Almassa omaan raporttipuuhun. Raportti itsessään koostuu kolmenlaisesta eri objektista, jotka ovat: Raportti, Raporttimäärittelylinkki ja Raporttipohjalinkki. Raportti-objektissa on yleiset raportin määrittelyt, kuten esimerkiksi sen nimi ja erilaisia muotoiluun liittyviä tietoja. Raporttimäärittelylinkki-objektissa on puolestaan määritelty data, joka itse raporttiin haetaan. Raporttipohjalinkki-objektilla määritellään raportin muotoilu, joka tehdään erillisellä raporttieditorilla.



KUVA 8. ALMA® raportit (Alma käyttöohje, 61)

Almassa suunnittelu eroaa osittain huomattavastikin ProElina-työkalusta. Almassa jokainen piiri ”tuotteistetaan” eli esimerkiksi hierarkiassa ensimmäisen paperikoneen alle luodaan yksittäinen tuote eli piiri. Tämän jälkeen piireille pystytään muokkaamaan ja lisäämään kaapeleita sekä asettamaan erilaisia asetuksia. Työkalulla pystytään myös generoimaan piirikaaviot ja muut vastaavat kuvat .dxf-formaattiin pohjakuvien avulla.

4 OPINNÄYTETYÖN VAATIMUKSET JA TAVOITTEET

Työn päätavoitteena oli kehittää yrityksen suunnittelijoiden käyttöön työkalu, jolla helpotettaisiin suunnittelijoiden työtä erityisesti pienemmissä projekteissa. Koska ProElina- ja Alma-työkalujen käyttöönotto on joiltakin osin hankalaa, eivät ne näin ollen sovellu pienemmille kokonaisuuksille. Lisäksi nämä työkalut eivät kustannusmielessä ole pienille projekteille edes järkevä valinta.

Uudessa työkalussa haluttiin pitää erityisesti taulukkolaskentaohjelmiston hyödyt suunnittelijan käytettävissä. Käytännössä tämä tarkoitti sitä, että suunnittelija voi halutessaan tässä raportissa myöhemmin mainittaville tietokantaluetteloille kopioida esimerkiksi asiakkaan lähtötietoluettelosta tietoja. Tämä mahdollistaa myös taulukkolaskentaohjelmien kopiointi- ja vastaavat ominaisuudet, jotka ovat hyödyllisiä suunnittelijoille, kun kyseessä on monia samanlaisia piirejä.

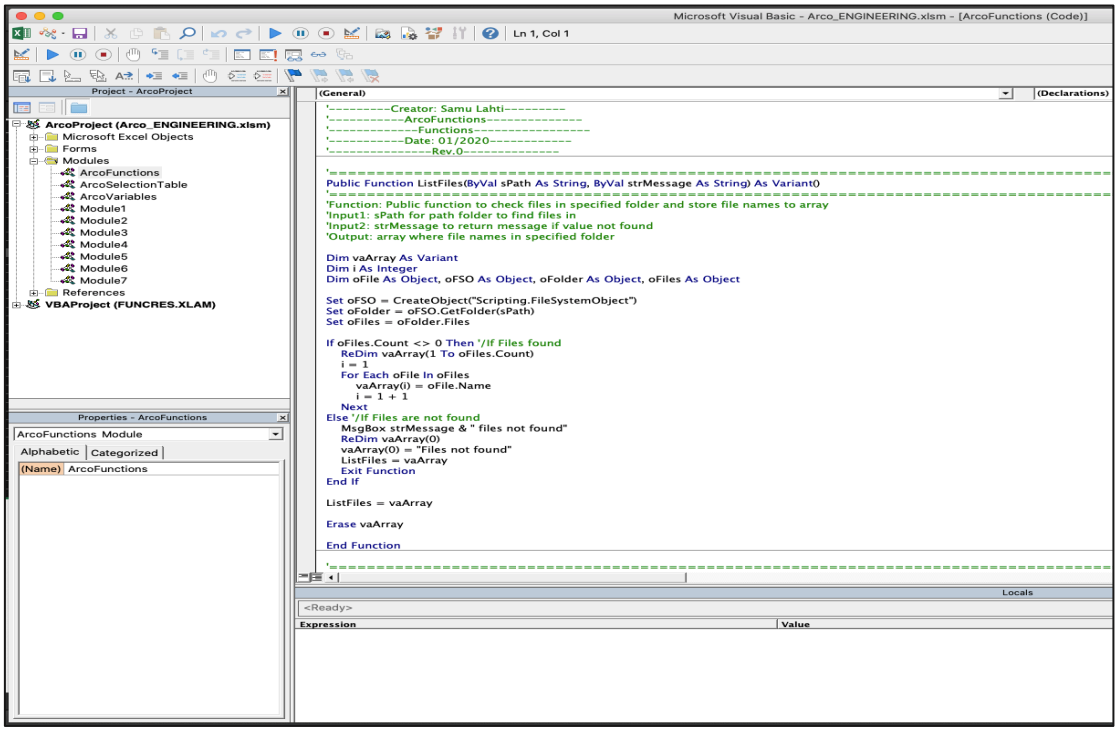
Työkalun tavoitteet ja vaatimukset luotiin yhdessä yrityksen työntekijöiden kanssa. Niitä myös seurattiin säännöllisesti pidettävien palaverien avulla, joissa käytiin läpi toteutettuja ominaisuuksia sekä niiden toimivuutta. Liitteessä 1 on esitetty työkalulle erikseen laadittu dokumentti vaatimuksista sekä tavoitteista. Tässä dokumentissa on erikseen eriteltynä ne ominaisuudet, jotka on sisällytetty tähän opinnäytetyöhön. Opinnäytetyön ulkopuolelle jätettävät ominaisuudet toteutetaan myöhemmin ja ovat osa työkalun jatkokehitystä. Toimintojen ohjelmoinnissa pyrittiin toteuttamaan mahdollisimman paljon erilaisia toimintoja, kuin että olisi toteutettu lukuisia ominaisuuksiltaan lähes samanlaisia toimintoja.

5 SUUNNITTELUTYÖKALU ARCO

5.1 Visual Basic for Applications -sovellus

VBA eli Visual Basic for Applications on Microsoftin kehittämä ohjelmointikieli. Se on tarkoitettu helpottamaan ja automatisoimaan erilaisia toimintoja varsinkin Excel- ja Word-ympäristöissä (Getting started with VBA in Office, Microsoft 2019). Sen etuna on, että käyttäjän ei tarvitse ladata erillisiä ohjelmistoja, vaan VBA-kieltä pystytään ohjelmoimaan yleisesti käytössä olevien Microsoftin Office 365 -paketissa mukana tulevien työkalujen avulla. Excelissä työkaluun pääsee käsiksi lisäämällä Developer-työkalut käyttöön erikseen asetuksista.

VBA mahdollistaa esimerkiksi valmiin Microsoft Word -pohjan täyttämisen taulukkolaskentaohjelmistoon täytetyillä tiedoilla. Sillä on siis mahdollista toimia myös eri sovellusten välillä ja lisäksi mahdollisuuksien mukaan sillä pystytään liittämään erilaisia ulkoisia dokumentinhallintajärjestelmiä Office365-paketin mukana tuleviin tuotteisiin. Kuvassa 9 on esitetty Office365-ympäristössä käytettävän alustan käyttöliittymä. Kuvan vasemmassa reunassa nähdään projektin rakenne ja eri moduulit ja keskiosassa varsinainen ohjelmoinnille tarkoitettu ikkuna.



KUVA 9. VBA-käyttöliittymä

VBA-ohjelmoinnissa käytetään eri toimintoihin niin sanottuja aliohjelmia. Kuvassa 10 on esitetty yleinen tapa luoda aliohjelma. Aliohjelmina voidaan käyttää muun muassa erilaisia funktioita, joita yleensä käytetään suorittaessa jokin toiminto useasti eri kohdissa varsinaista koodia. Funktio palauttaa aina ohjelmaan, josta sitä on kutsuttu, sille määritellyn muuttujatyypin. Yleisin aliohjelman muoto on kuitenkin VBA-ohjelmointikielessä "Sub"-aliohjelma. Tällaisen aliohjelman sisälle kirjoitetaan koodia, joka suoritetaan jostakin käyttäjän tekemästä toiminnosta, kuten esimerkiksi napin painalluksesta. Aliohjelma voi yleisesti olla joko julkinen (Public) tai yksityinen (Private). Julkista aliohjelmaa voidaan kutsua mistä tahansa kokonaisen ohjelman koodista, kun taas yksityistä aliohjelmaa voidaan kutsua ja käyttää ainoastaan siitä koodin osasta, johon se on määritetty.

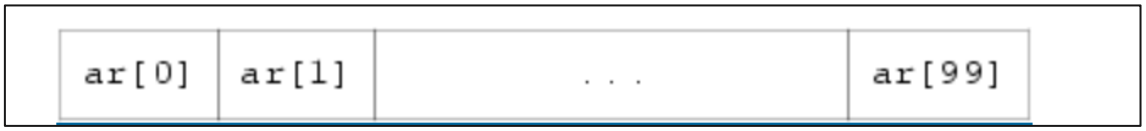
```
10    '====================================
11    Private Sub AddNewDocument_Click()
12    '====================================
13
14        '// add declarations
15
16    End Sub
17    '---------
```

KUVA 10. VBA-ohjelmointikielen aliohjelman kutsuminen

Ohjelmoidessaan ohjelmoijan tarvitsee usein kommentoida koodia ja VBA-kielessä se tapahtuu kirjoittamalla heittomerkki kommentin eteen. Tällöin koodia suorittaessa kääntäjä ohittaa kyseisen osan eikä ota sitä huomioon. Ohjelmakoodin kommentoinnilla voidaan selventää koodia huomattavasti ja se helpottaa kirjoitetun koodin ymmärtämistä ja muokkaamista myöhemmin esimerkiksi eri käyttäjille.

VBA-ohjelmoinnissa pystytään kutsumaan aliohjelmaa muutamilla eri tavoilla. Jos esimerkiksi jokin aliohjelma halutaan suorittaa käyttäjän kirjoittaessa tekstikenttään jokin arvo, voidaan se kutsua lisäämällä tekstikentän nimen perään loppupääte "_Change". Samalla tavalla voidaan kutsua aliohjelma esimerkiksi tilanteessa, jossa käyttäjä painaa jotakin painiketta, kun lisätään loppupääte "_Click".

Ohjelmoinnissa tulee usein vastaan tarve erilaisille muuttujille. Muuttuja on jokin ohjelmoijan luoma muuttuja, johon pystytään tallentamaan erilaisia tietoja eri kohdassa ohjelmaa. Yleisimmät muuttujien tyypit ovat string, integer, boolean ja array. String-tyyppinen muuttuja voi sisältää kirjaimia, numeroita ja jopa välilyöntejä. Käytännössä string-tyyppisellä muuttujalla voidaan hoitaa suuri osa ohjelmoinnista. Integer-tyyppiseen muuttujaan pystytään puolestaan tallentamaan eri lukuarvoja. Integer-tyyppinen muuttuja on yleisesti helpompi ja ohjelmallisesti kevyempi käsiteltävä, kun on tarve ainoastaan numeerisille arvoille. Boolean-tyyppiseen muuttujaan on mahdollista tallentaa kaksi eri arvoa, joko tosi (true) tai epätosi (false). Tätä muuttujaa voidaan siis yleisesti käyttää, kun tarkastetaan, toteutuuko jokin ehto. Array-tyyppinen muuttuja on kokoelma jonkin tietyn muuttujatyypin muuttujista. Se voidaan siis ajatella esimerkiksi kokoelmaksi, joka sisältää 100 string-tyyppistä muuttujaa. Kuvassa 11 on esitetty tällaisen muuttujan rakenne. Array-tyyppisellä muuttujalla on ohjelmoinnin kannalta tärkeää huomata, että sen tiedot alkavat paikasta nolla eikä yksi.



KUVA 11. 100 elementtisen array-muuttujan rakenne (GBdirect, The C book)

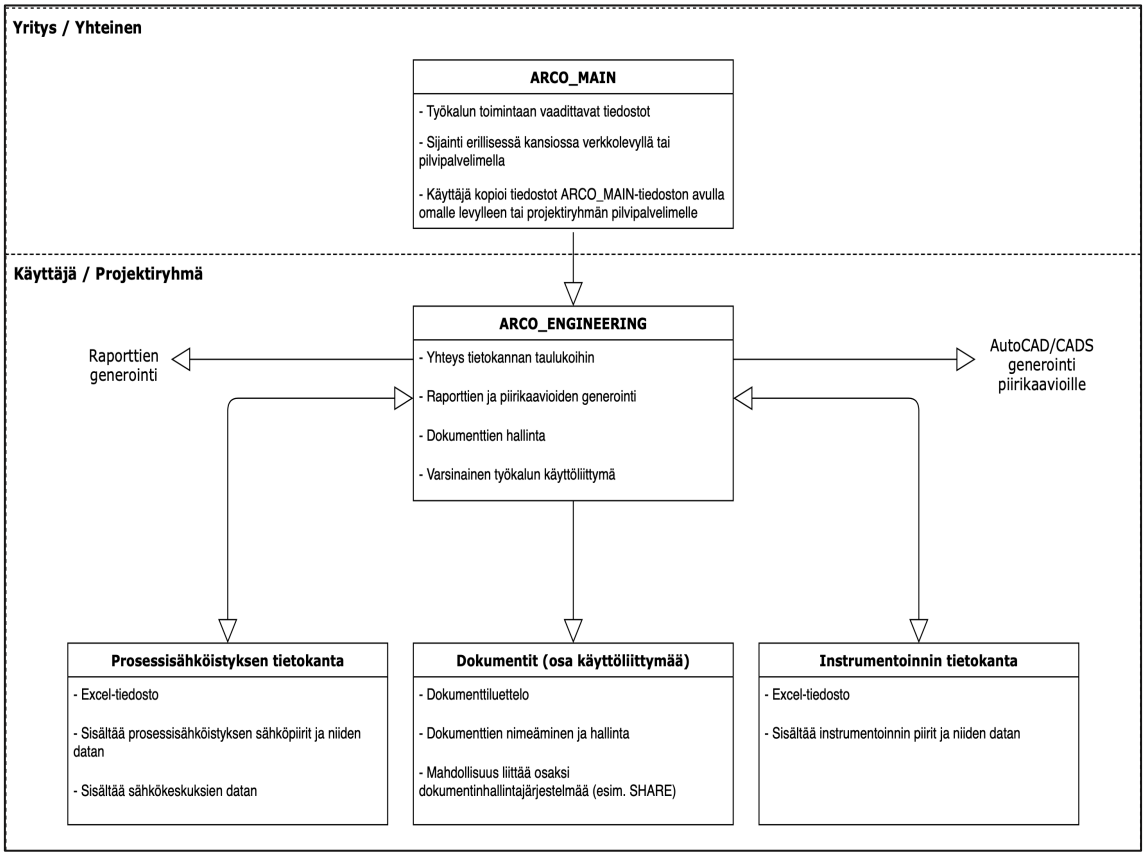
Muuttujat luodaan yleensä jonkin aliohjelman sisällä käyttäen dim-käskyä. Tällä käskyllä käyttäjä määrittää muuttujan nimen ja sen tyypin. Kuvassa 12 on esitetty tyypillinen esimerkki muuttujan määrittämisestä, jossa muuttuja "TrueOfFalse" määritellään boolean-tyyppiseksi.

```
10    'Declare variable
11    Dim TrueOfFalse As Bool
12
```

KUVA 12. Muuttujan määrittäminen

5.2 Käyttöliittymä ja rakenne

Työkalun päätavoitteena oli suunnittelutyön helpottamisen ohella säilyttää taulukkolaskentaohjelman edut, kuten helppous sekä käytettävyys. Käytännössä tämä tarkoitti sitä, että työkalulla tulisi olla mahdollista toteuttaa suunnittelua myös perinteisesti ilman käyttöliittymää, jolloin varsinainen käyttöliittymä irrotettaisiin itse työkalusta. Tämä ominaisuus toteutettiin erillisten tietokantana toimivien Excel-tiedostojen avulla. Kaikki projektissa käytettävä data siis säilytetään kahdessa erillisessä Excel-tiedostossa, joista toinen on prosessisähköistykselle ja toinen instrumentoinnille. Näitä tietokantoja pystytään puolestaan erillisen käyttöliittymänä toimivan Excelin avulla hallitsemaan ja varsinainen raporttien tuottaminen tapahtuu käyttöliittymän kautta. Tällöin suunnittelijalle mahdollistetaan työskentely myös ainoastaan tietokanta tiedostoilla ilman käyttöliittymää. Kuvassa 13 on esitetty työkalun päärakenne, jossa työkalun keskeisimmät toiminnot.



KUVA 13. Työkalun päärakenne

Työkalun helpon käyttöönottamisen takaamiseksi luotiin kaksi erillistä Excel-tiedostoa. Kuvassa 10 esiintyvällä "ARCO_MAIN"-tiedostolla pystytään varsinainen työkalu ja siihen tarvittavat lisäosat kopioimaan suunnittelijan omalle tietokoneelle tai tarvittaessa projektiryhmän pilvipalveluun, kuten esimerkiksi OneDrive-sivustolle. Varsinainen käyttöliittymä ja sen toiminnot ovat sijoitettuna "ARCO_ENGINEERING"-tiedostoon, joka kommunikoi prosessi- ja instrumentointisähköistyksen tietokantana toimivien tiedostojen kanssa ja jolla pystytään luomaan raportit sekä generoimaan piirustuksia.

Kuvassa 14 on esitetty "ARCO_MAIN"-tiedoston yleisnäkymä. "START"-painikkeella käyttäjä pystyy aloittamaan työkalun siirron itselleen ja "HELP"-painikkeella käyttäjä pääsee työkalun käyttöohjeeseen.

KUVA 14. "ARCO_MAIN"-tiedoston yleisnäkymä

Kuvassa 15 on esitetty tiedostojen kopioinnin suorittava ohjelmistokoodi. Koodissa rivillä seitsemän suoritetaan muuttujaan "fileExplorer" kansion valintaobjektin lisääminen. Sen avulla pystytään käyttäjältä kysymään haluttu kansio, johon työkalu tallennetaan. Tämän jälkeen valittu kansio tallennetaan "selectedFolder"-muuttujaan string-tyyppisenä rivillä 13. Toiseen samaa tyyppiä olevaan muuttujaan "ProjectName" kysytään "InputBox"-komennolla käyttäjältä projektin nimi rivillä 14. Projektin nimeä käytetään pääasiassa kansion luomisessa antamaan kansiolle nimi. Rivillä 15 muodostetaan lopullinen kansion polku muuttujaan "dFolder", jota käytetään myöhemmin. Koska työkalun toimimiseen tarvittavat lisäosat sijaitsevat avatun tiedoston juurikansiossa, pystytään rivillä 24 hakemaan muuttujaan "sFolder" tiedoston polku komennolla "ThisWorkbook.Path". Lopullinen tarvittavien lisäosien polku saadaan lisäämällä avatun tiedoston polkuun pääte "\Tools" eli kansio, jossa ne sijaitsevat. Riveillä 27-28 tarkistetaan, onko vastaava kansio jo luotu käyttäjän määrittämään polkuun sekä annetaan käyttäjälle tarvittaessa viesti kansion löytymisestä eikä luoda päälle uutta kansiota. Riveillä 29-32 luodaan puolestaan uusi kansio käyttäjän määrittämään polkuun sekä annetaan lopuksi viesti onnistuneesta siirrosta.

```
7   Set fileExplorer = Application.FileDialog(msoFileDialogFolderPicker) 'Let's ask users specified folder to store application
8
9   fileExplorer.AllowMultiSelect = False 'Do not allow multi select
10
11  With fileExplorer
12      If .Show = -1 Then 'If any folder is selected
13          selectedFolder = .SelectedItems(1)
14          ProjectName = InputBox("Please Enter Project name", "Project name", "Arco Project") 'Lets ask user what's the project's name
15          dFolder = (selectedFolder & "\" & ProjectName)
16          MsgBox ("Tools will be saved to: " & dFolder)
17      Else 'Else dialog is cancelled and closed and user will be informed
18          MsgBox "Cancelled, no folder selected"
19          Exit Sub
20      End If
21  End With
22
23  'Creating folder and copying tools with user given project name
24  sFolder = (Application.ThisWorkbook.Path & "\Tools") 'Specify where tools are using thisworkbook's path as a reference
25  Application.ScreenUpdating = False 'Disables screen updating while running code
26  Set fbObj = CreateObject("Scripting.FilesystemObject")
27  If fbObj.FolderExists(dFolder) Then
28      MsgBox "Folder already found", vbInformation, "Folder found"
29  Else
30      fbObj.CreateFolder (dFolder)
31      fbObj.Copyfolder sFolder, dFolder 'Copying necessary files to user specified path
32      MsgBox "Folder created successfully and all the necessary tools copied", vbExclamation, "Done!"
33  End If
```

KUVA 15. VBA-koodi tiedostojen kopiointiin

Työkalun ja sen tiedostojen kopioimisen jälkeen "ARCO_MAIN"-tiedosto sulkeutuu ja käyttäjälle avataan näkyviin lopulliseen polkuun tallennettu varsinainen käyttöliittymän Excel-tiedosto "ARCO_ENGINEERING". Varsinaiseen käyttöliittymän tiedostoon on eri välilehdille sijoitettu eri osa-alueiden toimintoja. Käyttäjä pystyy siis typpilliseen taulukkolaskentaohjelmiston tapaan siirtymään välilehdeltä toiselle niin halutessaan. Tämän lisäksi käyttöliittymässä on erilliset painikkeet, joilla käyttäjä pystyy liikkumaan eri osa-alueiden välillä. Kuvassa 16 on esitetty esimerkki ohjelmakoodista, jossa käyttäjän painaessa käyttöliittymän painiketta, siirrytään osa-alueelta toiselle eli aktivoidaan kyseinen välilehti.

```
455    '====================================
456    Sub ElectEngineering_Click()
457    '====================================
458
459    Sheet2.Activate  'Active Electrical Engineering sheet'
460
461    End Sub
462    '-----------------ENDSUB--------------------------
```

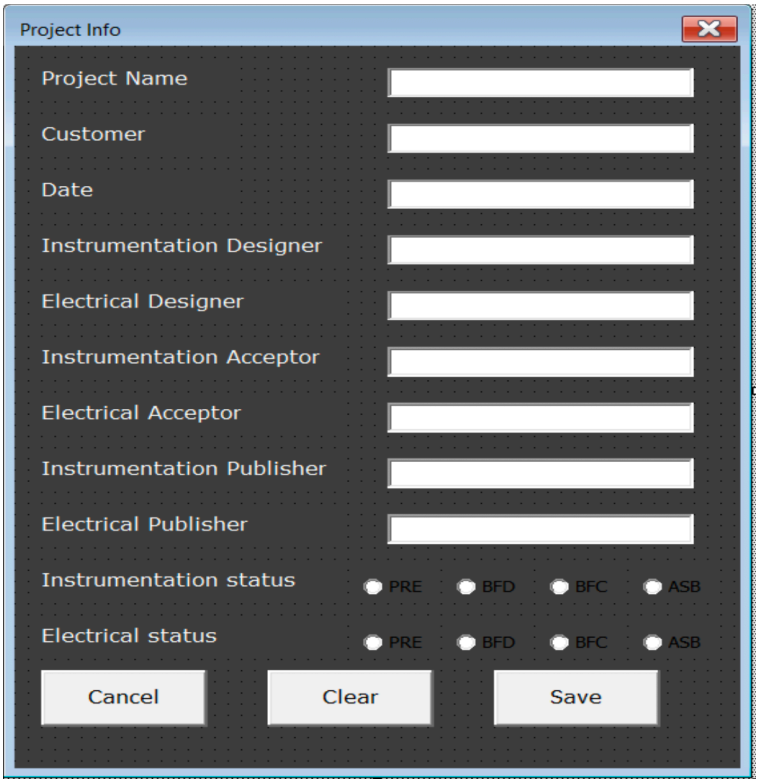
KUVA 16. Ohjelmakoodi siirtymisestä osa-alueelta toiselle

Taulukkoon 1 on kirjattu työkalun käyttöliittymään sijoitettuja välilehtiä eli työkalun osa-alueita ja niiden selityksiä. Jokaisen välilehden tarkoituksena on joko säilyttää joitakin projektin kannalta tärkeitä tietoja tai olla osana käyttäjälle näkyvää käyttöliittymää. Esimerkiksi "ProjectComponents"-välilehdellä pystytään säilyttämään projektissa tarvittavat komponentit. Sinne on mahdollisuus tallentaa esimerkiksi sähkökeskuksissa käytettävät kontaktorien ja moottorinsuojakytkinten mallit ja eri kokoluokat. Näin käyttäjällä on mahdollisuus suunnittelun edetessä käyttää valmiiksi luotuja komponentteja eikä etsiä niitä erikseen. Tätä voidaan siis pitää eräänlaisena komponenttivarastona, joka työkalun käyttöiän mukaan laajenee. Välilehdillä pyritään myös erottamaan työkalun eri osa-alueet selvästi toisistaan, jolloin varsinainen käyttöliittymä saadaan pidettyä yksinkertaisena ja selkeänä. Tämä helpottaa suunnittelijaa toimintojen etsimisessä ja yleisessä työkalun käyttämisessä, kun informaatiota ei koota liikaa yhteen näkymään.

TAULUKKO 1. Suunnittelutyökalun eri välilehdet

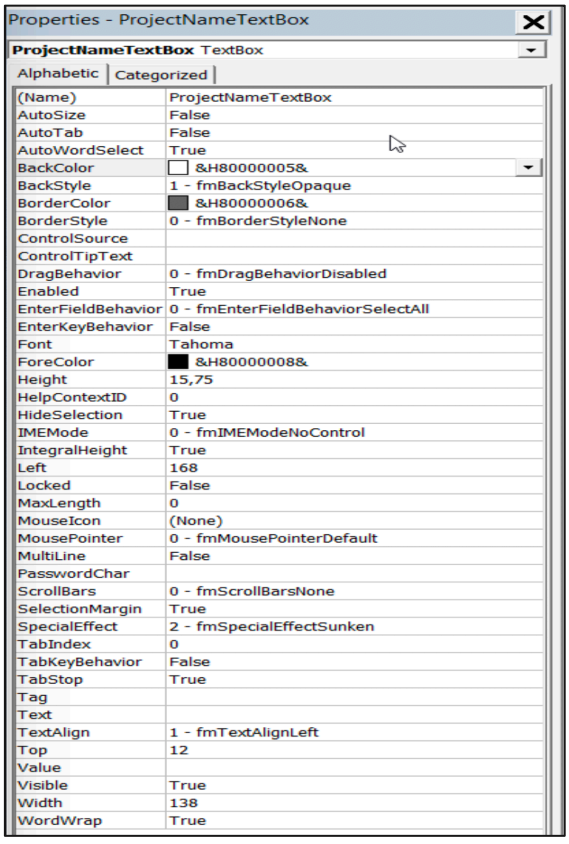
Välilehti	Selite
USER	Pääikkuna
Electrical Engineering	Prosessisähköistyksen ikkuna
Instrumentation Engineering	Instrumentoinnin ikkuna
SelectionTables	Prosessisähköistyksen valintataulukot
Project Info	Ikkuna, jossa on kaikki yleinen tieto
Documents	Dokumenttienhallinnan ikkuna
ProjectComponents	Ikkuna, jossa voidaan säilyttää kaikki projektissa käytettävät komponentit ja niiden tiedot.

Tietojen täydentämiseen ja luetteloiden sekä piirikaavioiden generoimiseen työkalussa käytetään erilaisia lomakkeita, joita kutsutaan VBA-koodissa nimellä "UserForms". Kuvassa 17 on esitetty lomake, jossa käyttäjää kehotetaan täyttämään yleiset tiedot projektille.



KUVA 17. Projektin tietojen täyttölomake

Näiden lomakkeiden laatimisessa ohjelmoijalla on todella paljon erilaisia mahdollisuuksia. Lomakkeisiin voidaan pääasiassa sijoittaa erilaisia tekstikenttiä, sekä painikkeita, joista voidaan ohjelmoida suoritettavaksi erilaisia toimintoja. Lisäksi lomakkeisiin voidaan sijoittaa erilaisia luetteloita, valikkoja ja kyllä tai ei -valintoja. Kuvassa 18 on esitetty yleisen tekstikentän asetuksia. Tekstikentälle, kuten kaikille lomakkeisiin sijoitettaville objekteille, on tärkeää antaa muuttujan nimi kohdassa "(Name)". Esimerkiksi kuvassa 18 kentälle, johon käyttäjä pystyy täyttämään projektin nimen, on annettu muuttujan nimi "ProjectNameTextBox". Tätä muuttujan nimeä pystytään puolestaan hyödyntämään ohjelmakoodissa eri toimintoihin, kuten esimerkiksi viemään käyttäjän antama tieto johonkin soluun Excelissä. Lisäksi tekstikentille pystytään määrittämään, onko se käyttäjän muokattavissa ja millaista fonttia siinä käytetään. Kaikille objekteille on tärkeää asettaa myös kohtaan "TabIndex" objektin järjestysnumero täyttölomakkeessa. Järjestysnumerolla pystytään määrittämään järjestys, jossa lomakkeen tekstikentissä edetään, kun niiden välillä siirrytään käyttäen tietokoneen tab-näppäintä.



KUVA 18. Tekstikentän asetukset

Käyttäjän aktivoidessa lomakkeen, suoritetaan koodissa ensimmäisenä varsinaisen lomakkeen alustaminen. Alustamisella varmistetaan, että tekstikenttiin ja muihin objekteihin tuodaan käyttäjän saataville tuorein tieto. Lomakkeen alustaminen saadaan suoritettua luomalla aliohjelma lomakkeen nimellä sekä lisäämällä nimen perään loppuosa "_Initialize". Kuvassa 19 on esitetty koodin pääkohdat projektin tietojen täyttölomakkeen alustamisessa. Riveillä 12-16 etsitään "ProjectInfo"-välilehdellä solusta A2 arvo. Mikäli solusta löytyy jokin arvo, täytetään solusta löytynyt arvo lomakkeen tekstikenttään. Samalla ohjelmakoodilla pystytään toteuttamaan kaikkien kenttien täyttö. Rivillä 19 haetaan rivillä yhdeksän määriteltyyn muuttujaan "InstStatus" arvoa. Arvo haetaan käyttäen "Cells(x,x)"-komentoa. Komennossa pystytään määrittämään tarkka solu määrittämällä ensin haluttu rivi ja sen jälkeen pilkulla erotettuna haluttu sarake. Mikäli solusta löytyy jokin riveillä 23, 25, 27 tai 29 mainituista teksteistä, aktivoidaan sen mukaan lomakkeesta kyllä tai ei -valinnan objekti. Valinta tapahtuu käyttäen "Case"-komentoa, joka on C-kielestä tutun "switch"-komennon vastine. Käytännössä sillä pystytään suorittamaan erilaisia toimintoja ennakkoon valituilla ehdoilla (Select Case statement 2018).

```
1   '=============================================================================
2   Private Sub UserForm_Initialize()
3   '=============================================================================
4   'Function: Initialize ProjectInfo Userform
5   'Input1: -
6   'Output: -
7   'Version History:
8
9   Dim InstStatus As String
10
11  'Find value for Project Name if any'
12  If IsEmpty(Range("A2")) = True Then
13      ProjectNameTextBox.Value = ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 1).Value
14  Else
15      ProjectNameTextBox.Value = ""
16  End If
17
18  'Define Inst- & ElectStatus
19  InstStatus = ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 10).Value
20
21  'Case to select which status is selected for instrumentation
22  Select Case InstStatus
23      Case "PRE"
24          StatusPREInst = True
25      Case "BFD"
26          StatusBFDInst = True
27      Case "BFC"
28          StatusBFCInst = True
29      Case "ASB"
30          StatusASBInst = True
31      Case Else
32          StatusPREInst = False
33          StatusBFDInst = False
34          StatusBFCInst = False
35          StatusASBInst = False
36  End Select
```

KUVA 19. Projektin tietojen täyttölomakkeen alustaminen

Lomakkeeseen sijoitetulla "CLEAR"-painikkeella käyttäjä pystyy nollaamaan kaikki täytetyt tiedot. Kuvassa 20 on esitetty lomakkeen tietojen tyhjentämiseen käytettävä koodi. Koodissa rivillä 10 kysytään "MsgBox"-komennolla, haluaako käyttäjä varmasti suorittaa lomakkeen tyhjentämisen. Tällä pystytään siis estämään, ettei käyttäjä vahingossa tyhjennä lomaketta. Riveillä 11-21 suoritetaan varsinainen arvojen tyhjentäminen jokaiselle solulle, jossa arvot sijaitsevat. Tyhjentämisen jälkeen riveillä 25-33 päivitetään jokaisen tekstikentän tiedot uusilla tyhjennetyllä arvoilla.

```
1  '================================================================================================================
2  Private Sub ClearButton_Click()
3  '================================================================================================================
4  'Function: Clear all values in project info userform
5  'Input1: -
6  'Output: -
7  'Version History:
8
9  'Load clear value to Cells'
10 If MsgBox("All info will be deleted. Do you want to continue?", vbYesNoCancel) = vbYes Then 'Make sure that user didn't press button by accident
11     ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 1).Value = "" 'Clear Project Name
12     ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 2).Value = "" 'Clear Customer Name
13     ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 3).Value = "" 'Clear Date
14     ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 4).Value = "" 'Clear Instrumentation Designer Name
15     ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 5).Value = "" 'Clear Electrification Designer Name
16     ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 6).Value = "" 'Clear Instrumentation Acceptor Name
17     ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 7).Value = "" 'Clear Electrification Acceptor Name
18     ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 8).Value = "" 'Clear Instrumentation Publisher Name
19     ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 9).Value = "" 'Clear Electrifcation Publisher Name
20     ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 10).Value = "" 'Clear Instrumentation Status
21     ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 11).Value = "" 'Clear Electrification Status
22 End If
23
24 'Update all the text boxes regarding cell value
25 ProjectNameTextBox.Value = ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 1).Value
26 CustomerTextBox.Value = ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 2).Value
27 DateTextBox.Value = ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 3).Value
28 InstDesignerTextBox.Value = ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 4).Value
29 ElectDesignerTextBox.Value = ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 5).Value
30 InstAcceptorTextBox.Value = ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 6).Value
31 ElectAcceptorTextBox.Value = ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 7).Value
32 InstPublisherTextBox.Value = ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 8).Value
33 ElectPublisherTextBox.Value = ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 9).Value
34
35 End Sub
36 '-----------------ENDSUB--------------------
```

KUVA 20. Lomakkeen tyhjentämiseen käytetty koodi

Kuvassa 21 on esitetty tietojen tallentamiseen luotu ohjelmakoodi. Kun käyttäjä on täyttänyt kaikki tarvittavat tiedot lomakkeeseen, voidaan "SAVE"-painikkeella viedä kaikki tiedot talteen "ProjectInfo"-välilehdelle. Koodissa riveillä 5-15 määritellään instrumentoinnin suunnittelun vaihe. Tämä vaihe tarkastetaan siten, että mikäli käyttäjä on valinnut minkä tahansa objekteista aktiiviseksi, sijoitetaan "InstStatusInt"-muuttujaan vaihetta vastaava luku. Muuttuja on määritetty integer-tyyppiseksi muuttujaksi, jolloin siihen on mahdollista sijoittaa kokonaislukuja.

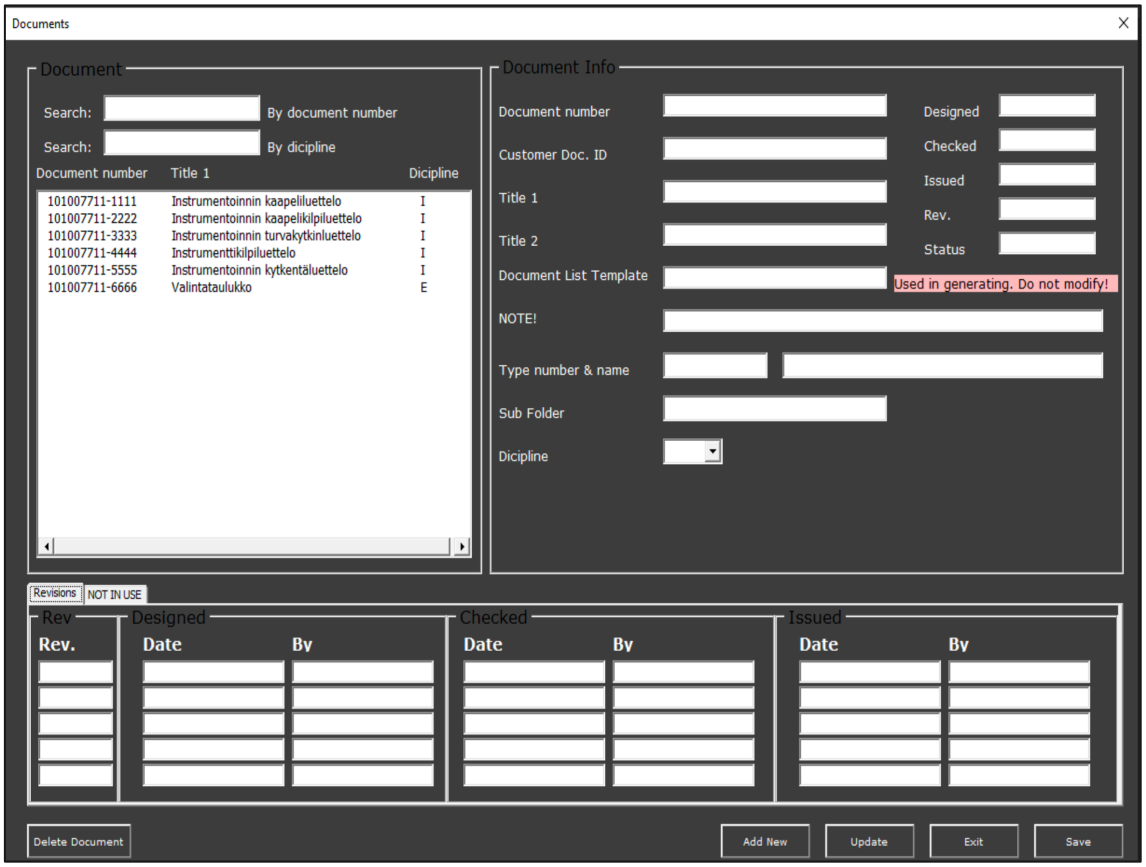
Muuttujaan on määritelty luvut yhdestä neljään, jolloin jokainen numero vastaa tiettyä suunnittelun vaihetta. Ohjelmakoodissa riveillä 18-29 näitä numeroita hyödynnetään käyttäen aikaisemmin mainittua "Case"-komentoa. Jokaisella eri arvolla pystytään siis sijoittamaan suunnitteluvaiheen kirjainyhdistelmä talteen. Varsinaisten tekstikenttien tallentaminen välilehden soluihin on suunnitteluvaiheen tallentamista yksinkertaisempi operaatio. Tekstikentän tallentamiseksi voidaan suoraan viitata halutun solun arvoon itse tekstikentän arvolla, kuten ohjelmakoodissa on riveillä 32-40 tehty.

```
1   Dim emptyRow As Long
2   Dim InstStatusInt As Integer, ElectStatusInt As Integer
3
4   'Define Instrumentation status integer for selected status
5   If StatusPREInst = True Then
6       InstStatusInt = 1
7   ElseIf StatusBFDInst = True Then
8       InstStatusInt = 2
9   ElseIf StatusBFCInst = True Then
10      InstStatusInt = 3
11  ElseIf StatusASBInst = True Then
12      InstStatusInt = 4
13  Else
14      MsgBox "Problem with instrumentation status, please try again"
15  End If
16
17  'Make case for instrumentation status selected
18  Select Case InstStatusInt
19      Case 1
20          ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 10).Value = "PRE"
21      Case 2
22          ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 10).Value = "BFD"
23      Case 3
24          ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 10).Value = "BFC"
25      Case 4
26          ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 10).Value = "ASB"
27      Case Else
28          MsgBox "Problem with instrumentation status, please try again"
29  End Select
30
31  'Load given information to cells
32  ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 1).Value = ProjectNameTextBox.Value
33  ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 2).Value = CustomerTextBox.Value
34  ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 3).Value = DateTextBox.Value
35  ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 4).Value = InstDesignerTextBox.Value
36  ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 5).Value = ElectDesignerTextBox.Value
37  ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 6).Value = InstAcceptorTextBox.Value
38  ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 7).Value = ElectAcceptorTextBox.Value
39  ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 8).Value = InstPublisherTextBox.Value
40  ThisWorkbook.Worksheets("ProjectInfo").Cells(2, 9).Value = ElectPublisherTextBox.Value
```

KUVA 21. Lomakkeen tietojen tallentaminen

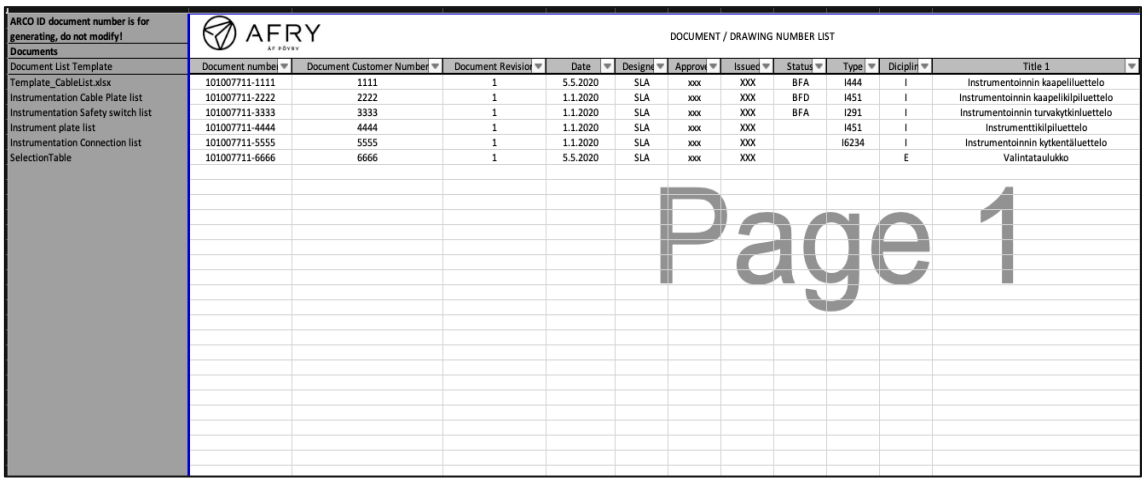
5.3 Dokumenttien hallinta

Työkalussa pyritään säilyttämään myös ProElina-työkalusta tuttu dokumenttien hallinta. Uudistetussa työkalussa dokumenttien hallinta toteutetaan käyttäen täyttölomaketta, kuten projektin yleisten tietojen täydentämisessäkin. Kuvassa 22 on esitetty tätä varten laadittu täyttölomake. Kun tätä verrataan aikaisemmin kuvassa 5 esitettyyn ProElinan dokumenttisivuun, voidaan huomata suuri yhtäläisyys. Tämän yhtäläisyyden tarkoituksena on helpottaa vanhoja työntekijöitä uuden työkalun toimintojen omaksumisessa. Lomakkeessa erilliseen luettelokenttään saadaan esiin kaikki projektissa määritellyt dokumentit. Käyttäjän aktivoidessa jonkin dokumenteista, haetaan oikealla oleviin tekstikenttiin kyseisen dokumentin tiedot. Lisäksi dokumentteja pystytään hakemaan erikseen kirjoittamalla hakukenttään etsittävän dokumentin dokumenttinumero kokonaan tai osittain.



KUVA 22. Dokumenttihallinnan täyttölomake

Kuvassa 23 on esitetty käyttöliittymän välilehti, johon dokumenttien tiedot kootaan. Välilehti on rakennettu siten, että käyttäjällä on mahdollisuus tulostaa siitä suoraan asiakkaille jaettava dokumenttiluettelo. Täyttölomakkeella tehtävät muutokset dokumentteihin tulevat käyttäjän tallentaessa suoraan näkyviin tälle dokumenttien välilehdelle. Lisäksi käyttäjä pystyy suoraan välilehdellä tekemään muutoksia dokumentteihin sekä lisäämään ja poistamaan niitä. Tämä oli työkalun kannalta tärkeä ominaisuus, jolla pyrittiin säilyttämään taulukkolaskentaohjelman edut.



ARCO ID document number is for generating, do not modify!
Documents
Document List Template
Template_CableList.xlsx
Instrumentation Cable Plate list
Instrumentation Safety switch list
Instrument plate list
Instrumentation Connection list
SelectionTable

AFRY

DOCUMENT / DRAWING NUMBER LIST

Document number	Document Customer Number	Document Revision	Date	Designe	Approve	Issued	Status	Type	Diciplin	Title 1
101007711-1111	1111	1	5.5.2020	SLA	xxx	XXX	BFA	I444	I	Instrumentoinnin kaapeliluettelo
101007711-2222	2222	1	1.1.2020	SLA	xxx	XXX	BFD	I451	I	Instrumentoinnin kaapelikilpiluettelo
101007711-3333	3333	1	1.1.2020	SLA	xxx	XXX	BFA	I291	I	Instrumentoinnin turvakytkinluettelo
101007711-4444	4444	1	1.1.2020	SLA	xxx	XXX		I451	I	Instrumenttikilpiluettelo
101007711-5555	5555	1	1.1.2020	SLA	xxx	XXX		I6234	I	Instrumentoinnin kytkentäluettelo
101007711-6666	6666	1	5.5.2020	SLA	xxx	XXX			E	Valintataulukko

KUVA 23. Dokumenttien välilehti

Kuvassa 24 on ohjelmakoodin pääosat, joilla alustetaan dokumenttien hallinnan täyttölomake. Koodissa ensimmäisenä riveillä 11 ja 12 sijoitetaan muuttujaan "Workbook" käyttöliittymän Excel-tiedoston nimi ja muuttujaan "WorkSheet" välilehden nimi. Näiden arvojen sijoittaminen muuttujiin on ohjelmoinnin kannalta hyödyllistä, koska tulevaisuudessa, jos esimerkiksi välilehden nimeä vaihdetaan, ei sitä tarvitse koodissa vaihtaa kuin yhdessä paikassa. Rivillä 14 haetaan "LastDocumentRow"-muuttujaan dokumenttiluettelossa oleva viimeinen rivi, jolla tietoja on. Rivillä 16 haetaan "DocumentColumn"-muuttujaan sarake, jossa sijaitsee "Document number"-teksti. Koska ohjelmakoodissa tullaan useasti hakemaan viimeistä riviä tai saraketta jostakin taulukosta, on sitä varten luotu erillinen funktio. Riveillä 23-43 on esitetty erikseen viimeisen rivin hakemiseen tarvittavan funktion ohjelmakoodi. Funktioita luodaan silloin, kun koodissa suoritetaan jokin tietty toiminto useita kertoja. Rivillä 20 on funktion aloitus, jossa määritellään, mitä tietoja sen sisälle tuodaan alkuperäisestä koodista. Tässä tapauksessa funktioon

tulee tuoda tiedoston nimi, välilehti, sarake ja tarvittaessa etsittävä arvo. Riveillä 24 ja 25 haku keskeytetään, mikäli funktiota kutsuttaessa ei ole määritelty tiedoston nimeä tai välilehteä. Riveillä 27-37 haetaan puolestaan viimeinen rivi joko kokonaiselta välilehdeltä, sarakkeesta tai etsien tiettyä arvoa. Funktiota kutsuttaessa se palauttaa rivin numeron long-tyyppisenä muuttujana, joka on integer-tyyppinen muuttuja sillä erolla, että se pystyy säilömään suuremman luvun (Long data type (Visual Basic), 2018).

```
10  'Declare workbook and worksheets to variables
11  Workbook = "Arco_ENGINEERING.xlsm"
12  Worksheet = "Documents"
13  'Pick up last used document row
14  LastDocumentRow = GetLastRow(Workbook, Worksheet, 2, "")
15  'Specify which column has each values in Dcouments sheet
16  DocumentColumn = GetLastColumn(Workbook, Worksheet, 4, "Document number")
17  DocumentTitle1Column = GetLastColumn(Workbook, Worksheet, 4, "Title 1")
18  DocumentTitle2Column = GetLastColumn(Workbook, Worksheet, 4, "Title 2")
19  '=============================================================================================================
20  Public Function GetLastRow(ByVal strWorkbook As String, ByVal strWorkSheet As String, ByVal strColumn As Long, Optional ByVal strSearchValue As String) As Long
21  '=============================================================================================================
22
23  On Error GoTo catchError
24  If strWorkbook = vbNullString Then Exit Function
25  If strWorkSheet = vbNullString Then Exit Function
26  If strSearchValue = vbNullString Then
27      If strColumn = 0 Then
28          'Find last used row in a sheet.
29          GetLastRow = Workbooks(strWorkbook).Worksheets(strWorkSheet).Cells.Find("*", SearchOrder:=xlByRows, SearchDirection:=xlPrevious).Row
30      Else
31          'Find last used row in a column
32          GetLastRow = Workbooks(strWorkbook).Worksheets(strWorkSheet).Cells(Rows.Count, strColumn).End(xlUp).Row
33      End If
34  Else
35      'Find last used row in a column with search value
36      GetLastRow = Workbooks(strWorkbook).Worksheets(strWorkSheet).Cells(strColumn).EntireColumn.Find(What:=strSearchValue, LookIn:=xlValues).Row
37  End If
38  exitSub:
39      Exit Function
40  catchError:
41      '/ Give user error message
42      MsgBox "Unidentified error occured while trying to find last Row, please try again", vbOKOnly, "ERROR"
43      GoTo exitSub
44
```

KUVA 24. Lomakkeen alustus ja aliohjelma

Myöhemmin täyttölomakkeen alustamisessa määritellään sarakkeiden määrä sekä käytettävän fontin koko dokumenttiluetteloon. Kuvassa 25 on esitetty riveillä 2-4 koodi, jolla pystytään määrittelemään sarakkeiden määrä kahteen ja fonttikoko 10. Koska sarakkeita määritellään kaksi, voidaan toiseen sarakkeeseen dokumenttinumeron lisäksi sijoittaa dokumentin nimitys. Myös erilaisten tekstikenttien fontit ja muut asetukset pystytään asettamaan erikseen VBA-koodissa, mikäli ohjelmoija katsoo sen tarpeelliseksi. Ensimmäiseen sarakkeeseen haetaan dokumenttinumero ja toiseen sarakkeeseen dokumentille asetettu nimi. Riveillä 7-10 suoritetaan varsinaisen luettelon täyttäminen. Täyttämisessä hyödynnetään for-toistosilmukkaa, jolla pystytään ajamaan tietty koodi monta kertaa läpi, kunnes jokin ennalta asetettu ehto täyttyy (For...Next statement, 2018). Jotta kaikki

dokumentit saadaan haettua, käydään läpi niistä jokainen eli siihen saakka, kunnes viimeinen dokumenttiluettelon rivinumero on saavutettu. Dokumenttien luetteloon viemisessä käytetään hyväksi ".List(x,x)"-toimintoa, jossa pystytään ensimmäisenä määrittämään haluttu rivi, johon arvo luetteloidaan, ja toisena sarake, johon arvo luetteloidaan. Riveillä 14-20 suoritetaan usean ohjelmakoodin lopussa oleva osa, jossa virheen havaittuaan ohjelma keskeyttää kyseisen toiminnon ja ilmoittaa siitä käyttäjälle.

```
1   'Specify multi column listbox
2   DocumentsListBox.ColumnCount = 2
3   DocumentsListBox.ColumnWidths = "100,249"
4   DocumentsListBox.Font.Size = 10
5
6   'Populate documents listbox
7   For i = 5 To LastDocumentRow
8       DocumentsListBox.AddItem
9       DocumentsListBox.List(i - 5, 0) = GetValue(Workbook,WorkSheet, i, DocumentColumn)
10      DocumentsListBox.List(i - 5, 1) = GetValue(Workbook,WorkSheet, i, DocumentTitle1Column)
11  Next i
12  End Sub
13
14      On Error GoTo catchError
15  exitSub:
16      Exit Sub
17  catchError:
18      '/ Give user error message
19      MsgBox "Unidentified error occured, please try again", vbButtonType, "Error"
20      GoTo exitSub
21
```

KUVA 25. Luettelon asetukset ja luettelon täyttäminen

Täyttölomakkeen alustuksen jälkeen täytyy suorittaa toiminto, jolla saadaan kaikissa tekstikentissä arvot vaihtumaan käyttäjän aktivoidessa jonkin dokumentin. Kuvassa 26 on esitetty ohjelmakoodi, jolla pystytään selvittämään valittu dokumentti luettelosta. Riviltä kaksi lähtien suoritetaan koodia, jossa ohjelma käy kaikki luetteloon haetut dokumentit läpi ja mikäli kyseinen dokumentti on valittu, muuttujaan "SelectedDocument" tallennetaan sen numero. Riveillä 10-25 on puolestaan esitetty toiminto, jossa dokumentin muuttuessa tyhjennetään jokainen tekstikenttä rivillä 14 esitetyllä tavalla. Sen jälkeen täytetään tekstikentät uudelleen rivillä 17 käytetyllä "GetValue"-funktiolla, jonka toiminta on esitetty rivillä 28. Ohjelmakoodin rivit 19-23 on tarkoitettu dokumentin tyypin valintaan, jossa kirjatun dokumenttikoodin mukaan haetaan sille erillinen nimitys "ProjectInfo"-välilehdelle tallennetuista nimityksistä. Myös dokumentin poistaminen tapahtuu samalla

tavalla kuin dokumentin tallennus. Dokumenttia poistettaessa ainoana erona on, että jokaiseen dokumentin tekstikenttään kirjoitetaan tilalle tyhjä arvo.

```
'Declare which document is selected in the listbox
For i = 1 To DocumentsListBox.ListCount 'Looping through listbox values
    If DocumentsListBox.Selected(i - 1) = True Then 'Check if item is selected in the list
        SelectedDocument = DocumentsListBox.List(i - 1, 0) 'Store selected document number to variable
        SelectedRow = i + 4
    End If
Next i

'When value from list box is changed this part of code will run
With Me
    'Check if any value is selected
    If DocumentsListBox.ListIndex >= 0 Then _
        'Clear all values in textboxes
        .DocumentNumberBox.Value = ""

        'Add values to textboxes
        .DocumentNumberBox.Value = GetValue(Workbook, Worksheet, SelectedRow, DocumentColumn)

        If Not IsError(Application.Match(Workbooks(Workbook).Worksheets(Worksheet).Cells(SelectedRow, DocumentTypeColumn).Value, Workbooks(Workbook).Worksheets("ProjectInfo").Range("L:L"),
        LastTypeRow = Workbooks(Workbook).Worksheets("ProjectInfo").Range("L:L").Find(What:=SelectedType, LookIn:=xlValues).row
        .TypeNameBox.Value = Workbooks(Workbook).Worksheets("ProjectInfo").Cells(LastTypeRow, 13).Value
        Else: .TypeNameBox.Value = "Type not found"
        End If
    End If
End With


GetValue = Workbooks(strWorkBook).Worksheets(strWorkSheet).Cells(strRow, strColumn).Value

```

KUVA 26. Lomakkeen dokumenttiarvojen täyttäminen

Jotta käyttäjä pystyy hakemaan dokumentteja luettelosta myös erikseen, täytyy lomakkeeseen ohjelmoida myös hakutoiminto. Kuvassa 27 on esitetty dokumentin hakua varten toteutettu ohjelmakoodi. Koodi suoritetaan joka kerta, kun lomakkeessa sijaitsevassa hakukentässä muutetaan arvoa. Jokaisella hakukentässä tehdyllä muutoksella saadaan kutsuttua tämä aliohjelma lisäämällä tekstikentän nimen perään "_Change".

```

'Search
Me.DocumentsListBox.Clear 'Clear documents list
If Len(TextBox5) = 0 <> "" Then 'Check if there's any text in search textbox
    If Worksheets(Worksheet).Range("B" & Worksheets(Worksheet).Rows.Count).End(xlUp).row > 1 And Trim(Me.TextBox5.Value) <> vbNullString Then
        arrList = Worksheets(Worksheet).Range("B5:B" & Worksheets(Worksheet).Range("B" & Worksheets(Worksheet).Rows.Count).End(xlUp).row).Value2
        arrList2 = Worksheets(Worksheet).Range("I5:I" & Worksheets(Worksheet).Range("I" & Worksheets(Worksheet).Rows.Count).End(xlUp).row).Value2
        For i = LBound(arrList) To UBound(arrList) 'Going through every item on array
            If InStr(1, arrList(i, 1), Trim(Me.TextBox5.Value), vbTextCompare) Then
                DocumentsListBox.AddItem arrList(i, 1)
                DocumentsListBox.List(x, 1) = arrList2(i, 1)
                x = x + 1
            End If
        Next i
    End If
Else
Call UserForm_Initialize 'If search textbox is empty retrieve all documents to list by initializing userform
End If
```

KUVA 27. Dokumentin hakutoiminto

Rivillä 12 suoritetaan koko dokumenttiluettelon tyhjennys, jolloin saadaan luettelo tyhjennettyä alkuperäisistä kohteista uusien hakuun täsmäävien kohteiden täyttämistä varten. Rivillä 13 tarkistetaan, onko kyseisessä hakukentässä yhtään tekstiä. Mikäli käyttäjä on syöttänyt merkkejä tekstikenttään, suoritetaan dokumenttien haku dokumenttinumeron perusteella. Jos puolestaan käyttäjä on esimerkiksi haun jälkeen tyhjentänyt koko tekstikentän, suoritetaan rivillä 26 lomakkeen alustaminen uudelleen. Tämä alustaa koko luettelon kaikilla alkuperäisillä arvoilla. Varsinainen hakutoiminto dokumenteille suoritetaan riveillä 14-23. Jotta voidaan tarkastaa, että dokumenttiluettelossa, jossa haku suoritetaan, on enemmän arvoja kuin yksi, suoritetaan rivin 14 ohjelmakoodi. Lisäksi rivillä 14 tarkistetaan, että puhdistettu hakukenttään syötetty arvo on jokin muu kuin nolla eli tyhjä arvo. Hakukentän arvon puhdistukseen hyödynnetään VBA:n trim-toimintoa. Tällä toiminnolla saadaan syötetystä tekstistä puhdistettua ylimääräiset välilyönnit tekstin alusta ja lopusta sekä eri sanojen välistä (LTrim, RTrim, and Trim functions 2018). Puhdistettua arvoa verrataan tämän jälkeen "vbNullString"-arvoon, joka tarkoittaa tyhjää string-tyyppistä arvoa. "vbNullString" on ohjelmakoodin suorittamisessa parempi vaihtoehto, kuin yleisesti käytetty ""-arvo. Tämä siksi, että "vbNullString" on suoraan arvo nolla, kun taas ""-arvo on tyhjä teksti. Tyhjä teksti vie ohjelmaa suoritettaessa enemmän tietokoneen RAM-muistia, kuin "vbNullString". Riveillä 15-16 haetaan array-tyyppisiin muuttujiin "arrList" ja "arrList2" dokumenttiluettelosta dokumentin numero ja kuvaus. Ohjelmakoodissa seuraavilla riveillä 18-21 saadaan näistä muuttujista tallennettua luetteloon uudet arvot käyttäen for-toistosilmukkaa.

5.4 Prosessisähköistys

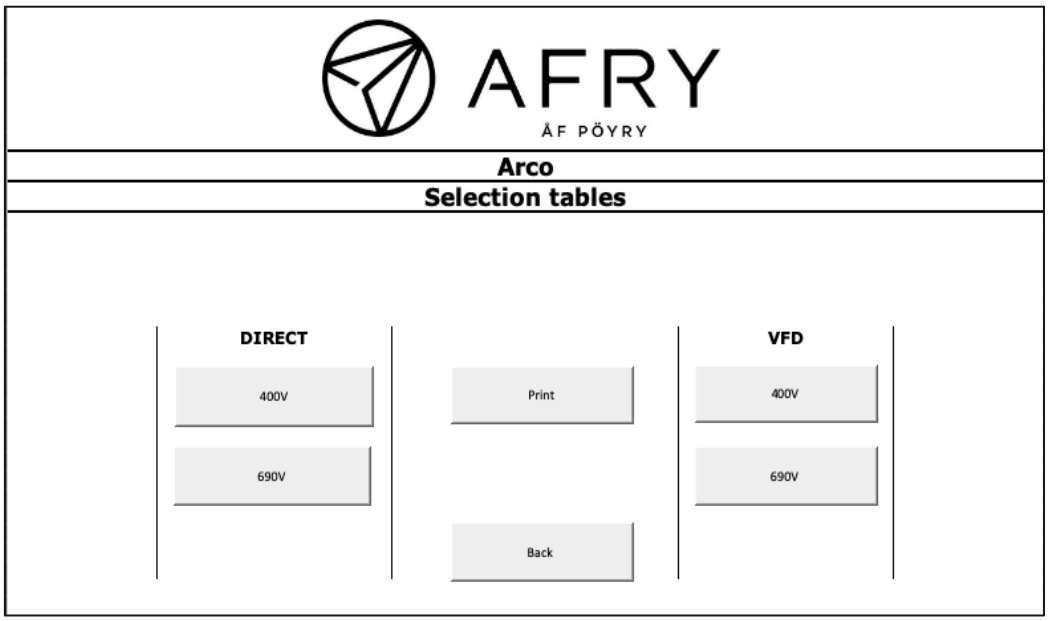
Opinnäytetyön osalta prosessisähköistyksessä keskitytään pääosin valintataulukoihin ja sen myötä komponenttien automaattiseen valintaan. Työkalussa on prosessisähköistystä varten rakennettu oma tietokantatiedosto, johon kerätään kaikki tarpeellinen informaatio prosessisähköistyksen sähköpiireistä. Kuvassa 28 on esitetty osa Excel-tiedostosta, johon tietokanta on rakennettu. Tiedostossa kootaan jokaiseen sarakkeeseen oma tietonsa. Yhdellä rivillä tiedostossa on vaihtoehtoisesti joko yksi kokonainen sähköpiiri tai yksi sähköpiirin signaali. Kuvassa 28 on esimerkiksi sarakkeeseen S sijoitettu turvakytkimen tunnuksen tieto. Tämä Excel-tiedoston välilehti sisältää kaiken mahdollisen tiedon piiristä. Käyttäjällä on myös mahdollisuus lisätä sarakkeita tarpeen mukaan, koska ohjelmakoodi on suunniteltu siten, että se ei ole suoraan riippuvainen absoluuttisesta sarakkeen osoitteesta, vaan se hakee tiedon nimen perusteella. Tiedostossa on myös muita välilehtiä kuten "MCC"-välilehti, joka sisältää tietoja sähkökeskuksen liitynnöistä ja komponenteista. Näiden välilehtien toiminnallisuudet on jätetty tämän opinnäytetyön sisällöstä pois.

L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
MOTOR							SAFETY SWITCH		FC		MCC				
m_man	m_type	m_v	m_kw	m_a	m_rpm		ss_nbr	ss_type	fc_nbr	fc_type		mcc_nbr	mcc_sec	mcc_s	mcc_f
MOTOR MANUFACTURER	MOTOR TYPE	VOLTAGE V	POWER Kw	CURRENT A	SPEED RPM	HOOK-UP	SAFETY SWITCH NBR	SAFETY SWITCH TYPE	FREQUENCY CONVERTER NBR	FREQUENCY CONVERTER TYPE	MCC TYPE	MCC NUMBER	MCC SECTION	SWITCH FUSE	FUSE
		400	7,5	14	2950		1000-M1-Q2	OT25E	1000-M1-U1	ACS880	INV	MCC1	-02-03	OESA	25gG
		400	5	10	1500		2000-M1-Q2	OT16			DIRECT	MCC1	-02-06	OESA	16aM
		400	5	10	1500		2000-M1-Q2	OT16			DIRECT	MCC1	-02-06	OESA	16aM
		400	5	10	1500		2000-M1-Q2	OT16			DIRECT	MCC1	-02-06	OESA	16aM
		400	5	10	1500		2000-M1-Q2	OT16			DIRECT	MCC1	-02-06	OESA	16aM
		400	10	20							SUPPLY	MCC1	-03-01	OESA	25gG

KUVA 28. Prosessisähköistyksen tietokantatiedosto

Yleinen prosessisähköistyksessä käytettävä dokumentti on moottoreiden ja niiden tarvitsemien komponenttien valintataulukko. Uudistetussa työkalun käyttöliittymässä on valintataulukkoja varten luotu oma välilehti, jossa käyttäjä pääsee luomaan ja tulostamaan tarpeelliset valintataulukot. Koska työkalu on pääsääntöisesti tarkoitettu pienempien projektien suunnitteluun, luotiin valintataulukkoon mahdollisuudet neljälle yleisimmälle tyypille. Yleisimmät tyypit prosessisähköistyksessä ovat 400 V jännitetason suorat- ja taajuusmuuttajalähdöt sekä 690 V

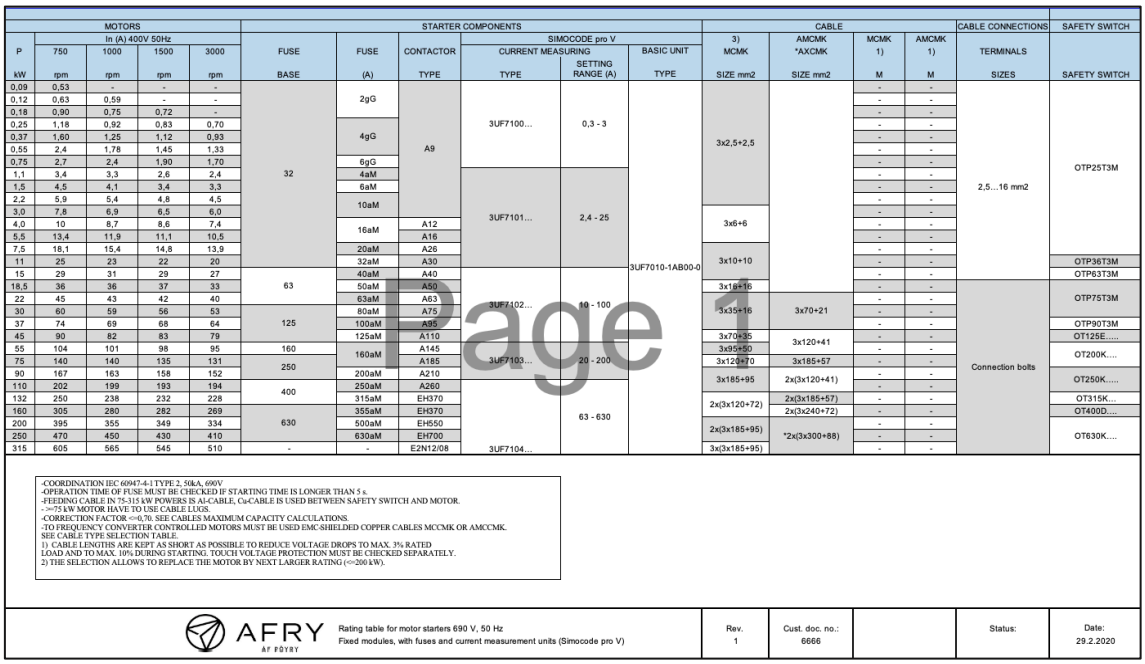
jännitetason suorat- ja taajuusmuuttajalähdöt. Kuvassa 29 on esitetty käyttöliittymän valintataulukko-välilehden yleisnäkymä. Käyttöliittymästä on mahdollisuus klikkaamalla siirtyä varsinaisen valintataulukko-tiedoston välilehdelle, jossa valittu valintataulukko sijaitsee. Nämä taulukot on koottu yhteiseen Excel-tiedostoon, jossa käyttäjällä on mahdollisuus muokata jokaista valintataulukkoa haluamansa muotoiseksi. Lisäksi käyttöliittymästä on mahdollisuus tulostaa valintataulukot yhdessä tai erikseen pdf-tiedostoformaattiin asiakasta varten.



KUVA 29. Valintataulukkojen välilehden yleisnäkymä

Kuvassa 30 on esitetty 400 V jännitetason suorakäynnisteisten moottorilähtöjen valintataulukko. Valintataulukoihin kirjataan kaikki tieto moottoreista, jotta pystytään valitsemaan moottorilähdölle komponentit. Käytännössä valintataulukko luodaan niin, että jokaiselle moottorin teholle on taulukon mukaan valittavissa komponentit. Automaattisessa komponentin valinnassa hyödynnetään valitun moottorin tehon sekä sille asetetun virran arvoja. Jos esimerkiksi moottorille on asetettu tehoksi 11 kW sekä virraksi alle 25 A, voidaan moottorille hakea arvot taulukosta riviltä, jossa on mainittu 11 kW teho. Tällöin kyseiselle moottorille valikoituisi esimerkin mukaan 32 A sulakepohja, jossa 32aM sulakkeet. Automaattinen valinta toteutetaan käyttäen funktiota, jota voidaan muualta koodista kutsua tarpeen tullen. Tämä antaa joustavuutta koodin toteutukselle myöhemmin, kun alkuperäisessä koodissa ei tarvitse ohjelmoida tai kopioida koodia joka kerta uudelleen, vaan se voidaan kutsua yksinkertaisesti. Tuloksena funktio palauttaa valinnan mukaan saadut arvot array-tyyppisessä muuttujassa, jossa jokainen alkio

vastaa valintataulukon saraketta. Näin ollen esimerkiksi array-muuttujan alkiossa 6 on sijoitettuna sulakkeen koko.



MOTORS					STARTER COMPONENTS						CABLE				CABLE CONNECTIONS	SAFETY SWITCH
P	In (A) 400V 50Hz 750	1000	1500	3000	FUSE	FUSE	CONTACTOR	SIMOCODE pro V CURRENT MEASURING TYPE	SETTING RANGE (A)	BASIC UNIT TYPE	3) MCMK	AMCMK *AXCMK	MCMK 1)	AMCMK 1)	TERMINALS	
kW	rpm	rpm	rpm	rpm	BASE	(A)	TYPE				SIZE mm2	SIZE mm2	M	M	SIZES	SAFETY SWITCH
0,09	0,53	-	-	-									-	-		
0,12	0,63	0,59	-	-		2gG							-	-		
0,18	0,90	0,75	0,72	-									-	-		
0,25	1,18	0,92	0,83	0,70		4gG		3UF7100...	0,3 - 3				-	-		
0,37	1,60	1,25	1,12	0,93			A9				3x2,5+2,5		-	-		
0,55	2,4	1,78	1,45	1,33									-	-		
0,75	2,7	2,4	1,90	1,70		6gG							-	-		OTP25T3M
1,1	3,4	3,3	2,6	2,4	32	4aM							-	-		
1,5	4,5	4,1	3,4	3,3		6aM							-	-		
2,2	5,9	5,4	4,8	4,5		10aM		3UF7101...	2,4 - 25				-	-	2,5...16 mm2	
3,0	7,8	6,9	6,5	6,0									-	-		
4,0	10	8,7	8,6	7,4		16aM	A12				3x6+6		-	-		
5,5	13,4	11,9	11,1	10,5			A16						-	-		
7,5	18,1	15,4	14,8	13,9		20aM	A26						-	-		
11	25	23	22	20		32aM	A30			3UF7010-1AB00-0	3x10+10		-	-		OTP36T3M
15	29	31	29	27		40aM	A40						-	-		OTP63T3M
18,5	36	36	37	33	63	50aM	A50				3x16+16		-	-		
22	45	43	42	40		63aM	A63	3UF7102...	10 - 100				-	-		OTP75T3M
30	60	59	56	53		80aM	A75				3x35+16	3x70+21	-	-		
37	74	69	68	64	125	100aM	A95						-	-		OTP90T3M
45	90	82	83	79		125aM	A110				3x70+35	3x120+41	-	-		OT125E.....
55	104	101	98	95	160	160aM	A145				3x95+50		-	-		OT200K....
75	140	140	135	131	250		A185	3UF7103...	20 - 200		3x120+70	3x185+57	-	-	Connection bolts	
90	167	163	158	152		200aM	A210				3x185+95	2x(3x120+41)	-	-		OT250K.....
110	202	199	193	194	400	250aM	A260					2x(3x185+57)	-	-		
132	250	238	232	228		315aM	EH370				2x(3x120+72)	2x(3x240+72)	-	-		OT315K...
160	305	280	282	269		355aM	EH370						-	-		OT400D....
200	395	355	349	334	630	500aM	EH550		63 - 630		2x(3x185+95)		-	-		
250	470	450	430	410		630aM	EH700					*2x(3x300+88)	-	-		OT630K....
315	605	565	545	510	-	-	E2N12/08	3UF7104...			3x(3x185+95)		-	-		

-COORDINATION IEC 60947-4-1 TYPE 2, 50kA, 690V
-OPERATION TIME OF FUSE MUST BE CHECKED IF STARTING TIME IS LONGER THAN 5 s.
-FEEDING CABLE IN 75-315 kW POWERS IS Al-CABLE, Cu-CABLE IS USED BETWEEN SAFETY SWITCH AND MOTOR.
- >=75 kW MOTOR HAVE TO USE CABLE LUGS.
-CORRECTION FACTOR <=0,70. SEE CABLES MAXIMUM CAPACITY CALCULATIONS.
-TO FREQUENCY CONVERTER CONTROLLED MOTORS MUST BE USED EMC-SHIELDED COPPER CABLES MCCMK OR AMCCMK.
SEE CABLE TYPE SELECTION TABLE.
1) CABLE LENGTHS ARE KEPT AS SHORT AS POSSIBLE TO REDUCE VOLTAGE DROPS TO MAX. 3% RATED
LOAD AND TO MAX. 10% DURING STARTING. TOUCH VOLTAGE PROTECTION MUST BE CHECKED SEPARATELY.
2) THE SELECTION ALLOWS TO REPLACE THE MOTOR BY NEXT LARGER RATING (<=200 kW).

AFRY AF PÖYRY — Rating table for motor starters 690 V, 50 Hz
Fixed modules, with fuses and current measurement units (Simocode pro V)

Rev.	Cust. doc. no.:		Status:	Date:
1	6666			29.2.2020

KUVA 30. 400 V jännitetason moottorilähtöjen valintataulukko

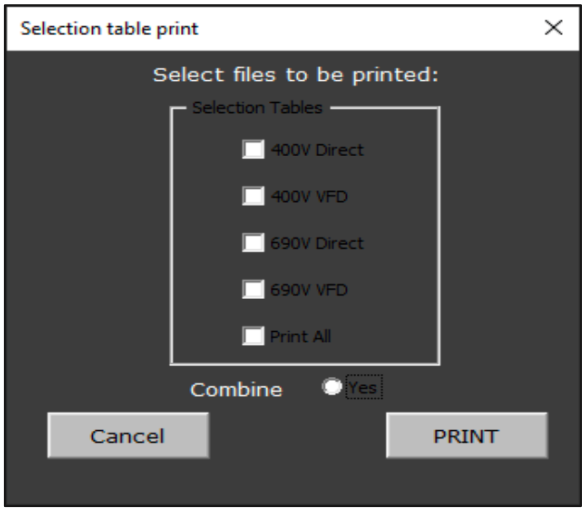
Kuvassa 31 on esitetty ohjelmakoodi automaattiselle valinnalle. Ohjelmakoodissa koodin alussa määritellään muuttujat jokaiselle valintataulukon sarakkeelle. Nämä muuttujat määritellään long-tyyppisiksi, jotta niihin pystytään tallentamaan sarakkeen numero, jossa tieto sijaitsee. Lisäksi täytyy määritellä muuttuja "vaArray", johon tallennetaan array-tyyppisenä kaikki tiedot palautettavaksi alkuperäiseen koodiin, josta funktiota kutsutaan. Riveillä 12-13 on esitetty esimerkit, joissa haetaan muuttujaan sen sarakkeen numero, jossa kyseinen tieto sijaitsee. Tätä pystytään myöhemmin käyttämään hyödyksi tiedon hakemisessa "vaArray"-muuttujaan. Rivillä 16 haetaan "lRow"-muuttujaan sen rivin numero, jossa haettavan moottorin teho sijaitsee. Jatkossa tällä rivinumerolla saadaan haettua kyseiselle moottorin teholle tiedot sen osoittamalta riviltä. Jotta pystytään tarkistamaan, että moottorille asetettu virta-arvo vastaa sille asetettua tehoa, suoritetaan riveillä 19-24 tarkistus. Tarkistuksessa varmistetaan jokaisesta pyörimisnopeuden sarakkeesta, että arvo sarakkeessa on suurempi kuin asetettu virta-arvo. Jos virta-arvo on pienempi kuin yksikään eri kierroslukumäärille asetetuista arvoista, voidaan jatkaa alkuperäisellä rivinumerolla. Jos puolestaan moottorille asetettu virta-arvo on suurempi kuin jokin pyörimisnopeussarakkeen arvoista, lisätään al-

kuperäiseen rivinumeroon yksi. Tällöin saadaan valittua komponentit mitoituksel-taan seuraavalle moottorikoolle. Riveillä 27 ja 28 sijoitetaan saadut arvot array-muuttujaan "GetValue"-funktiolla, joka palauttaa käyttäjälle arvon rivi- ja sarake-numeron perusteella string-tyyppisenä. Viimeiseksi rivillä 31 sijoitetaan arvot siir-rettäväksi alkuperäiseen koodiin.

```
2   Public Function AutomaticSelection(ByVal strWorkbook As String, ByVal strWorkSheet As String, ByVal strMotorPower As String, ByVal strMotorCurrent As Long) As Variant()
3
4   'Declare  Variables
5   Dim vaArray As Variant
6   Dim lRow As Long, lColumn As Long
7   Dim lPowerColumn As Long, l750rpmColumn As Long, l1000rpmColumn As Long, l1500rpmColumn As Long, l3000rpmColumn As Long, lFuseBaseColumn As Long, _
8   lFuseColumn As Long, lContactorTypeColumn As Long, lCurrentTypeColumn As Long, lCurrentRangeColumn As Long, lBUColumn As Long, lCable1Column As Long, _
9   lCable2Column As Long, lSafetySwitchColumn As Long
10
11  'Assign column values to variables
12  lPowerColumn = GetLastColumn(strWorkbook, strWorkSheet, 4, "kW")
13  l750rpmColumn = GetLastColumn(strWorkbook, strWorkSheet, 4, "rpm (750)")
14
15  'Assing selected power row number to variable
16  lRow = GetLastRow(strWorkbook, strWorkSheet, lPowerColumn, strMotorPower)
17
18  'Check if current is right for selected motor power and if it exceeds current value then get values from next motor size
19  If Cells(lRow, l750rpmColumn).Value > strMotorCurrent Or Cells(lRow, l1000rpmColumn).Value > strMotorCurrent Or _
20  Cells(lRow, l1500rpmColumn).Value > strMotorCurrent Or Cells(lRow, l3000rpmColumn).Value Then
21      lRow = lRow
22  Else
23      lRow = lRow + 1
24  End If
25
26  'Save values to array
27  vaArray(0) = GetValue(strWorkbook, strWorkSheet, lRow, lPowerColumn)
28  vaArray(1) = GetValue(strWorkbook, strWorkSheet, lRow, l750rpmColumn)
29
30  'Place motor selectection values to array
31  AutomaticSelection = vaArray
32
33  End Function
```

KUVA 31. Automaattisen valinnan ohjelmakoodi

Valintataulukoiden tulostamista varten hyödynnettiin täyttölomaketta. Kuvassa 32 on esitetty tätä varten laadittu lomake, jossa käyttäjä pystyy valitsemaan ha-luamansa taulukot sekä halutessaan yhdistää ne yhdeksi tiedostoksi.



KUVA 32. Valintataulukkojen tulostuksen täyttölomake

Kuvan 33 ohjelmakoodissa riveillä 91-93 tarkastetaan, onko käyttäjä valinnut valintataulukoiden yhdistämisen yhdeksi tiedostoksi. Mikäli valinta on tehty, voidaan julkiseksi asetettuun muuttujaan "PublicSelectionTableCombine" tallentaa boolean-tyyppisenä arvo tosi. Riveillä 101-115 käydään läpi samalla tavalla muut valinnat täyttölomakkeessa ja tallennetaan array-tyyppisen muuttujan "PublicSelectionTableValue" alkioihin rivien 96-100 mukaan arvot. Arvot tallennetaan julkiseksi asetettuihin muuttujiin, koska varsinainen valintataulukon tulostamisen ohjelmakoodi on eri osassa ohjelmaa. Tällöin muuttujat ja niiden arvot eivät muuten välittyisi eteenpäin.

```
86    '=============================================================================
87    Public Sub PrintSelectionTable_Click()
88    '=============================================================================
89
90    'Declare if user wants pdf's to be combined and store that value to public boolean
91    If CombineYes = True Then
92        PublicSelectionTableCombine = True
93    End If
94
95    'Declare what user wants to be printed and store value to public integer
96    'Value 0= 400V Direct
97    'Value 1= 400V VFD
98    'Value 2= 690V Direct
99    'Value 3= 690V VFD
100   'Value 4= Print all
101   If (CheckBox400VDirect.Value = True) = True Then
102       PublicSelectionTableValue(0) = 0
103   End If
104   If (CheckBox400VVFD.Value = True) = True Then
105       PublicSelectionTableValue(1) = 1
106   End If
107   If (CheckBox690VDirect = True) = True Then
108       PublicSelectionTableValue(2) = 2
109   End If
110   If (CheckBox690VVFD = True) = True Then
111       PublicSelectionTableValue(3) = 3
112   End If
113   If (CheckBoxSelectAll = True) = True Then
114       PublicSelectionTableValue(4) = 4
115   End If
116
117   Unload Me
118
119   End Sub
120   '------------ENDSUB-----------------
```

KUVA 33. Ohjelmakoodi valintataulukkojen arvojen tallentamiseen

Valintataulukkojen tulostamista varten työkaluun on ohjelmoitu oma moduuli. Kuvassa 34 on esitetty moduulin ohjelmakoodista sen pääosat. Kuvan riveillä 10-19 on ensimmäisenä määritelty laskureiden muuttujat sekä niiden nollaus. Laskureita käytetään ohjelmakoodissa myöhemmin hyväksi tulostusmäärien laskennassa ja taulukoiden yhdistyksessä. Koodissa riveillä 22 ja 23 puhdistetaan julkinen muuttuja "PublicSelectionTableValue" ja alustetaan sen arvo epätodeksi sen varalta, että muuttujaan on jäänyt aluksi jokin muu arvo. Tämän jälkeen rivillä 26 kutsutaan varsinainen täyttölomake, josta haetaan arvot aikaisemmin mainitun mukaisesti array-tyyppiseen muuttujaan "PublicSelectionTableValue". Riveillä 29-32 tallennetaan muuttujaan "PDFFileName" lopullinen käyttäjän valitsema tiedostonimi sekä sen polku. Käyttäjälle saadaan näkyviin resurssienhallintaikkuna

kehotteella "Application.GetSaveAsFileName". Tälle kehotteelle asetetaan oletusarvo tiedostonimestä sekä suodatus mahdollisista tiedostoformaateista. Rivillä 36 tarkastetaan, että käyttäjä on varmasti valinnut tiedostonimen sekä polun tallentamista varten. Tämän jälkeen riveillä 39-41 määritellään Excel-tiedosto, johon valintataulukkojen arvot on tallennettu sekä avataan kyseinen tiedosto tietojen keruuta varten. Jotta ohjelmakoodi saadaan suoritettua nopeasti ja tehokkaasti, suoritetaan riveillä 43 ja 189 koodi, jossa näytönpäivitys kytketään pois päältä koodin suorittamisen ajaksi. Riviltä 46 alkaen aloitetaan suorittaa ohjelmaa, mikäli käyttäjä ei ole valinnut tiedostojen yhdistämistä. Jos käyttäjä puolestaan on valinnut tiedostojen yhdistämisen, suoritetaan ohjelma alkaen riviltä 115.

```
10   Dim PDFFileName As Variant
11   Dim iCounter As Integer, cCounter As Integer 'Set CombineCounter and integerCounter as Integers
12   Dim pCounter As Integer, rCounter As Integer 'Set pageCounter and randomCounter as Integer
13   Dim EndFolder As String, CurrentFolder As String 'Set Folder variables as a String
14
15   'Format Counter Values
16   iCounter = 0
17   cCounter = 0
18   rCounter = 0
19   pCounter = 0
20
21   'Erase public array and integer
22   Erase PublicSelectionTableValue
23   PublicSelectionTableCombine = False
24
25   'Show Selection Table Form Userform for user
26   SelectionTableForm.Show
27
28   'Ask folder and file name from user
29   PDFFileName = Application.GetSaveAsFilename( _
30   InitialFileName:="Selection Table", _
31   FileFilter:="PDF, *.pdf", _
32   Title:="Save As PDF")
33
34   'Declare what user wants to be printed and store value to public integer
35   'Value 0= 400V Direct 'Value 1= 400V VFD 'Value 2= 690V Direct 'Value 3= 690V VFD 'Value 4= Print all
36   If PDFFileName <> False Then
37
38       'Open Selection Table excel for printing
39       EndFolder = "\SelectionTables\SelectionTables.xlsm"
40       CurrentFolder = (Application.ThisWorkbook.Path) 'Find out where this current main file is located to find end file
41       Workbooks.Open Filename:=CurrentFolder & EndFolder, UpdateLinks:=1
42
43       Application.ScreenUpdating = False 'Set Screen Updating to false to help code run
44
45       'If User doesn't want to combine then run this part of the code
46   >   If PublicSelectionTableCombine = False Then⋯
114      'If User wants to combine PDF's then this part will be run
115  >   ElseIf PublicSelectionTableCombine = True Then⋯
187      End If
188
189      Application.ScreenUpdating = True 'Set Screen Updating to true as code has been terminated
190      MsgBox "PDF's Printed succesfully"
191
192  End If
```

KUVA 34. Valintataulukko tulostuksen ohjelmakoodin pääosat

Mikäli käyttäjä ei ole halunnut yhdistää tiedostoja ja on valinnut tulostettavaksi kaikki valintataulukot, on siihen käytetty ohjelmakoodi esitetty tarkemmin kuvassa 35. Rivillä 48 tarkastetaan täyttölomakkeessa aiemmin asetetun muuttujan "PublicSelectionTableValue" arvo. Mikäli arvo on neljä eli kaikkien taulukoiden tulostusta vastaava arvo, edetään ohjelmassa riviltä 49 alkaen. Rivillä 49 määritellään

"iCounter"-laskurin avulla for-toistosilmukkaa, joka suoritetaan, kunnes kaikki valintataulukon välilehdet on käyty läpi. Jotta jokaiselle välilehdelle saadaan asetettua sopivat asetukset tulostamista varten, käytetään rivillä 50 with-komentoa. Tällä komennolla pystytään suorittamaan yhdelle objektille eri komentoja kirjoittamatta itse objektia joka kerta uudelleen (With...End With Statement (Visual Basic), 2015). Tämä hyödyttää erityisesti, kun komentoja suoritetaan for-toistosilmukkaa sisällä. Tässä tapauksessa tulostusta varten määritellään välilehdelle asetukset, joissa saadaan jokaiselle sivulle yläotsikko, sivun suunta vaakatasoon, sivun suurennus pois päältä ja paperinkoko A4-kokoon. Lisäksi osa komennoista on kommentoitu pois myöhempää käyttöä varten. Riviltä 59 alkaen suoritetaan varsinainen tulostus pdf-tiedostoiksi. Tätä varten käytetään "ExportAsFixedFormat"-komentoa, jossa määritellään tiedostotyyppi, tiedostonimi ja muita tarvittavia asetuksia. Näiden asetusten mukaan komennon avulla saadaan vietyä Excel-tiedosto muihin tiedostomuotoihin (Workbook.ExportAsFixedFormat method (Excel), 2019). Lopulta käyttöliittymä avaa käyttäjälle lopputuloksen järjestelmän oletukseksi asetetulla ohjelmistolla. Viimeiseksi valintataulukoiden Excel-tiedosto suljetaan rivillä 70 tallentamatta itse tiedostoa.

```
45      'If User doesn't want to combine then run this part of the code
46      If PublicSelectionTableCombine = False Then
47          'If User has selected "Print All" -value without combine then run this part of the code
48          If PublicSelectionTableValue(4) = 4 Then
49              For iCounter = 1 To Worksheets.Count
50                  With Worksheets(iCounter).PageSetup
51                      .CenterHeader = "Selection Table"
52                      .Orientation = xlLandscape
53                      '.PrintArea = Here we can set finding for last column and row if needed
54                      .Zoom = False
55                      .PaperSize = xlPaperA4
56                      '.FitPagesToTall = False
57                      '.FitPagesToWide = 1
58                  End With
59                      Worksheets(iCounter).ExportAsFixedFormat _
60                      Type:=xlTypePDF, _
61                      Filename:=PDFFileName & "_" & iCounter, _
62                      Quality:=xlQualityStandard, _
63                      IncludeDocProperties:=False, _
64                      IgnorePrintAreas:=False, _
65                      OpenAfterPublish:=True
66                      'From:= Here can be set from page to start printing if needed
67                      'To:= End page to end printing
68              Next
69          'Close Selection Table excel after printing is done
70          ActiveWorkbook.Close SaveChanges:=False
```

KUVA 35. Kaikkien valintataulukoiden tulostamisen ohjelmakoodi

Mikäli käyttäjä haluaa suorittaa tulostuksen yksittäisistä valintataulukoista, on siihen käytetty ohjelmakoodi esitetty kuvassa 36. Rivillä 72 tarkastetaan, että täyttölomakkeessa ei ole asetettu kaikkien valintataulukoiden tulostamista. Tämän jälkeen muuttujaan "pCounter" asetettua laskuria hyödyntäen, voidaan käydä läpi jokainen sivu yksitellen. Asetukset tulostamista varten asetellaan samalla tavalla, kuin kaikkien valintataulukoiden tulostamisen tapauksessakin. Varsinaisessa pdf-tiedostoksi viemisessä hyödynnetään "From"- ja "To"-komentoja, joilla saadaan määriteltyä ensimmäinen välilehti, josta tulostus aloitetaan sekä viimeinen sivu, johon tulostus päätetään. Molemmat sivut asetellaan tässä tapauksessa samaksi, koska tiedostojen ei haluta yhdistyvän yhdeksi samaksi tiedostoksi.

```
71          'If user has selected any other value than "Print all" without combine then this code will be executed
72          ElseIf PublicSelectionTableValue(4) <> 4 Then
73              For pCounter = 0 To 3
74                  If PublicSelectionTableValue(pCounter) = pCounter Then
75                      With Worksheets(pCounter + 1).PageSetup
76                          .CenterHeader = "Selection Table"
77                          .Orientation = xlLandscape
78                          '.PrintArea = Here we can set finding for last column and row if needed
79                          .Zoom = False
80                          .PaperSize = xlPaperA4
81                          '.FitPagesToTall = False
82                          '.FitPagesToWide = 1
83                      End With
84                      ActiveWorkbook.ExportAsFixedFormat _
85                      Type:=xlTypePDF, _
86                      Filename:=PDFFileName & "_" & pCounter, _
87                      Quality:=xlQualityStandard, _
88                      IncludeDocProperties:=False, _
89                      IgnorePrintAreas:=False, _
90                      OpenAfterPublish:=True, _
91                      From:=pCounter + 1, _
92                      To:=pCounter + 1
93                  End If
94              Next
95          'Close Selection Table excel after printing is done
96          ActiveWorkbook.Close SaveChanges:=False
97          End If
```

KUVA 36. Yksittäisten valintataulukon tulostamisen ohjelmakoodi

Jos käyttäjä on valinnut täyttölomakkeessa tiedostojen yhdistämisen yhdeksi pdf-tiedostoksi, suoritetaan kuvassa 37 esitetty ohjelmakoodi. Ohjelmakoodi pystytään suunnittelemaan hyvin samalla tavalla, kuin tiedostojen erikseen tulostuksessakin. Mikäli käyttäjä on valinnut tulostettavaksi kaikki valintataulukot, suoritetaan koodi riviltä 101 alkaen. Tässä koodissa ainoana erona aikaisempaan on, että ei käytetä for-toistosilmukkaa sivujen erittelyyn. Tällöin ohjelma osaa automaattisesti tulostaa koko Excel-tiedoston välilehdet ja koota ne yhdeksi tiedostoksi. Jos halutaan yhdistää erikseen eri valintataulukkoja, voidaan suorittaa koodi riviltä 122 alkaen. Tässä koodissa hyödynnetään "cCounter"-muuttujaan

asetettua laskuria. Laskurin avulla for-toistosilmukkaa hyödyntäen järjestellään valitut valintataulukot Excel-tiedoston ensimmäisiksi välilehdiksi. Lisäksi "rCounter"-muuttujan avulla lasketaan, kuinka monta valintataulukkoa käyttäjä on halunnut tulostaa. Lopulta riveillä 145 ja 146 pystytään tätä muuttujaa hyödyntäen määrittelemään tulostettavat valintataulukot. Tulostuksen jälkeen välilehdet Excel-tiedostossa on järjestettävä uudelleen alkuperäiseen järjestykseen rivien 148-151 avulla.

```
99      'If User wants to combine PDF's then this part will be run
100     ElseIf PublicSelectionTableCombine = True Then
101         If PublicSelectionTableValue(4) = 4 Then
102             With ActiveSheet.PageSetup
103                 .CenterHeader = "Selection Table"
104                 .Orientation = xlLandscape
105                 '.PrintArea = Here we can set finding for last column and row if needed
106                 .Zoom = False
107                 .PaperSize = xlPaperA4
108                 '.FitPagesToTall = False
109                 '.FitPagesToWide = 1
110             End With
111                 ActiveWorkbook.ExportAsFixedFormat _
112                 Type:=xlTypePDF, _
113                 Filename:=PDFFileName, _
114                 Quality:=xlQualityStandard, _
115                 IncludeDocProperties:=False, _
116                 IgnorePrintAreas:=False, _
117                 OpenAfterPublish:=True
118                 'From:= Here can be set from page to start printing if needed
119                 'To:= End page to end printing
120         'Close Selection Table excel after printing is done
121         ActiveWorkbook.Close SaveChanges:=False
122         ElseIf PublicSelectionTableValue(4) <> 4 Then
123             For cCounter = 0 To 3
124                 If PublicSelectionTableValue(cCounter) = cCounter Then
125                     Sheets(cCounter + 1).Move Before:=Sheets(1)
126                     rCounter = rCounter + 1 'Add value to rCounter to see how many pages needs to be printed
127                 End If
128             Next
129             With ActiveSheet.PageSetup
130                 .CenterHeader = "Selection Table"
131                 .Orientation = xlLandscape
132                 '.PrintArea = Here we can set finding for last column and row if needed
133                 .Zoom = False
134                 .PaperSize = xlPaperA4
135                 '.FitPagesToTall = False
136                 '.FitPagesToWide = 1
137             End With
138                 ActiveWorkbook.ExportAsFixedFormat _
139                 Type:=xlTypePDF, _
140                 Filename:=PDFFileName, _
141                 Quality:=xlQualityStandard, _
142                 IncludeDocProperties:=False, _
143                 IgnorePrintAreas:=False, _
144                 OpenAfterPublish:=True, _
145                 From:=1, _
146                 To:=rCounter - 1
147         'Replace sheets
148         Sheets("400V_Direct").Move Before:=Worksheets(1)
149         Sheets("400V_VFD").Move After:=Worksheets("400V_Direct")
150         Sheets("690V_Direct").Move After:=Worksheets("400V_VFD")
151         Sheets("690V_VFD").Move After:=Worksheets("690V_Direct")
152
153         'Close Selection Table excel after printing is done
154         ActiveWorkbook.Close SaveChanges:=True
155         End If
```

KUVA 37. Valintataulukkojen tulostus yhdistäessä tiedostot

Valintataulukkojen tulostuksen avulla käyttäjä pystyy helpottamaan työtään etenkin, kun valintataulukot halutaan tulostaa asiakasta varten. Tällöin käyttäjän ei tarvitse erikseen käyttää kolmannen osapuolen sovelluksia tiedostojen yhdistämiseen. Lisäksi aikaa säästyy, kun samalla työkalulla saadaan nopeasti tiedostot yhdessä tai erikseen.

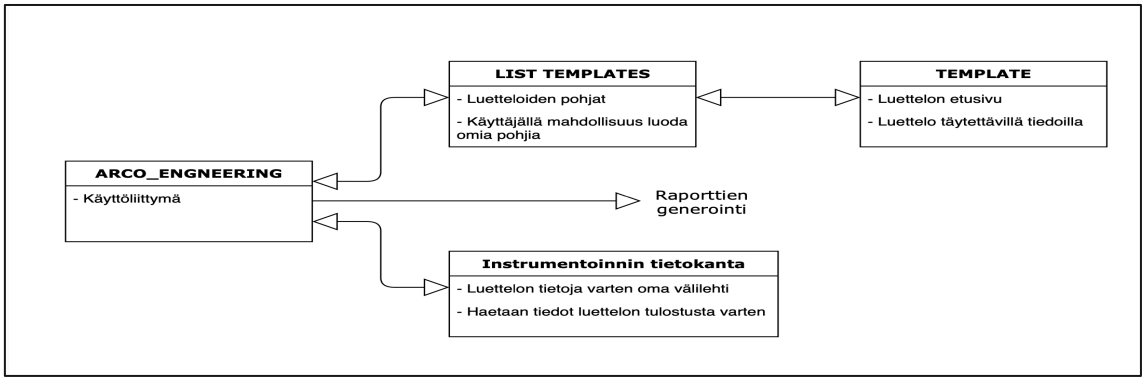
5.5 Instrumentointi

Uudistetussa työkalussa instrumentoinnin osalta keskitytään erityisesti erilaisten luetteloiden generoimiseen. Kuten prosessisähköistystäkin varten, myös instrumentointia varten on työkaluun rakennettu oma tietokantatiedosto. Instrumentoinnissa jokaisella tietokannan rivillä on esitetty oma instrumentoinnin signaali. Tähän tiedostoon on myös lisätty omia välilehtiä eri luetteloita varten. Opinnäytetyön osalta keskitytään erityisesti "CABLES"-välilehteen, johon on kaapeliluettelon tulostamista varten kerätty kaikki tarpeellinen tieto. Kuvassa 38 on esitetty näyttöleike tästä välilehdestä. Välilehdelle käyttäjä pystyy lisäämään eri tietoja omiin sarakkeisiin ja halutessaan käyttämään myös taulukkolaskentaohjelmiston funktioita.

	A	B	C	D	E	F	G	H	I	J	K
1	CABLE NUMBER	CABLE TYPE	FROM	FROM DESCRIPTION	TO	TO DESCRIPTION	SUPPLIER	INSTALLER	LENGHT (ESTIMATED)	CLASS	CABLE NOTE
2	xxxx-W001	MCMK 2x2,5+2,5	XXXX	XXXX	XXXX	XXXX	ES	ES	105	P	-
3											

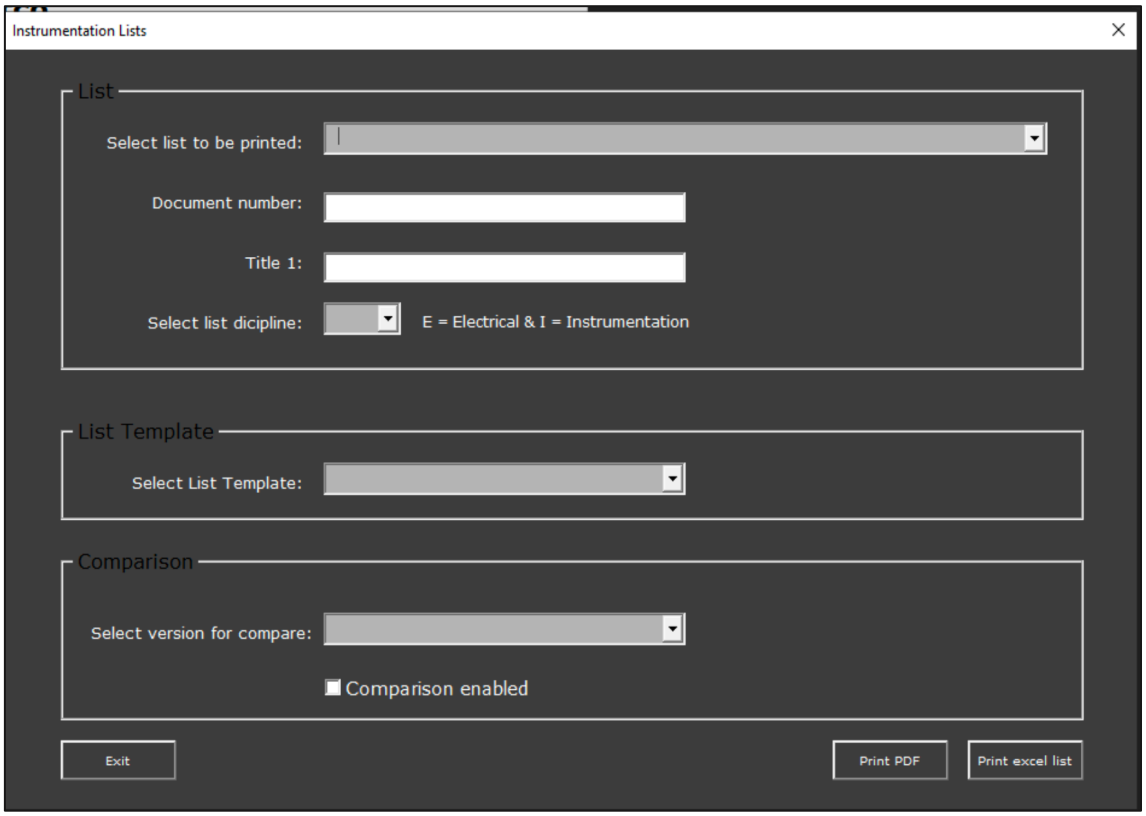
KUVA 38. Instrumentointikaapeleiden välilehti

Luetteloiden generoimisen toteutuksesta on esitetty kuvassa 39 yksinkertaistettu kaavio. Kaaviossa on esitetty, kuinka varsinainen luetteloiden data haetaan instrumentoinnin tietokantatiedostosta. Jotta myös käyttäjällä on mahdollisuus luoda omia pohjia luetteloille, haetaan nämä pohjat erillisestä kansiostosta. Jokaiseen luettelon pohjaan käyttäjän tulee luoda etusivu, johon täytetään yleiset projektin tiedot sekä selite tuotettavasta dokumentista. Lisäksi pohjaan tulee luoda varsinainen luettelo-välilehti, johon data tallennetaan.



KUVA 39. Luetteloiden generoimisen kaavio

Dokumenttienhallinnan sekä projektin yleisten tietojen täyttämisen tapaan, myös luetteloiden generoimiselle on kehitetty täyttölomake. Generoimiseen käytetty täyttölomake on esitetty kuvassa 40. Se muodostuu kolmesta eri osasta, jotka ovat luettelo, luettelon pohja ja vertailu. Täyttölomake on suunniteltu siten, että sitä on mahdollista käyttää myös prosessisähköistyksen puolella. Luettelo-osuudessa käyttäjä valitsee ensimmäisenä suunnittelualueen, jonka mukaan ohjelmakoodi suodattaa dokumenttiluettelosta kyseisen alueen dokumentit pudotusvalikkoon. Seuraavaksi ohjelma pyytää pohjaa, josta varsinainen luettelo luodaan. Käyttäjällä on myös tarvittaessa mahdollisuus suorittaa vertailu aikaisempaan luetteloon. Ohjelma hakee automaattisesti valmiiden luetteloiden kansiosta mahdolliset vertailtavat luettelot ja vertailee uutta luetteloa niihin. Vertailussa työkalu värjää muuttuneet solut tunnusväreillä sekä yliviivaa poistuneet rivit. Vertailua ei tämän työn aikana saatu toteutettua kokonaan.



KUVA 40. Luetteloiden generoimisen täyttölomake

Luetteloiden generoimiseen käytetyn täyttölomakkeen alustuksen ohjelmakoodi on esitetty osittain kuvassa 41. Riveillä 10-14 haetaan julkisiin muuttujiin dokumenttiluettelosta viimeinen rivi sekä eri tietoja sisältävien sarakkeiden numerot. Numeroiden haku erikseen mahdollistaa varsinaisen dokumenttiluettelon muokkaamisen, kun tietoa ei haeta absoluuttisen osoitteen avulla, vaan tiettyä nimeä hakemalla. Lopuksi tallennetaan suunnittelualueen pudotusvalikon valinta muuttujaan "strDicipline".

```
6   ArcoWorkbook = "Arco_ENGINEERING.xlsm" 'Specify main excel name to variable
7   ArcoDocumentsSheet = "Documents" 'Specify main excel document sheet to variable
8
9   'Pick up last used document row and document number column
10  lLastDocumentRow = Workbooks(ArcoWorkbook).Sheets(ArcoDocumentsSheet).Cells(Rows.Count, 2).End(xlUp).row
11  lDocumentNumberColumn = Workbooks(ArcoWorkbook).Worksheets(ArcoDocumentsSheet).Range("4:4").Find(What:="Document number", LookIn:=xlValues).column
12  lDocumentTitleColumn = Workbooks(ArcoWorkbook).Worksheets(ArcoDocumentsSheet).Range("4:4").Find(What:="Title 1", LookIn:=xlValues).column
13  lDocumentDiciplineColumn = Workbooks(ArcoWorkbook).Worksheets(ArcoDocumentsSheet).Range("4:4").Find(What:="Dicipline", LookIn:=xlValues).column
14  lDocumentTypeColumn = Workbooks(ArcoWorkbook).Worksheets(ArcoDocumentsSheet).Range("4:4").Find(What:="Type", LookIn:=xlValues).column
15
16  'Get Dicipline value
17  strDicipline = DiciplineComboBox.Value
18
```

KUVA 41. Luetteloiden alustuksen ohjelmakoodi

Kuvassa 42 on esitetty loppuosa alustuksen ohjelmakoodista. Riveillä 20-22 asetellaan täyttölomakkeen luettelot sisältävän pudotusvalikon asetukset. Niillä määritellään sarakkeiden määrä kahteen, koska toiseen sarakkeeseen sijoitetaan dokumentin numero ja toiseen dokumentin nimitys. Lisäksi määritellään fonttikoko sekä sarakkeiden leveydet, jotta pudotusvalikosta saadaan käyttäjälle miellyttävän näköinen. Ennen varsinaista pudotusvalikon täyttämistä tyhjennetään se mahdollisten vanhojen dokumenttien varalta rivillä 24. Riviltä 27 alkaen käydään for-toistosilmukkaa käyttäen jokainen dokumenttiluettelon dokumentti läpi ja lisätään jokainen ehdon täyttänyt dokumentti. Rivillä 28 tarkastetaan, että dokumenttiluettelossa asetettu suunnittelualue vastaa täyttölomakkeessa asetettua arvoa. Rivillä 29 käydään läpi, että dokumentille on asetettu dokumenttityyppi. Dokumenttityyppi on pakollinen tieto, jotta luettelo osataan tallentaa oikeaan kansioon. Varsinainen pudotusvalikon täyttäminen suorittaan riveillä 30-31. Lopuksi riveillä 39-50 tallennetaan oletusarvot pudotusvalikkoihin, jotta käynnistäessä täyttölomake, käyttäjällä olisi valinnat valmiina.

```
19  'Settings for list combobox
20  ComboBoxInstLists.ColumnCount = 2
21  ComboBoxInstLists.Font.Size = 9
22  ComboBoxInstLists.ColumnWidths = "100,450"
23
24  ComboBoxInstLists.Clear
25
26  'Populate list combobox
27  For i = 5 To lLastDocumentRow
28      If Workbooks(ArcoWorkbook).Sheets(ArcoDocumentsSheet).Cells(i, lDocumentDiciplineColumn).Value = strDicipline Then
29          If Workbooks(ArcoWorkbook).Sheets(ArcoDocumentsSheet).Cells(i, lDocumentTypeColumn).Value <> "" Then
30              ComboBoxInstLists.AddItem
31              ComboBoxInstLists.List(i - 5, 0) = Workbooks(ArcoWorkbook).Sheets(ArcoDocumentsSheet).Cells(i, lDocumentNumberColumn).Value
32              ComboBoxInstLists.List(i - 5, 1) = Workbooks(ArcoWorkbook).Sheets(ArcoDocumentsSheet).Cells(i, lDocumentTitleColumn).Value
33          Else
34              MsgBox "One or more document doesn't have document type assigned"
35          End If
36      End If
37  Next i
38
39  If ListTemplateComboBox.ListCount = 0 Then
40  ListTemplateComboBox.AddItem "Select"
41  End If
42
43  If ComparisonFileCombo.ListCount = 0 Then
44  ComparisonFileCombo.AddItem "Select"
45  End If
46
47  If DiciplineComboBox.ListCount = 0 Then
48  DiciplineComboBox.AddItem "E"
49  DiciplineComboBox.AddItem "I"
50  End If
```

KUVA 42. Luetteloiden alustuksen ohjelmakoodin loppuosa

Käyttäjän vaihtaessa suunnittelualaa pudotusvalikosta, suoritetaan joka kerta täyttölomakkeen alustus. Tällöin saadaan kaikki dokumentit luetteloitua näkyviin. Kun jokin dokumentti valitaan aktiiviseksi pudotusvalikosta, haetaan dokumenttiluettelosta kaikki kyseiselle dokumentille asetetut tiedot. Niiden perusteella luodaan kuvan 43 koodin mukaisesti tarvittaessa alikansio sekä tallennetaan jatkoa varten kansioiden tiedot julkisiin muuttujiin. Näitä muuttujia pystytään hyödyntämään myöhemmin ohjelmakoodissa dokumentin tallentamisessa sekä vanhojen versioiden määrittelemisessä. Lopuksi riveillä 74 ja 75 kutsutaan vertailun ja pohjaluetteloiden hakutoiminnot.

```
54  'Find out where this current main file is located to find end file
55  CurrentFolder = (Application.ThisWorkbook.Path)
56
57  'Give path for template folder
58  strTemplateFolder = CurrentFolder & "\Lists\Templates\"
59
60  'Specify EndFolder (filename also) and LastFolder (no file name) as selected list
61  If strSelectedItemSubFolder = "" Then
62      EndFolder = CurrentFolder & "\Project Documents\Automation\" & strSelectedItemFolder & "\" & strSelectedItemDOCID
63      strLastFolder = CurrentFolder & "\Project Documents\Automation\" & strSelectedItemFolder & "\"
64  Else
65      strFolderName = CurrentFolder & "\Project Documents\Automation\" & strSelectedItemFolder & "\" & strSelectedItemSubFolder & "\"
66      strFolderExists = Dir(strFolderName, vbDirectory)
67      If strFolderExists = "" Then
68          MkDir (CurrentFolder & "\Project Documents\Automation\" & strSelectedItemFolder & "\" & strSelectedItemSubFolder) 'Create Subfolder
69      End If
70      EndFolder = CurrentFolder & "\Project Documents\Automation\" & strSelectedItemFolder & "\" & strSelectedItemSubFolder & "\" & strSelectedItemDOCID
71      strLastFolder = CurrentFolder & "\Project Documents\Automation\" & strSelectedItemFolder & "\" & strSelectedItemSubFolder & "\"
72  End If
73
74  Call ComparisonFileCombo_Change
75  Call ListTemplateComboBox_Change
```

KUVA 43. Kansioiden luonti ja tallentaminen muuttujiin

Kuvassa 44 on esitetty vertailun sekä luettelopohjien hakutoiminnon ohjelmakoodi. Vertailun ohjelmakoodissa riveillä 8-17 käytetään hyödyksi aiemmin "strLastFolder"-muuttujaan tallennettua tiedostopolkua, jonka avulla pystytään luetteloimaan kaikki kyseisessä kansiossa sijaitsevat tiedostot. Pohjaluetteloiden ohjelmakoodissa pystytään hyödyntämään sopivilta osin samaa toteutusta riveillä 35-44.

```
1   '=============================================
2   Private Sub ComparisonFileCombo_Change()
3   '=============================================
4   Dim i As Integer
5   Dim ListFiles1()
6
7   'Get different templates from template folder to combobox
8   If ComparisonFileCombo.ListCount = 1 Or ComparisonFileCombo.ListCount = 2 Then
9       If strLastFolder <> "" Then
10          ListFiles1 = ListFiles(strLastFolder, "comparison") 'ListFiles Function in Module1
11          For i = LBound(ListFiles1) To UBound(ListFiles1)
12              ComparisonFileCombo.AddItem (ListFiles1(i))
13          Next i
14      Else
15      MsgBox "List not selected"
16      End If
17  End If
18
19  'Get selected value from combobox and save value to variable
20  strComparisonFile = ComparisonFileCombo.Value
21
22  'Erase array
23  Erase ListFiles1
24
25  End Sub
26  '--------------ENDSUB---------------------
27
28  '=============================================
29  Private Sub ListTemplateComboBox_Change()
30  '=============================================
31  Dim i As Integer
32  Dim ListFiles2()
33
34  'Get different templates from template folder to combobox
35  If ListTemplateComboBox.ListCount = 1 Then
36      If strLastFolder <> "" Then
37          ListFiles2 = ListFiles(strTemplateFolder, "template") 'ListFiles Function in Module1
38          For i = LBound(ListFiles2) To UBound(ListFiles2)
39              ListTemplateComboBox.AddItem (ListFiles2(i))
40          Next i
41      Else
42      MsgBox "List not selected"
43      End If
44  End If
45
46  'Get selected value from combobox and save value to variable
47  strSelectedListTemplate = ListTemplateComboBox.Value
48
49  'Erase array
50  Erase ListFiles2
```

KUVA 44. Vertailun sekä luettelopohjien hakutoiminto

Täyttölomakkeen "Print excel list"-painiketta painamalla käyttäjä pystyy aloittamaan varsinaisen luettelon generoimisen. Ennen generointia suoritetaan toimenpiteet, joilla saadaan asetettua tietokantatiedosto valitun suunnittelualan mukaan sekä automaattisesti luotua uusi revisio dokumentista. Kuvassa 45 on esitetty näihin toimenpiteisiin laadittu ohjelmakoodi. Riveillä 226-233 saadaan yksinkertaisilla if-lausekkeilla määriteltyä tietokantatiedosto valitun suunnittelualan mukaan. Mikäli suunnittelualaa ei ole valittu, annetaan käyttäjälle virheviesti sekä keskeytetään toiminto. Riveillä 240-242 tarkastetaan, että jokin dokumentti on valittu, pohjaluettelo on valittu sekä vertailu on asetettu pois päältä. Lopulta riveillä 246-261 tarkastetaan uutta revisiota varten kansiossa olevat vanhat versiot. Mikäli työkalu löytää vanhan dokumentin kansiosta, luo se uuden tiedostonimen uudella revisiomerkinnällä.

```
224   'Set instrumentation or electrical settings
225   strDicipline = DiciplineComboBox.Value
226   If strDicipline <> "" Then
227       If strDicipline = "I" Then
228           strDiciplineText = "Automation"
229           strDataBaseExcel = "Instrumentation.xlsx"
230       ElseIf strDicipline = "E" Then
231           strDiciplineText = "Electrical"
232           strDataBaseExcel = "Electrification.xlsx"
233       End If
234   Else
235       MsgBox "Dicipline not selected"
236       Exit Sub
237   End If
238
239   'If user Selects Cable lists, run this part of the code
240   If strSelectedItem <> vbNullString Then
241       If strSelectedListTemplate <> vbNullString Then
242           If ComparisonEnabled = False Then
243               For i = 0 To 13
244               'Create new workbook to Lists folder and check if file already exists and create next revision if it exists
245                   FileExists = Dir(EndFolder & "_Rev." & i & ".xlsx")
246                   If FileExists = "" Then
247                       Set wb = Workbooks.Add
248                       Application.DisplayAlerts = False
249                       wb.SaveAs Filename:=EndFolder & "_Rev." & i & ".xlsx"
250                       Application.DisplayAlerts = True
251                       lastFile = strSelectedItemDOCID & "_Rev." & i & ".xlsx"
252                   Exit For
253                   Else
254                       Set wb = Workbooks.Add
255                       Application.DisplayAlerts = False
256                       wb.SaveAs Filename:=EndFolder & "_Rev." & i + 1 & ".xlsx"
257                       Application.DisplayAlerts = True
258                       lastFile = strSelectedItemDOCID & "_Rev." & i + 1 & ".xlsx"
259                   Exit For
260                   End If
261               Next
```

KUVA 45. Luettelon generoinnin aloittamisen ohjelmakoodi

Tiedostonimen luomisen jälkeen aloitetaan varsinainen luettelon luominen avaamalla määrittelyyn tiedostopolkuun uusi Excel-tiedosto. Tämän jälkeen haetaan määritellystä pohjaluettelosta etusivun lehti sekä liitetään se uuteen dokumenttiin. Kuvassa 46 on esitetty luettelon laatimiseen kehitetty ohjelmakoodi. Kuvassa

rivillä 264 luodaan uusi Excel-tiedosto ja riveillä 267-275 liitetään pohjaluettelon etusivu tähän tiedostoon. Ohjelmakoodissa käytetään hyväksi "closedBook"-komentoa, jolla mahdollistetaan etusivun kopiointi avaamatta varsinaista tiedostoa. Tämä nopeuttaa toimintoon vaadittavaa suoritusaikaa. Tämän jälkeen etusivu täytetään dokumentille asetetuilla tiedoilla. Rivillä 278 haetaan dokumenttiluettelosta rivi, jolla dokumentti sijaitsee. Riviä pystytään tämän jälkeen käyttämään hyödyksi kutsumalla funktiota "FillReportFrontSheet", joka tulostaa varsinaiset tiedot etusivulle. Funktion sisälle on rakennettu toiminto, jossa etusivulta etsitään tiettyä sanaa ja löytäessään ohjelma korjaa tämän sanan dokumenttiluettelosta löytämällään arvolla. Funktion suorittamisen jälkeen kopioidaan pohjaluettelosta myös varsinainen luettelon välilehti riveillä 285-289 ja nimetään se uudelleen rivillä 293. Koska uudelleen luodussa Excel-tiedostossa on aina yksi välilehti oletuksena, poistetaan tämä välilehti tiedostosta riveillä 296-298.

```
263         'Open new workbook to create list to it
264         Workbooks.Open Filename:=CurrentFolder & "\Project Documents\" & strDiciplineText & "\" & strSelectedItemFolder & "\" & strSelectedItemSubFolder & "\" & lastFile, UpdateLinks:=1
265
266         'Copy front sheet to new workbook
267         If strSelectedListTemplate <> "" Then
268             Application.ScreenUpdating = False
269             Set closedBook = Workbooks.Open(CurrentFolder & "\Lists\Templates\" & strSelectedListTemplate)
270                 closedBook.Sheets(1).Copy Before:=Workbooks(lastFile).Sheets(1)
271                 closedBook.Close SaveChanges:=False
272         Else
273             MsgBox "Template not selected, exiting from lists"
274             Exit Sub
275         End If
276
277         'Write values to Front Sheet
278         lRow1 = Workbooks(ArcoWorkbook).Sheets(ArcoDocumentsSheet).Range("B:B").Find(What:=strSelectedItem, LookIn:=xlValues).row 'Find row for specific document
279
280         Call FillReportFrontSheet(ArcoWorkbook, ArcoDocumentsSheet, lRow1)
281
282         Workbooks(lastFile).Worksheets(2).Activate
283
284         'Copy list sheet to new workbook
285         Application.ScreenUpdating = False
286         Set closedBook = Workbooks.Open(CurrentFolder & "\Lists\Templates\" & strSelectedListTemplate)
287             closedBook.Sheets(2).Copy After:=Workbooks(lastFile).Sheets(1)
288             closedBook.Close SaveChanges:=False
289         Application.ScreenUpdating = True
290
291         'Rename worksheet to list
292         Set NameWS = Worksheets("Sheet 2")
293         NameWS.Name = "List"
294
295         'Delete unnecessary sheet
296         Application.DisplayAlerts = False
297         Sheets("Sheet1").Delete
298         Application.DisplayAlerts = True
```

KUVA 46. Uuden luettelon luomiseen käytetty ohjelmakoodi

Luettelon välilehden lisäämisen jälkeen aloitetaan tiedon hakeminen tietokantatiedostosta varsinaiseen luetteloon. Kuvassa 47 on esitetty loppuosa ohjelmakoodista, jossa luettelo luodaan. Ohjelmakoodissa rivillä 309 avataan tietokantatiedosto käyttäjän valitseman suunnittelualan mukaan. Rivillä 312 kutsutaan luettelon tietojen hakemiseen kehitettyä aliohjelmaa. Aliohjelmassa haetaan pohja-

luettelosta käyttäjän antamien tietojen perusteella yhtäläisyyksiä tietokantatiedostossa olevalta välilehdeltä. Yhtäläisyydet löytäessään ohjelma tulostaa aina kyseiseen sarakkeeseen kaikki löytämänsä tiedot. Lopuksi luettelo-välilehdelle asetetaan tulostusalue toteutuneen luettelon mukaan, joka helpottaa tulostamisessa.

```
308             'Open database excel
309             Workbooks.Open Filename:=CurrentFolder & "\DataBase Excels\" & strDataBaseExcel, UpdateLinks:=1
310
311             'Print selected values to list
312             Call GenerateList(ArcoWorkbook, ArcoDocumentsSheet, lRow1)
313
314             Application.ScreenUpdating = True
315
316             'Set print area for cable list
317             PrintAreaLastRow = Workbooks(lastFile).Worksheets("List").Cells(Rows.Count, 2).End(xlUp).row + 5
318             Workbooks(lastFile).Worksheets("List").PageSetup.PrintArea = "$A$1:$M$" & PrintAreaLastRow
319
320             'Save workbook
321             Workbooks(lastFile).Save
322
323
324         Else
325         MsgBox "Comparison selected"
326         End If
327     Else
328     MsgBox "List template not selected"
329     Exit Sub
330     End If
331 Else
332 MsgBox "List not selected, window will be closed"
333 Exit Sub
334 End If
```

KUVA 47. Tiedon haku tietokannasta luetteloon

5.6 Piirikaavioiden generointi

Prosessisähköistyksen ja instrumentoinnin piirikaaviot ovat keskeinen osa suunnittelun aikana tuotettavaa dokumentaatiota. Uudistetussa työkalussa niiden generointi pidetään käyttäjälle mahdollisimman yksinkertaisena. Käyttäjän tulee luoda aluksi pohjakuvat, joiden mukaan generointi voidaan suorittaa varsinaisiin piirikaavioihin. Näihin kuviin sijoitetaan tietokantatiedostojen mukaisia numero- ja kirjainyhdistelmiä, joilla generoinnin aikana suorittava ohjelmakoodi löytää vastaavuuden tietokantatiedostoista ja korvaa nämä yhdistelmät varsinaisilla arvoilla. Jokaiselle tuotettavalle piirille annetaan pohjakuvan ja lopullisen dokumentin tiedostonimet. Generoitaessa ohjelmakoodi hakee pohjakuvan omasta kansiostaan ja korvaa siinä olevat yhdistelmät sekä lopulta luo tästä uuden tiedoston käyttäjän määrittelemällä tiedostonimellä. Piirikaavioita luodessa on vaihtoehtona tulostaa tiedostot joko .dxf- tai .dwg-tiedostoformaatteihin. Nopeuden kannalta edullisin vaihtoehto on tekstipohjainen tiedostoformaatti .dxf. Luottamuksellisuuden vuoksi varsinaista piirikaavioiden generoinnin ohjelmakoodia ei esitetä tässä työssä.

6 SUUNNITTELUTYÖ AIEMMIN JA TULEVAISUUDESSA

6.1 Suunnittelutyö aiemmin

Sähkö- ja instrumentointisuunnittelua on toteutettu jo monta vuosikymmentä. Nykypäivän suunnittelijoilla saattaa olla verrattain tuoreet muistikuvat siitä, kun työpaikoilla taisteltiin käytössä olevista tietokoneista. Lisäksi eri yrityksillä on saattanut olla hyvin rajallinen määrä käytössä olevien ohjelmistojen lisenssejä. Tämän johdosta aamulla aikaisin töihin saapuneet työntekijät ovat saattaneet vielä saada tietokoneen tai lisenssin käyttöönsä, kun taas myöhemmin töihin saapuvat ovat jääneet ilman ja joutuneet niin sanottuihin paperitöihin.

Vielä ennen tietokoneitakin on suunnittelua tehty erilaisilla alustoilla tai piirustuspöydillä. Tällöin piirustukset on toteutettu kynällä paperille piirtämällä. Tällöin piirustuksia ovat usein piirtäneet niin sanotut puhtaaksipiirtäjät. Suunnittelija ja piirustuksen piirtäjä ovat siis saattaneet olla eri henkilöitä, kun nykyään pääsääntöisesti he ovat aina yksi ja sama henkilö. Joidenkin kauan toimineiden yritysten piirustuksissa käytettävissä nimiöissä on näiltä ajoilta jääneet tietokentät, joissa täytetään eri kenttiin piirtäjän ja suunnittelijan puumerkit. Aikaa suunnitteluun on kulunut siis paljon ja se on tullut yrityksille erityisen kalliiksi.

6.2 Suunnittelutyö tulevaisuudessa

Tulevaisuudessa suunnittelutyön voidaan uskoa muuttuvan paljonkin. Jo nykyäänkin monet eri yritykset pyrkivät luomaan ohjelmistoihin ja työkaluihin entistä enemmän suunnittelijoita helpottavia toimintoja. Sen lisäksi suunnittelijoilla on usein useita eri omia työkaluja, joiden tarkoituksena on helpottaa tietyissä tehtävissä.

Nykypäivänä puhutaan todella paljon AI:sta (Artificial Intelligence) eli tekoälystä. Tekoäly on käytännössä erityisen laaja käsite, mutta se tulee varmasti vaikuttamaan myös suunnitteluun. Tekoäly on tähän päivään mennessä tuonut yhteis-

kunnalle muun muassa itsestään ajavia autoja, erilaisia robottiavustajia sekä paljon muita sovelluksia. Karkeasti voidaan ajatella, että tekoälyssä tietokone pyrkii oppimaan käyttäjien käytöksestä tai tietokonetta pystytän opettamaan toimimaan tietyllä tavalla. Lisäksi se pystyy soveltamaan ja hakemaan yhtäläisyyksiä ongelmiin muista, jo ratkaistuista ongelmista tai käyttäjien käytöksestä. Jo lähitulevaisuudessa tekoäly tai koneoppi voi tulla toimimaan vahvasti suunnittelijan rinnalla eräänlaisena apuna. Tulevaisuudessa voi olla hyvinkin mahdollista, että tekoäly suorittaa jo suurimman osan suunnittelijan tehtävistä vähentäen näin kustannuksia ja mahdollisia virheitä. Jo pelkästään jatkuvan kustannusten alentamisen sekä ajankäytön vähentämisen vuoksi kannustetaan keksimään ratkaisuja näihin ongelmiin, ja tekoälyn käyttäminen on varmasti yksi vaihtoehto.

7 JOHTOPÄÄTÖKSET

Vanhojen työkalujen tutkiminen ja käyttö suunnittelutyössä antoi laajaa kuvaa sähkö- ja automaatioinsinöörin työstä nykypäivänä. Opinnäytetyön laatimisen aikana huomattiin, että uudistetuille työkaluille on työntekijöiden keskuudessa kysyntää. Huomattiin myös, että monet suunnittelutyötä tekevät työntekijät ovat turhautuneita nykyisten työkalujen toimimattomuuteen, hitauteen ja vanhanaikaisuuteen. Työntekijät etsivät omassa työssään jatkuvasti uusia vaihtoehtoisia tapoja helpottaa ja nopeuttaa omaa suunnittelutyötään. Suurena haasteena eri työkaluille ja niiden toiminnollisuuksille kohdattiin loppuasiakkaiden erilaiset vaatimukset tuotettavien dokumenttien suhteen. Tämä puolestaan vaatii työkaluilta joustoa tuotettavien dokumenttien suhteen.

Uudistettuun työkaluun saatiin toteutettua monia eri toimintoja hyvinkin joustaviksi. Tärkeimpinä ominaisuuksina voidaan pitää dokumenttien hallintaa, luetteloiden ja piirikaavioiden generointia sekä prosessisähköistyksen komponenttien automaattisen valinnan pohjaa. Yrityksen työntekijöiden kanssa yhteistyössä laadittuihin vaatimuksiin ja tavoitteisiin päästiin lähes kokonaan. Joitakin tavoitteissa mainittuja toimintoja ei päästy kuitenkaan toteuttamaan kokonaisuudessaan, mutta toisaalta toisissa ominaisuuksissa onnistuttiin odotettuakin paremmin. Erityisesti dokumenttienhallinta sekä erilaisten luetteloiden generointi onnistuttiin toteuttamaan menestyksekkäästi. Vaatimuksissa ja tavoitteissa asetettiin työn kannalta hieman epärealistiset tavoitteet ja ohjelmoinnin laajuus osoittautui suunniteltua paljon suuremmaksi.

Kokonaisuudessaan uudistetun työkalun kehityksessä tuli vastaan todella suuri määrä haasteita ja ongelmia. Myös ohjelmoinnin laajuus uudistetun suunnittelutyökalun laatimisessa on todella suuri. Prosessisähköistyksen sekä instrumentoinnin suunnittelussa on lukuisia toimintoja, joita suunnittelijat tarvitsevat työssään ja tämän takia erityisen laajan työkalun suunnittelu sekä toteutus vie vuosia. Opinnäytetyön osalta työkalun toteutuksessa päästiin kuitenkin käytössä olevaan aikatauluun nähden pidemmälle, kuin aluksi osattiin odottaa. Työkalun kehityksessä pystyttiin alusta asti huomioimaan eri toimintoja ja suunnittelemaan sekä

toteuttamaan ne laadukkaasti. Työkalua on mahdollisuus lähteä kehittämään entistäkin laajemmaksi sekä laadukkaammaksi tämän työn pohjalta. Työkalun ohjelmakoodi pyrittiin myös kommentoimaan ja rakentamaan mahdollisimman selkeästi ylläpidon kannalta. Opinnäytetyön osalta työkalun käyttöönotossa ei päästy pitkälle ja käyttöönotto jatkuu laajemmin opinnäytetyön ulkopuolella. Alustava yrityksen työntekijöiltä saatu palaute työkalun toiminnallisuudesta on ollut positiivista.

Suunnittelutyökalun kehityksessä opittiin paljon ohjelmoinnista ja erilaisista työkaluista ja se oli erityisen mielenkiintoista. VBA-ohjelmoinnista opittiin opinnäytetyön ohella valtavasti ja sen oppiminen oli sopivan haastavaa. Kehityksen aikana AFRY Finland Oy:n työntekijät olivat myös vahvasti mukana eri toimintojen laatimisessa, joka auttoi ja innosti uudistetun työkalun kehityksessä. Työkalun odotetaan säästävän suunnittelijoiden aikaa sekä samalla helpottavan heidän jokapäiväistä työtään.

LÄHTEET

ABB, Sähkömiehen taskukirja. (2006). ABB Oy, Helsinki, Suomi, s.4. Luettu 27.01.2020.

ALMA, Tieto on kaikki kaikessa. Luettu 28.01.2020. https://www.alma.fi/

ALMA, Käyttöohje. Luettu 28.01.2020. https://www.alma.fi/

For...Next statement, Microsoft. 2018. Luettu 04.03.2020. https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/fornext-statement

GBdirect, The C book. Luettu 23.02.2020. https://publications.gbdirect.co.uk//c_book/

Getting started with VBA in Office, Microsoft. 2019. Luettu 28.02.2020. https://docs.microsoft.com/en-us/office/vba/library-reference/concepts/getting-started-with-vba-in-office

Long data type (Visual Basic), Microsoft. 2018. Luettu 07.03.2020. https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/data-types/long-data-type

LTrim, RTrim, and Trim functions, Microsoft. 2018. Luettu 03.03.2020. https://docs.microsoft.com/en-us/office/vba/Language/Reference/User-Interface-Help/ltrim-rtrim-and-trim-functions

ProElina, Sähkökoulutusmateriaali. (2008). Pöyry Plc. Luettu 25.01.2020.

Select Case statement, Microsoft. 2018. Luettu 03.03.202. https://docs.microsoft.com/en-us/office/vba/Language/Reference/User-Interface-Help/select-case-statement

With...End With Statement (Visual Basic), Microsoft. Luettu 05.03.2020. https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/with-end-with-statement

Workbook.ExportAsFixedFormat method (Excel), Microsoft. Luettu 04.03.2020. https://docs.microsoft.com/en-us/office/vba/api/Excel.Workbook.ExportAsFixedFormat

LIITTEET

Liite 1. Tavoitteet

Suunnittelutyökalun tavoitteet ja vaatimukset

Arco

jatkuu

1 JOHDANTO

Tämä dokumentti määrittelee vaatimukset sekä tavoitteet vuonna 2020 laadittavalle suunnittelutyökalulle AFRY Finland Oy:n käyttöön. Suunnittelutyökalu on osa Samu Lahden Tampereen Ammattikorkeakoululle laadittavaa opinnäytetyötä. Työkalu ja sen vaatimukset eivät ole suoranaisesti yhteydessä opinnäytetyöhön, vaan osa siitä suoritetaan opinnäytetyön ulkopuolella.

jatkuu

2 TAVOITTEET

Työkalun tavoitteet jaetaan kolmeen pääryhmään käyttöliittymä, instrumentointi ja prosessisähköistys. Työkalun suunnittelun edetessä vastaavia ryhmiä voidaan joutua lisäämään. Tavoitteiden tarkoituksena on antaa selvät suuntaviivat työkalun toteuttamiselle.

Opinnäytetyön ulkopuolelle jätettävät ominaisuudet on merkattu x -merkillä.

2.1 Käyttöliittymä

Vaatimukset sekä tavoitteet käyttöliittymälle:

- Excel -pohjainen, jossa säilytettävä Excelin ominaisuudet.
- Mahdollisimman yksinkertainen. Mahdollisesti ProElina -työkalua jäljittelevä nopeamman omaksumisen vuoksi vanhemmille työntekijöille.
- Käyttäjällä ei pidä olla mitään osaamista VBA-koodaamisessa. Muutostarpeissa mahdollisimman vähän tarvetta projektin ulkopuoliselle ”IT-Tuelle”.

2.2 Instrumentointi

Vaatimukset sekä tavoitteet työkalun instrumentointi osuudelle:

- Kytkentäluetteloiden generointi (Kotelo ja ristikytkentä) x
- Kaapeliluetteloiden generointi
- Instrumentti piirikaavioiden generointi .dxf & .dwg -muodoissa
- Ristikytkentäluetteloissa runkokaapelien kaikki johtimet oltava näkyvissä x
- Kytkentäluetteloissa myös automaatio- ja kenttäkaappien välireleet yms. näkyvissä. x

2.3 Prosessisähköistys

Vaatimukset sekä tavoitteet työkalun prosessisähköistys osuudelle:

- Keskuslähtöluetteloiden generointi x
- Kaapeliluetteloiden generointi x
- Prosessisähköistyksen piirikaavioiden generointi generointi .dxf & .dwg -muodoissa
- Komponenttien kuten kaapelien ja keskuskomponenttien automaattinen valinta.

2.4 Dokumenttien hallinta

Vaatimukset sekä tavoitteet työkalun dokumenttien hallinnalle:

- Mahdollisuus integroida Share/Bulk Tool työkaluun x