Tuomas Kurittu

Sähkösuunnittelun tehostaminen Excel VBA-ohjelmoinnin avulla

Opinnäytetyö
Sähkötekniikan koulutusohjelma

Joulukuu 2014

MAMK
University of Applied Sciences

KUVAILULEHTI

MAMK University of Applied Sciences	**Opinnäytetyön päivämäärä** 29.11.2014
Tekijä(t) Tuomas Kurittu	**Koulutusohjelma ja suuntautuminen** **Sähkötekniikan koulutusohjelma**

Nimeke

Sähkösuunnittelun tehostaminen Excel VBA-ohjelmoinnin avulla

Tiivistelmä

Opinnäytetyöni toimipaikka oli Kouvolan toimiston Etteplan Design Center. Työni tavoitteena oli tehostaa ja nopeuttaa sähkösuunnittelua lyhentämällä sähkösuunnitteluprojektin alussa kuluvaa turhaa aikaa. Tämä oli tarkoitus toteuttaa tietokoneohjelmointia apuna käyttäen.

Tulimme siihen tulokseen, että Microsoft Visual Basic for Applications olisi paras mahdollinen ohjelma opinnäytetyöni tekemiseen. Tehtäväni oli luoda yhtenäinen Excel-taulukko tilaajalle lähetettäväksi aina ennen suunnittelutyön aloitusta. Taulukon tulisi sisältää kaikki tarvittavat tiedot järjestelmistä ja laitteista, joita asiakas haluaisi projektissa käytettävän

Uuden toimintamallin perusidea on siis se, että tilaaja täyttää hänelle laaditun yhtenäisen Excel-taulukkopohjan ja lähettää sen takaisin Etteplanille. Sen jälkeen tiedot laitteista ja järjestelmistä on tarkoitus ajaa Etteplanilla käytössä olevaan Access-tietokantaan. Minun tehtävänä oli siis taulukon luomisen lisäksi suunnitella Excel VBA -ohjelmoinnin avulla linkki Excelin ja Accessin välille.

Opinnäytetyöni haluttu lopputulos saavutettiin, eli Excel-taulukkopohjan sisältämät tiedot pystytään nyt monien eri vaiheiden jälkeen siirtämään Accessiin nopeasti ja vaivattomasti. Uuden käytännön ansiosta lähtötietojen keräämiseen kuluu jatkossa paljon vähemmän aikaa ja se pystytään toteuttamaan paljon yksinkertaisemmin.

Asiasanat (avainsanat)

Excel, VBA, Access-tietokanta, ohjelmointi, tiedot.

Sivumäärä	**Kieli**	**URN**
28	Suomi	

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi Hannu Honkanen	**Opinnäytetyön toimeksiantaja** Etteplan Design Center Oyj, Kouvola

DESCRIPTION

MAMK University of Applied Sciences	**Date of the bachelor's thesis** 29.11.2014
Author(s) Tuomas Kurittu	**Degree programme and option** Electrical Engineering

Name of the bachelor's thesis

Intensification of electrical design with Excel VBA-programming

Abstract

Etteplan Design Center Kouvola was the office where I made my thesis. The objective of my work was to optimize and speed up the electric planning by eliminating the waste of time in the beginning of electric planning project. This was planned to be implemented by using programming.

We came to the conclusion that Microsoft Visual Basic for Applications would be the best programme to do this thesis. My job was to create a coherent Excel chart to be sent to a subscriber. The chart should include all the necessary information about the systems and equipment which customer would like to use in his project.

The main idea is that the customer will fill in the coherent Excel chart made for him and then send it back to Etteplan. After that the information about equipment and systems are transferred to an access database which Etteplan is using. Besides creating the chart, my job was to plan the link between Excel and Access using Excel VBA-programming.

The desired result of my thesis was achieved so the information of Excel chart can now be transmitted to Access quickly and effortlessly. Due to the new data processing system collecting the initial data consumes much less time in the future and it can be implemented much more simply.

Subject headings, (keywords)

Excel, VBA, access database, programming, information.

Pages	**Language**	**URN**
28	Finnish	

Remarks, notes on appendices

Tutor	**Bachelor's thesis assigned by**
Hannu Honkanen	Etteplan Design Center Oyj, Kouvola

Sisällys

1 JOHDANTO

Nykyaikana vaatimukset sähkösuunnittelutyölle ovat paljon kovemmat kuin mitä ne olivat entisaikaan. Sen lisäksi että suunnittelulta vaaditaan korkeatasoista laatua, niin myös työnteon nopeudella on hyvin suuri merkitys suunnitteluyrityksen toiminnassa. Kilpailu työmarkkinoilla on entistä kovempaa, joten työnteon tehokkuus vaikuttaa todella paljon markkinoilla pärjäämiseen.

Tärkeimpänä käännekohtana sähkösuunnittelupalvelujen kehittämisessä voidaan pitää sitä aikaa, jolloin alettiin käyttämään tietokoneita työnteon tukena. Verrattuna käsin piirtoon tämä mahdollistaa sähkökuvien muokkauksen jälkeenpäin. Kuvat pystytään myös piirtämään paljon nopeammalla tahdilla. Työntekoon käytettävät ohjelmistot kehittyvät koko ajan, ja käytännöissä tapahtuu jatkuvasti suuria muutoksia.

Pyrin opinnäytetyössäni tietokoneohjelmoinnin osalta kertomaan Excel VBA-ohjelmoinnista oleellisimmat asiat sekä erityisesti, mitä tämäntyyppisen ohjelman tekemisessä vaaditaan ja minkä omaksumisessa meni eniten aikaa tehdessäni ohjelmointityötä. Sen jälkeen aion lyhyesti kertoa sähkösuunnittelun kannalta oleellisista ohjelmistoista ja tiedonhallintajärjestelmistä sekä siitä, kuinka nykyaikana tietokantoja voidaan käyttää sähkösuunnittelun tukena.

Tämän jälkeen kerron vielä siitä, kuinka opinnäytetyöni ansiosta sähkösuunnittelutyötä pystytään tehostamaan. Esittelen tilaajalle laatimani taulukon ja sen toimintaperiaatteen verraten sitä Etteplanilla käytössä olevaan vanhaan toimintamalliin. Lopuksi näytän vielä, kuinka Excel-taulukon sisältämät tiedot saadaan ajettua Access-tietokantaan valmiin koodin avulla. Viimeiseksi kerron vielä ohjelmointityöskentelyni eri vaiheista tehdessäni ohjelmakoodia.

2 ETTEPLAN DESIGN CENTER

Opinnäytetyön toimeksiantaja on Kouvolan toimiston Etteplan Design Center. Suomalaislähtöisen Etteplanin erikoisalaa ovat teknisen tuoteinformaation palvelut ja ratkaisut sekä teollisten laitteistojen suunnittelu. Etteplanin päätoimipiste on Hollolassa, jossa yhtiö myös perustettiin vuonna 1983. Kaiken kaikkiaan Etteplan työllistää lähestulkoon 1800 asiantuntijaa ja suunnittelijaa yli 40 maassa. Eniten toimipisteitä on Suomessa ja Ruotsissa, mutta niitä on myös Kiinassa, Yhdysvalloissa, Alankomaissa sekä Tanskassa. /1./

Tällä hetkellä Etteplanilla on hyvin vahva markkina-asema Pohjoismaissa ja Kiinassa. Se on saavutettu mm. kustannustehokkailla palveluratkaisuilla, palveluiden ja tuotteiden kilpailukykyisyydellä, ennakoinnilla sekä ennen kaikkea asiakaslähtöisyydellä. Vastaavasti yrityksen tärkeimpiin pääarvoihin kuuluvat asiakastyytyväisyys, ammattitaitoinen toimintatapa sekä henkilöstön hyvinvointi. /1./

Kouvolan toimisto perustettiin LCA Engineering Oy –nimikkeellä vuonna 1993 ja vuonna 2007 siitä tuli osa Etteplan Oyj –konsernia. Kouvolassa toiminta painottuu erityisesti puunjalostusteollisuuteen. Toimipisteen alaisuuteen kuuluu monipuolisesti automaatio-, sähkö-, prosessi-, energia-, LVI-, mekaniikka- ja tehdassuunnittelu. Sähkö- ja automaatio-osasto vastaa mm. automaation ja prosessisähköistyksen suunnittelusta. Toimintaan liittyy myös tele- ja turvatekniikka sekä asennusten käyttöönotto ja valvonta./1./

3 OHJELMOINTI

Yksinkertaisimmillaan ohjelmointi voidaan määritellä siten, että se on toimintaohjeiden antamista ennalta määritettyjen toimenpiteiden suorittamiseksi. Helppona esimerkkinä ohjelmoinnista voidaan pitää mikroaaltouunin käyttämistä. Tällöin uunille annetaan selkeät ohjeet siitä, kuinka suurella teholla sekä kuinka kauan sen pitää toimia. Ohjelmoinniksi lasketaan myös se, kun annamme jollekin henkilölle puhelimessa ajo-ohjeet, joiden avulla hänen pitäisi selviytyä perille vieraaseen paikkaan./2./

Mikroaaltouunin käyttäminen ja ajo-ohjeiden antaminen puhelimessa ovat ohjelmoinnin näkökulmasta täysin samanlaisia toisiinsa nähden, mutta niissä käytetään eri työvälineitä. Tehtävän ratkaisuun käytettävissä olevista välineistä määräytyy se, mitä työvälineitä ohjelmointiin tullaan valitsemaan. Ihmiset voivat kommunikoida joko puhumalla, kirjoittamalla tai näiden yhdistelmänä. Samoin ohjelmoinnissa pystytään valitsemaan eri toteutustapoja riippuen tehtävän luonteesta./2./

Kaiken kaikkiaan ohjelmoinnissa on olemassa hyvin paljon erilaisia tasoja. Tehtävän ratkaisuun valittu työväline määrää sen, millä tasolla ohjelmointia voidaan harjoittaa. Työskentely on kehittyneintä silloin, kun käytetään korkean tason työvälineitä, joiden tekemiseen ja kehittämiseen on panostettu. Kyseisten työvälineiden käsitteet ja ilmaisut muistuttavat jopa ilmaisuja ja käsitteitä, joita käytetään luonnollisessa kielessä. Matalan tason työvälineet puolestaan mahdollistavat työskentelyn ainoastaan alkeellisilla ja yksinkertaisilla käsitteillä sekä ilmaisuilla. /2./

3.1 Tietokoneohjelmointi

Tietokoneohjelmoinnissa mitään ei pystytä tekemään ilman ohjeita eli ohjelmaa. Tietokoneen toiminta perustuu siihen, että se noudattaa tietokoneohjelman antamia käskyjä, joiden mukaan se toimii jättäen kaiken muun toiminnan ulkopuolelle. Sen takia tietokone soveltuu erinomaisesti operaatioihin, jotka toistuvat useaan kertaan sekä vaativat nopeaa ja tarkkaa toistamista. Tietokone onkin ihmiseen verrattuna hyvin paljon tarkempi, nopeampi, sekä sillä ei ole keskittymisongelmia. Se ei myöskään kyllästy tai häiriinny kovastakaan metelistä./3./

Tietokoneelle annetaan ohjeita syöttäen sille ykkösiä ja nollia, mikä perustuu virrankulun vaihteluun. Perusperiaate on hyvin yksinkertainen, ykkösellä virta kulkee ja nollalla ei kulje. Kun niitä laitetaan monta peräkkäin, niin lopputuloksena voi olla vaikkapa tekstinkäsittelyohjelma tai peli./3./

Ykkösten ja nollien kirjoittamisessa ei ole kuitenkaan mitään järkeä, koska se olisi hyvin työlästä. Tämän takia käytetäänkin symboleilla toimivia korkeamman tason ohjelmointikieliä. Tulkin tai ohjelmointikielen kääntäjän tehtävänä on muuntaa ohjelmakoodi ykkösiksi ja nolliksi, jotta tietokone ymmärtäisi sen./3./

3.2 Ohjelmointikielen määritelmä

Tietokoneohjelman tulee siis toimia ennalta määriteltyjen ohjeiden mukaisesti, mikä tapahtuu ohjelmointikielen avulla. Ohjelmointikieli on tavallaan ohjelmointia varten luotu ilmaisukieli, jonka avulla ohjelmoijat tekevät työtään. Ohjelmointikielet muodostavat eri kategorioita, ja kaikissa ohjelmointikielissä on erilaisia rakenteita ja komentoja. Näiden avulla ohjelmoija puolestaan tekee täsmälliset toimintaohjeet. Ohjelmointikieli voidaan kuvata liittymäksi ohjelmistotuotantoon. Ohjelmoija käyttää kyseistä liittymää apuna silloin, kun hän koodaa tietokoneohjelmaa. Tapahtumaa voidaan kuvata eräänlaiseksi kielenkäyttöprosessiksi. /4./

Laitteistoa, ohjelmistoa sekä ohjelmaa käyttävät ihmiset asettavat rajat, joiden puitteissa tietokoneohjelma toimii. Yksikäsitteisyyttä voidaan pitää ohjelmointikielen yhtenä merkittävimpänä ominaisuutena. Se tarkoittaa käytännössä sitä, että ohjelmointikielellä luodut ilmaukset ovat tulkittavissa ainoastaan yhdellä tavalla. Syntaksi puolestaan on termi, millä tarkoitetaan ohjelmointikielen kielioppia. Sillä määritetään se, että minkälaisia ohjelmia voidaan kirjoittaa. Ohjelmointikieliä kutsutaan formaaleiksi kieliksi. /4./

Asiantuntijan käsissä ohjelmointikielellä voidaan saada hyvin paljon aikaan. Ohjelmointikieli on kuin väline, mutta se ei vielä kuitenkaan takaa korkeaa laatua. On olemassa erityisiä tekniikoita, joiden avulla parhaat ohjelmoijat pystyvät hyödyntämään koko kapasiteetin välineistänsä saavuttaen täten parhaan mahdollisen lopputuloksen. Tietokoneohjelmointi onkin kehittynyt hyvin paljon lyhyellä aikavälillä. Uusia mene-

telmiä on kehittynyt hyvin paljon tietojenkäsittelijöiden toimesta. Sen ansiosta ohjelmista on tullut luotettavampia ja ohjelmoijat kykenevät työskentelemään entistä tehokkaampaan sekä tuottoisampaan tahtiin. /4./

3.3 Kapselointi ja tiedon kätkeminen

Kapseloinnilla tarkoitetaan sitä, että jokin asia tai esine, kuten vastus, on käyttökelpoinen komponentti aivan sellaisenaan. Tiedon kätkeminen puolestaan viittaa siihen, että kapseloitu komponentti on käyttökelpoinen, vaikka sen sisäistä osaa ei tunnettaisi. /4./

Kun esimerkiksi sähköinsinööri valitsee oikeanlaisen vastuksen johonkin laitteeseen, niin hän ei tietenkään ala sitä itse rakentamaan. Sen sijaan hän hakee varastosta tarvittavan vastuksen, joka on tunnistettavissa värikoodien avulla. Sähköinsinöörille ei ole mitään väliä sillä, mitä vastuksen sisällä tapahtuu tai miten se toimii. Komponentin tarpeisiin vastaaminen on kaikista oleellisin asia. /4./

Kapselointia käytetään erityisesti niin sanotussa olio-ohjelmoinnissa. Olio-ohjelmoinnissa huomio kohdistuu ohjelmien järjestelytapoihin, joilla ei kuitenkaan viitata ohjelmointikielten ominaisuuksiin, vaan ohjelmointitekniikoihin. Ne ovat kuitenkin käyttökelpoisia melkein kaikissa ohjelmointikielissä. Kapselointitekniikalla on pystytty parantamaan hyvin paljon ohjelmistojen laatua, ei ainoastaan olio-ohjelmoinnissa vaan myös paljon ennen sen syntymistä. Kapselointi ja tiedon kätkeminen ovat siis hyvin oleellisia termejä ohjelmoinnissa. /5./

4 EXCEL-OHJELMOINTI

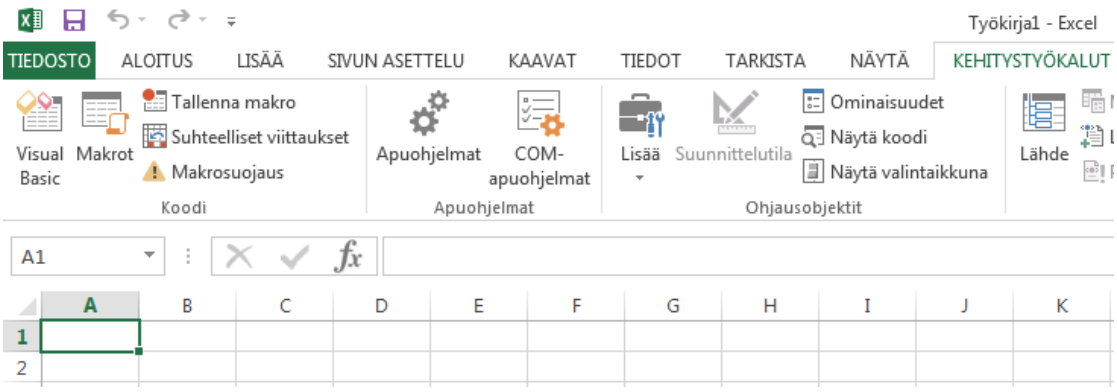
Excel-ohjelma on hyvin paljon laajempi ja monipuolisempi kuin monet ihmiset tietävätkään. Sen käyttöä voidaan paitsi automatisoida, niin myös laajentaa hyvin merkittävästi Visual Basic for Applications –ohjelmien avulla, mikä on erittäin laaja aihepiiri. Siitä käytetään lyhennettä VBA. VBA-ohjelman tekemisestä yksinkertaisin esimerkki on se, kun Excelissä luodaan toimenpidesarja, joka nauhoitetaan. Samalla kun nauhoitetaan, niin Excel kirjoittaa VBA-ohjelmakoodin. Käytännössä VBA-ohjelmoinnista hyödytään kuitenkin vasta sitten, kun ohjelmakoodi kirjoitetaan itse. Myös Wordin, PowerPointin ja erityisesti Accessin käytössä voidaan hyödyntää VBA-ohjelmointia./6, s. 1./

Jotta Excel VBA-ohjelmoinnin voisi oppia hyvin, täytyy Excelin käyttö hallita hyvin. Ohjelmointi on hyvin vaativaa, mutta kuitenkin mahdollista oppia, vaikka aikaisempaa ohjelmointikokemusta ei olisikaan. Alussa suurimmat vaikeudet Excel-ohjelmoinnin suhteen liittyvät varmasti VBA-kielen oleellisimpien rakenteiden hallitsemiseen sekä Excelin objektimallin käsittämiseen. /6, s.1./

Excel-ohjelmoinnissa ohjelmointikieli ja sen omaksuminen ei välttämättä ole se kaikista suurin haaste. Kun ohjelma on onnistunut ja hyödyllinen, niin tehtävä saadaan suoritettua tai ongelma ratkaistua. Aluksi tehtävän suorittamista tai ongelman ratkaisua pitää miettiä vaiheittain. Tämän jälkeen voidaan alkaa pohtia sitä, miten vaiheet pystytään parhaiten suorittamaan ohjelmointikieltä käyttäen. /6, s.1./

4.1 Ohjelmoinnin aloittaminen

Kun Excel-ohjelmointi aloitetaan, niin aivan ensiksi täytyy ottaa käyttöön Excelin kehitystyökalut. Jotta ne saataisiin käyttöön, pitää ensiksi painaa Tiedosto-nappia, jonka jälkeen valitaan ”Asetukset” ja sieltä sitten ”Muokkaa valintanauhaa”. Tämän jälkeen valitaan ”Kehitystyökalut” Päävalintalehdet-luettelon kohdalta ja viimeiseksi painetaan OK-nappia. Kehitystyökalujen käyttöönoton jälkeen ovat ne myöhemmin aina automaattisesti käytössä. /6, s.2./



KUVA 1. Kehitystyökalut /6, s.2/

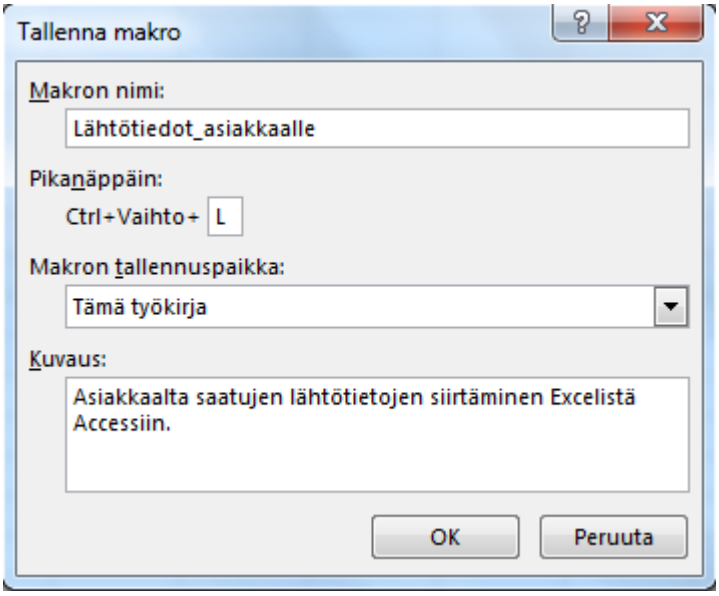
Ennen kuin ohjelmointi aloitetaan, niin pitää myös tarkistaa, että Excelin makrosuojaus on kunnossa. Sen tarkoituksena on estää makrojen, eli ohjelmien suorittamista, jos käyttäjä ei ole antanut siihen lupaa. Oikean suojaustason saamiseksi valitaan ”Makrosuojaus” Kehitystyökalut-välilehdeltä. Paras suojaustaso saadaan valitsemalla ”Poista käytöstä kaikki makrot ja ilmoita”. Tämän ansiosta tulee varoitus, jos avataan työkirja, joka sisältää ohjelmia. Mikäli käy niin, että Excel varoittaa makroista, täytyy vain valita ”Salli”. /6, s.2./

Kun makrosuojaus on kunnossa, niin sen jälkeen voidaan alkaa nauhoittamaan makroa. Ensin valitaan ”Nauhoita makro” Kehitystyökalut-välilehdeltä, jonka jälkeen halutut tiedot makrosta voidaan tallentaa Nauhoita makro-lomakkeen kohdalle. Ainakin makron tallennuspaikka sekä nimi kannattaa tallentaa. Myös makron kuvaus sekä käynnistyskirjain on mahdollista tallentaa. Sen jälkeen valitaan ”OK” sekä suoritetaan nauhoitettavat toimenpiteet. Jos halutaan poistaa edellisen toimenpiteen nauhoitus, voidaan käyttää Kumoa-toimintoa. Kun kaikki tarvittavat toimet on saatu suoritettua, niin viimeiseksi valitaan ”Lopeta nauhoittaminen”. /6, s.3./

Makroa tallennettaessa kannattaa se nimetä yksilöidysti siten, että makron nimi kuvaa sen toimintoa. Tallennuspaikkana voi käyttää kuvan 2 mukaisesti aktiivista työkirjaa, jolloin makro tallentuu kohtaan ”Tämä työkirja”. Sen voi kuitenkin tallentaa myös uuteen työkirjaan tai sitten omaan makrotyökirjaan, mikä löytyy kohdasta ”Omat.xls”. Uuteen- tai aktiiviseen työkirjaan tallennettuja makroja voi käyttää vain silloin, kun kyseinen työkirja on käytössä. Kun makron tallennuspaikkana käytetään Omat.xls- työkirjan paikkaa, niin makrot ovat käytettävissä aina, kun Exceliä käytetään. /7./

Termit ”Suhteellinen viittaus” ja ”Suora viittaus” on tärkeä hallita, ennen kuin makroa aletaan nauhoittamaan. Lopeta nauhoittaminen -työkaluriviltä löytyy Suhteellinen viittaus -painike, jonka avulla kyseisiä osoiteviittauksia pystytään käsittelemään. Kun laskennassa käytetään Suhteellista viittausta, niin se muuttuu aina lähtökohdan muuttuessa. Valitaan esimerkkinä solu B1 ja haluttu viittauspaikka kaavassa on vasemmanpuoleinen viereinen solu. Tällöin uudeksi soluosoitteeksi tulee A1. Vastaavasti kun kaavassa kopioidaan solu B1 alaspäin soluun B2, niin tällöin viitataan suhteellisesti soluun A2. Asia voidaan tiivistää siten, että käytettäessä suhteellista viittausta silloin viitataan lähtökohdasta aina yhtä kauas määritettyä suuntaa kohden. Kun käytetään suoraa eli absoluuttista viittausta, niin silloin viitataan lähtökohdasta huolimatta aina samaan kohtaan. Esimerkkinä kopioidaan solu B1 alaspäin. Siinä käytetään suoran viittauksen sisältävää kaavaa =$A$1. Tällöin solussa B2 viitataan yhä soluun =$A$1. Käytettäessä suoraa viittausta viitataan siis aina samaan osoitteeseen. /7./

Kuvasta 2 käy ilmi, että makroon voidaan myös lisätä käynnistyskirjain, jolloin valittua kirjainta ja CTRL-nappia painettaessa makro käynnistyy. Tämä toiminto onnistuu Nauhoita makro-lomakkeen kautta. Käynnistyskirjaimina voidaan käyttää niin isoja- kuin pieniäkin kirjaimia. Käynnistyskirjainta voi muuttaa tai sen voi poistaa Kehitystyökalut- välilehdeltä. Sieltä Makrot- kohdasta valitaan Makron nimi, painetaan Asetukset-nappia, tehdään halutut muutokset ja lopuksi painetaan OK- painiketta. /6, s.3./



KUVA 2. Nauhoitetun makron tallentaminen

Nauhoitettu makro pystytään suorittamaan Kehitystyökalujen kohdalta. Sieltä Makrot-kohdasta valitaan makron nimi ja painetaan Suorita–painiketta. Makron suorittaminen voidaan myös liittää käynnistyspainikkeeseen. Tällöin kehitystyökalut-välilehdeltä valitaan ”Lisää”, josta valitaan ”Lomakkeen ohjausobjektit”, josta voidaan lisätä painike. Painike yksinkertaisesti piirretään taulukkoon hiirellä. Painikkeen piirtämisen jälkeen valitaan liitettävän makron nimi ja painetaan OK-nappia. Painike sisältää tekstin, jota napsauttamalla käyttäjä voi kirjoittaa sen tilalle haluamansa tekstin. Lopuksi hiirtä täytyy vielä klikata painonapin ulkopuolella, jonka jälkeen painikkeeseen liitetty makro voidaan suorittaa klikkaamalla painiketta. Jos painiketta klikataan CTRL-nappi pohjassa, painike voidaan valita makroa käynnistämättä. /6, s.5./

4.1.1 VBE - Visual Basic Editor

Excel on paljon muutakin kuin pelkkä apuväline perinteiseen taulukkolaskentaan. Tämän mahdollistaa Excelin Visual Basic -editori. Sitä voidaan hyödyntää paitsi pelkästään Excelin ohjelmoimisessa, myös eri ohjelmien välisessä vuorovaikutuksen automatisoimisessa. Mm. työkirjan laskenta-arkkien välinen tiedonsiirto olisi hyvin vaivalloista, mikäli toiminta ei olisi automatisoitua. /8./

Visual Basic -editorin avulla koodi muodostuu automaattisesti, samalla kun suoritettavat toimet nauhoitetaan. Tällöin nauhuri muodostaa tehdyistä toimista VBA-projektiin koodin sisältävän moduulin. Nauhoitettua koodia joudutaan kuitenkin poikkeuksetta korjaamaan, joten hyvää ohjelmointitaitoa vaaditaan. Visual Basic ei kuitenkaan ole tyyliltään tai muotoseikoistaan kovin pikkutarkka, joten se on sinänsä suhteellisen helppo ohjelmointikieli. Muuttujia ei myöskään ole tarpeellista määritellä, sillä niistä muodostuu automaattisesti Variant-tyyppisiä muuttujia. Näin tapahtuu siis aina, kun muuttujille ei anneta tarkkaa määritystä./8./

Myös Excelin objektikirjaston tunteminen on hyvin tarpeellista, sillä Excelin osien hallinta onnistuu sen kautta. Kyseiseen kirjastoon päästään, kun valitaan Visual Basic-editorin ikkunan kohdalta ”View”, jonka kautta päästään Object Browseriin. Sieltä löytyviä objekteja ovat esimerkiksi Sheets ja Cells. Niiden avulla on mahdollista käsitellä laskenta-arkkeja sekä siellä olevia soluja. Laskenta-arkin alueita pystytään määrittele-

mään Range-objektin avulla. Sille määritetään jokin alue toimenpiteitä varten suoritettavaksi, esimerkiksi ”Range(A2:E6).Select.”. Tämä tarkoittaa sitä, että joltain arkilta tulee valituksi kyseinen alue. Sama asia pystytään tekemään Cells-ominaisuuden avulla. /8./

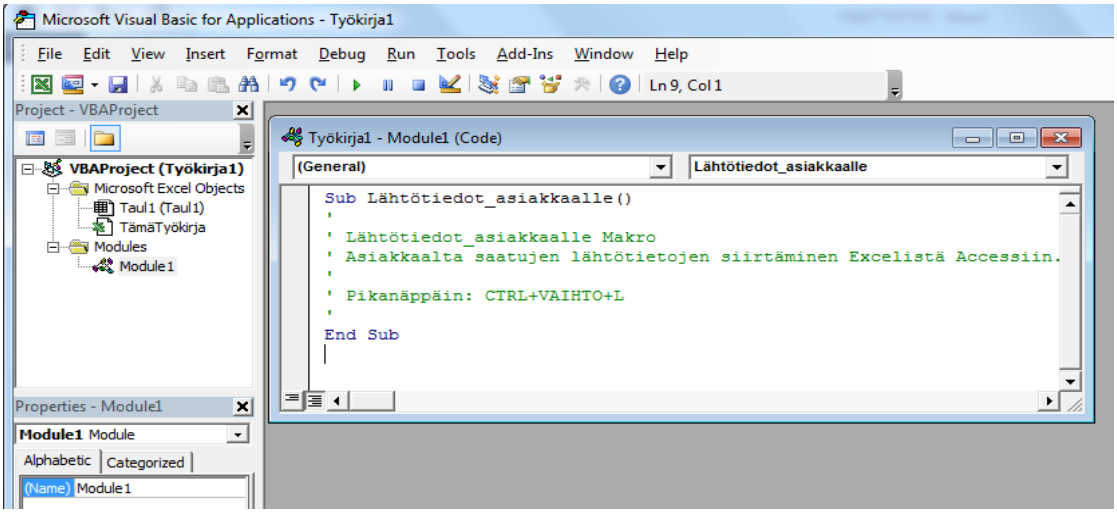
Voidaan ajatella, että objektit sisältävät osia, joita ovat ominaisuudet ja menetelmät. Objektin ominaisuuksilla tarkoitetaan erilaisia piirteitä, jotka liittyvät objektiin. Ne ovat muutettavissa, paitsi jotkut tietyt piirteet, joita voidaan ainoastaan lukea. Menetelmä puolestaan on toiminto, jonka objekti osaa ja mikä pystytään myös suorittamaan. Objektit eivät ole samanlaisia kuin perinteiset muuttujat, sillä objektien sisälle on kapseloitu niin ominaisuudet kuin menetelmätkin. Objektien ymmärtämiseksi niille on tehty ns. luokkakuvaukset. Jokaisen objektin voidaan ajatella kuuluvan johonkin tiettyyn luokkaan. Eri luokista puolestaan muodostuu luokkahierarkia. Ylimpiä luokkia kutsutaan isäluokiksi ja alimpia lapsiluokiksi. /8./

Yleisimmät objektit Excelissä ovat Application, Workbook, Worksheet, Window ja Range. Application kattaa Excelin kokonaisuudessaan, ja siinä on kaikki yleisasetukset. Se sisältää lisäksi Excelin sisäiset funktiot. Workbookiin puolestaan kuuluu ominaisuudet ja menetelmät, jotta Excelin työkirjaa pystytään käsittelemään. Laskentataulukon käsittelyä varten myös Worksheet-objektilla pystytään hallitsemaan ominaisuuksia sekä menetelmiä. Ikkunan minimointia sekä maksimointia varten on luotu Window-objekti, kun taas solualueiden sekä solujen hallinta onnistuu Range-objektin avulla. /8./

Excel-ohjelmoinnissa menetelmillä siis tehdään jotain, kun taas muuttujat ovat käyttökelpoisia joko funktioissa tai proseduureissa. Nämä eroavat toisistaan siten, että funktion avulla voidaan palauttaa arvo, toisin kuin proseduureilla, jotka ovat kykeneväisiä ainoastaan jonkin tehtävän suorittamiseen. Proseduuri aloitetaan Sub-sanalla, jonka perään kirjoitetaan ohjelmakoodi muuttujien avulla. Kun koodi on saatu kirjoitetuksi, loppuun kirjoitetaan vielä ”End Sub”. /8./

Siirtyminen VBE:n ja Excelin välillä tapahtuu näppäinyhdistelmän ALT-F11 avulla. Kuva 3 havainnollistaa sen, miltä työkirjaan tallennettu makro näyttää Visual Basic Editorissa. Oli Excelin kieliversio sitten mikä tahansa, niin VBE:n käyttöliittymä on silti aina englannin kielellä. Sen takia VBE:n käyttäminen voi olla aluksi hyvin vaikeaa.

Ensi alkuun kannattaakin makronauhurin tekemää koodia tarkkailla samaan aikaan, kun makroa nauhoitetaan. Näin saadaan hyvä käsitys Excel-toimintojen ja objektien VBA-kielisistä vastineista. Jopa VBA-ohjelmoinnin ammattilaiset käyttävät työssään apuna makronauhuria ja jälkikäteen muokkaavat koodin oikeanlaiseksi. /6, s. 5, 6./

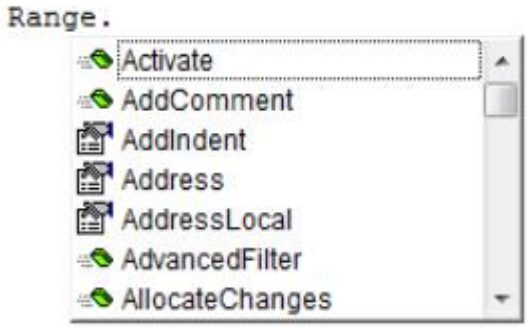


KUVA 3. Työkirjaan tallennettu makro Visual Basic Editorissa

4.1.2 Ominaisuudet ja työkirja

Objektien piirteisiin kuuluvat erilaiset ominaisuudet, kuten esimerkiksi sijainti, koko ja väri. Mm. Application-objektiin liittyy hyvin paljon ominaisuuksia, jotka olisi tarpeellista hallita. Kun objektin eteen kirjoitetaan sana ”Active”, niin se tarkoittaa sitä, että viitataan aktiivisena olevaan objektiin. Jos esim. VBE:iin kirjoitetaan ”ActiveWorkbook”, niin silloin viitataan aktiiviseen työkirjaan. Viittausta voidaan myös täsmentää, sillä jos Active-sanan perään kirjoitetaan ”Sheet”, silloin viittauksen kohteena on aktiivisen työkirjan sivu. Myös ”ActiveCell”, jolla viitataan aktiiviseen soluun, on hyvin usein käytetty ominaisuus./6, s. 15./

Tilanteeseen sopivan ominaisuuden pystyy löytämään helposti, sillä VBE auttaa sen löytämisessä. Ei tarvitse muuta kuin kirjoittaa objektin nimi ja sen jälkeen piste. Kun näin tehdään, niin VBE antaa listan, mikä sisältää kirjoitetun objektin menetelmiä ja ominaisuuksia. Kuva 4 havainnollistaa asiaa, eli siinä kirjoitetaan ”Range” ja sen perään piste./6, s. 16./



KUVA 4. Objektin menetelmien ja ominaisuuksien löytäminen /6, s. 16/

Nuolinäppäinten avulla pystytään selaamaan luetteloa, mutta helpommalla päästään, kun kirjoitetaan ainoastaan ominaisuuden aloituskirjain. Tällöin päästään automaattisesti menetelmiin ja ominaisuuksiin, jotka alkavat sillä kirjaimella. Halutun ominaisuuden päällä painetaan TAB/SARKAIN-painiketta, jolloin VBE automaattisesti kirjoittaa koodiin kyseisen ominaisuuden./6, s. 16./

Workbooks-kokoelmasta löytyy Add-menetelmä, jonka avulla pystytään luomaan uusi työkirja kirjoittamalla ”Workbooks.Add”. SaveAs-menetelmällä luonnollisesti tallennetaan työkirja ja nimeäminen tapahtuu Filename-argumenttia käyttäen. Työkirjan tallennuspaikkana käytetään oletuskansiota, mikäli polkua ei ole määritelty./6, s. 16./

Workbooks-kokoelmassa olevaa Open-menetelmää käytetään työkirjaa avattaessa, jolloin siitä tulee Workbooks-kokoelman osa. Kun työkirja halutaan sulkea, niin silloin käytetään Close-menetelmää, jonka avulla voidaan myös sulkea kaikki avoimet työkirjat. Workbook-objektista löytyy lisäksi FullName-ominaisuus, jota käyttämällä saadaan työkirjan nimi sekä tallennuspolku selville./6, s.17./

4.2 VBA

VBA-ohjelmointiin liittyy hyvin oleellisesti muuttujat sekä vakiot. Niiden ansiosta laskutoimitusten tekeminen sekä asioiden välisten suhteiden vertaileminen on mahdollista. Muuttujat ovat hyvin keskeisiä peruspilareita ohjelman tekemisessä ja niissä säilötään tietoa, mikä saattaa muuttua ohjelman ajamisen aikana. Tällaista tietoa voi olla vaikkapa

arvoa nostattava laskuri ajon aikana. Pysyvää arvoa ohjelman suorittamisen aikana kutsutaan vakioksi. Arvoilla on kuitenkin samanlaiset käyttötavat, oli kyse sitten muuttujista tai vakioista. /9./

Kun muuttujaa aletaan esittelemään, niin silloin käytetään joko Dim, Public tai Private-sanaa. Mikäli muuttuja on kaikille ohjelmille yhteinen, aloitetaan sen esittely Public-sanalla. Muuttujaa nimettäessä käytetään kuitenkin melkein aina Dim-sanaa, jos se määritellään aliohjelmassa. Tällöin muuttujan arvo on nolla ennen ohjelman ajoa, mutta ajon jälkeen siitä tulee tyhjä. Mikäli määrittelyyn käytetään Static-sanaa, säilyy muuttujan arvo, kun aliohjelmasta poistutaan. /9./

Jotta muuttujien määrittely olisi helpompaa, on niillä olemassa erilaisia tietotyyppejä. Ne on esitelty kuvassa 5. Tärkeimmät niistä ovat teksti, lyhyt luku sekä pitkä luku. Tietotyypin määrityksessä olisi järkevää käyttää aina tarkkaa määritystä, mutta jos sitä ei määritellä, muuttujan tietotyypiksi muodostuu tällöin automaattisesti Variant. Tällöin on sallittua käyttää sekä numeroita että kirjaimia, sillä Variant on yleinen tietotyyppi. Kun tietotyyppiä aletaan määrittämään, käytetään alussa As-sanaa. /9./

Tietotyyppi	Muistitilan tarve	Arvoalue
Boolean	2 tavua	True tai False
Byte	1 tavu	0 – 255
Integer	2 tavua	Kokonaisluku välillä -32.768 – 32.767
Long	4 tavua	Kokonaisluku välillä -2.147.483.648 – 2.147.483.647
Double	8 tavua	-1,79769313486232E308 – -4,94065645841247E-324 4,94065645841247E-324 –1,79769313486232E308
Currency	8 tavua	-922.337.203.685.477,5808 – 922.337.203.685.477,5807
Date	8 tavua	Tammikuun 1. vuonna 0100 – Joulukuun 31. vuonna 9999
Object	4 tavua	Mikä tahansa objekti
String (pituus voi vaihdella)	10 tavua + merkkijono	0 – noin 2 miljardia merkkiä
String (kiinteä pituus)	merkkijonon pituus	1 – noin 65.400 merkkiä
Variant	16 tavua tai enemmän	Jos muuttjan tyyppiä ei määritellä, niin se on Variant

KUVA 5. Yleisimmät muuttujan tietotyypit/6, s. 30/

Yleensä käy niin, että sama objekti toistuu ohjelmassa monta kertaa, jolloin se kannattaa asettaa Object-tyyppisen muuttujan arvoksi. Tämä onnistuu Set-lauseen avulla, esimerkiksi ”Set Solu =Range(”B2:E6”)”. Kyseisessä esimerkissä solualueesta tulee Range-tyyppisen muuttujan arvo. /6, s. 30./

Kun muuttuja on määritelty Dim-lauseella ohjelman sisällä, ei se ole käytettävissä missään muualla. Muuttuja on olemassa end Sub-lauseeseen saakka, jonka jälkeen se lakkaa olemasta. Dim-lauseinen muuttuja voidaan kuitenkin myös määritellä moduulin alussa, eli ennen ohjelmia. Tällöin sen arvo säilyy aina siihen asti, kunnes työkirja suljetaan. Jotkut haluavat, että muuttujaa voidaan käyttää myös sellaisten moduulien ohjelmissa, jotka liittyvät toiseen työkirjaan. Silloin muuttuja pitää määritellä Public-lauseella moduulin alussa. /6, s. 32./

Vakiot puolestaan ovat hyödyllisiä silloin, kun ohjelmaa tarvitsee päivittää useasti. Ne myös selventävät ohjelman luettavuutta. Vakioksi kannattaa määrittää se arvo, mikä ohjelmassa toistuu eniten. Vakion esittelyssä käytetään useimmiten Const-sanaa. Julkisuus puolestaan toteutuu samalla tavalla, oli kyseessä sitten vakio tai muuttuja. /6, s. 32; 9./

4.2.1 Aliohjelmat ja funktiot

Kun ohjelmoidaan, niin lähes aina jossain vaiheessa aliohjelmien ja funktioiden käyttö tulee tarpeelliseksi. Niitä käytetään silloin, kun ohjelmassa täytyy toistaa monta kertaa jokin tietty kohta. Aliohjelmat saavat tietyt parametrit, joiden pohjalta ne suorittavat vaaditun koodin osan. Funktion avulla myös arvon palauttaminen on mahdollista. Aliohjelmien määrityksessä käytetään Sub-käskyä, kun taas funktiot määritellään Function-käskyä käyttäen. /10, s. 33./

Aliohjelmien avulla pystytään selkeyttämään sekä lyhentämään koodia. Hyvin kirjoitettu toimiva aliohjelma on usein myös myöhemmin käyttökelpoinen joissain muissa ohjelmissa. Aliohjelmien sijoituspaikka on aina Sub…End Sub-lauseiden välissä. Ensiksi siis tulee aliohjelman nimi, jonka jälkeen kirjoitetaan sulkuihin sen parametrit ja tyypit. Suluissa oleville muuttujille ei tarvitse antaa erikseen eri määrittelyjä, sillä ne toimivat täysin samalla nimellä aliohjelman sisällä. Parametrejä voi olla enemmänkin kuin yksi, ja ne erotetaan pilkulla toisistaan. /10, s. 33./

Funktioiden perustoiminta on lähes samankaltainen kuin aliohjelmilla, tosin ne omaavat muuttujatyypin sekä kykenevät palauttamaan jonkin arvon, toisin kuin aliohjelmat. Aluksi siis funktio määritellään, ja heti perään tulee muuttujatyyppi, mikä funktion on

tarkoitus palauttaa. Funktion palautusarvon määrittely puolestaan tapahtuu samoin kuin muuttujien kanssa, eli yhtäsuuruusmerkkiä käyttäen. /10, s. 34./

Samoin kuin muuttujissa, niin myös funktioissa ja aliohjelmissa voidaan avainsana sijoittaa määrittelyn eteen. Kyseiset avainsanat ovat siis joko Private tai Public. Jos käytetään Private-avainsanaa, niin silloin aliohjelma tai funktio pätee vain sen sijaintipaikassa, joko moduulin, luokan tai formin sisällä. Käytettäessä Public-avainsanaa ovat aliohjelmat ja funktiot kutsuttavissa mistä tahansa ohjelman kohdasta. /10, s. 34./

VBA kattaa myös joukon sisäänrakennettuja valmisfunktioita. Ohjelmaa kirjoitettaessa on mahdollisuus saada luettelo näistä kyseisistä valmisfunktioista. Tarvitsee vain kirjoittaa VBA ja sen perään piste. Tämä edellyttää kuitenkin sen, että Auto List Members -asetuksen täytyy olla päällä. Sen voi tarkistaa menemällä Tools-valikkoon, josta löytyy Options ja sieltä edelleen Editor. /6, s. 32./

VBA-ohjelmoinnissa kannattaa erityisesti alun testausvaiheessa käyttää MsgBox-funktiota. Sen avulla ohjelmoija saa ilmoituksen, joka voi olla vaikka jonkun muuttujan arvo. Ilmoitus kuitataan OK-painikkeella, jonka jälkeen ohjelman suorittamisen jatkaminen on mahdollista. Se, mitä ilmoituksessa halutaan näkyvän, kuten painikkeet, kuvake, teksti ja otsikko, on mahdollista määritellä argumentteina. /6, s. 33./

Käyttäjän aikaansaamaa tietoa on puolestaan mahdollista pyytää InputBox-funktion avulla. Tiedon tallennuspaikkana voidaan silloin käyttää String-tyyppistä muuttujaa. Tätä parempi vaihtoehto on kuitenkin Application-objektin InputBox-menetelmä. Sen avulla on nimittäin mahdollista määritellä vastaanotettavan tiedon tyyppi (kuva 6) etukäteen käyttäen Type-argumenttia. Täten sen käyttömahdollisuudet ovat rajattomammat. /6, s. 33, 34./

Value	Meaning
0	A formula
1	A number
2	Text (a string)
4	A logical value (True or False)
8	A cell reference, as a Range object
16	An error value, such as #N/A
64	An array of values

KUVA 6. Vastaanotettavan tiedon tyyppi InputBox-menetelmällä /6, s. 34/

4.2.2 Ehto-ja silmukkarakenteet

Ohjelman suoritusta voidaan muuttaa ehtojen avulla, mitkä mahdollistavat mm. ohjelman haarautumisen tai pysähtymisen. Silmukat puolestaan ovat käyttökelpoisia silloin, kun samaa koodia on tarvetta toistaa siihen asti, kunnes ehto toteutuu. Myös silloin, kun taulukosta pitää käydä läpi kaikki siellä olevat tiedot, on silmukkarakenne hyödyllinen. Sekä ehtojen että silmukoiden toiminta määräytyy lausekkeiden avulla, jotka ovat arvoltaan joko tosi tai epätosi. Ne määräävät suoritettavan koodin osan sekä päättävät sen, milloin silmukasta on tarpeellista poistua. /10, s. 26./

Ohjelma sisältää niin lukuja, merkkijonoja kuin muuttujiakin, joilla kaikilla on omat arvonsa. Useampien arvojen yhdistelmästä voi muodostua laskutoimitus, jota kutsutaan lausekkeeksi. Lauseke voi kuitenkin olla myös yksittäinen arvo itsessään. Vertailemalla lausekkeita keskenään saadaan lopputuloksena totuusarvo, mikä voi olla joko tosi tai epätosi. Vertailussa käytettävät termit ovat pienempi, suurempi, yhtä suuri, pienempi tai yhtä suuri, suurempi tai yhtä suuri sekä eri suuri. Ehtolauseke voi muodostua myös yksittäisestä luvusta tai laskutoimituksesta. Jos luku ei ole nolla, niin silloin ehtolausekkeen arvo on aina tosi. /10, s. 26./

If-rakennetta käytetään silloin, kun ehtoa on kyettävä testaamaan. Testin lopputulos puolestaan määrää sen, mitä ohjelmassa tapahtuu seuraavaksi. /5 s. 34/. If-sana siis aloittaa ehtorakenteen ja se lopetetaan end if-käskyllä. Ehtorakenteen loppuun voidaan

liittää myös Else-lause, jonka jälkeen olevan koodin suorittaminen tapahtuu silloin, jos kaikki edelliset ehdot ovat arvoiltaan epätosia. /10, s. 26, 27./

Silmukkarakenteessa sen toiminta aloitetaan Do-sanalla ja se päätetään Loop-sanaan. Jos alkuun tai loppuun liitetään sana ”While”, niin silloin silmukkaa toistetaan niin kauan kuin ehtolausekkeen arvo on tosi. Jos taas silmukan halutaan jatkuvan niin kauan kunnes arvoksi tulee tosi, käytetään Until-sanaa. Alussa oleva ehto ei kuitenkaan vaadi silmukkaan menemistä, mutta jos ehto on lopussa, niin silloin silmukka käydään vähintään yhden kerran läpi. /10, s. 28./

Silmukan ehtosanaksi valitaan siis joko While tai Until. Ehtosanan valitsemisessa kannattaa yksinkertaisesti vain käyttää maalaisjärkeä miettimällä sanojen merkitystä. Mikäli silmukan alussa tai lopussa ei ole käytetty mitään ehtoa, niin silloin täytyy käyttää Exit Do–komentoa silmukasta poistumiseen. Tämä on välttämätöntä ohjelman toimimisen kannalta. Silmukkaan voi olla myös tarpeellista lisätä DoEvents-komento, jos silmukan suoritus kestää kauan aikaa. Tällä estetään käyttöjärjestelmän mahdollinen jumiutuminen suorituksen aikana. /10, s. 28, 29./

5 OHJELMISTOT JA TIEDONHALLINTAJÄRJESTELMÄT

Tietokoneen voidaan sanoa olevan nykypäivänä ehdoton edellytys suunnittelutyön tekemiselle. Sen avulla työt saadaan tehtyä nopeaan tahtiin ja mahdollisten muutosten tekeminen onnistuu helposti ja vaivattomasti. Esimerkiksi AutoCad on hyvin yleinen ohjelmisto, joka myös Etteplanilla on käytössä. Sen takia kerron siitä tarkemmin seuraavassa kappaleessa. Esittelen myös tiedonhallinnan toimintaperiaatteen, koska sillä oli keskeinen osa opinnäytetyössäni. Kerron myös siitä, kuinka tietokantoja pystytään hyödyntämään sähkösuunnittelussa nopeuttaen työn tekemistä huomattavasti.

5.1 AutoCad

AutoCad voidaan mieltää standardiksi, joka on kehitetty tietokoneavusteisen suunnittelun tueksi. Sen avulla on mahdollista tehdä sähköpiirustuksia sekä muita teknisiä dokumentaatioita. Suunnittelukohteet vaihtelevat pienistä yksittäisistä komponenteista suuriin rakennuksiin. /11./

Yhdysvaltalainen Autodesk on luonut Autocadin, jonka avulla on mahdollista piirtää ja muokata sekä 2D- että 3D-kuvia. Autodesk julkaisi ensimmäisen version AutoCadista vuonna 1982, jonka jälkeen uusia versioita on tullut markkinoille melkein joka vuosi. Autocad on saavuttanut suuren suosion, minkä takia se onkin CAD-ohjelmistoista kaikista käytetyin. /12./

AutoCadiin on siis tehty monia eri versioita, joista yleisimmät ovat eri toimialoille suunnatut Electrical ja Mechanical. Markkinoilta löytyy myös halvempi versio ohjelmasta, AutoCad LT, joka ei ole ominaisuuksiltaan kovin kattava. Tableteille ja mobiililaitteille on myös kehitetty oma versio, AutoCad 360. /12./

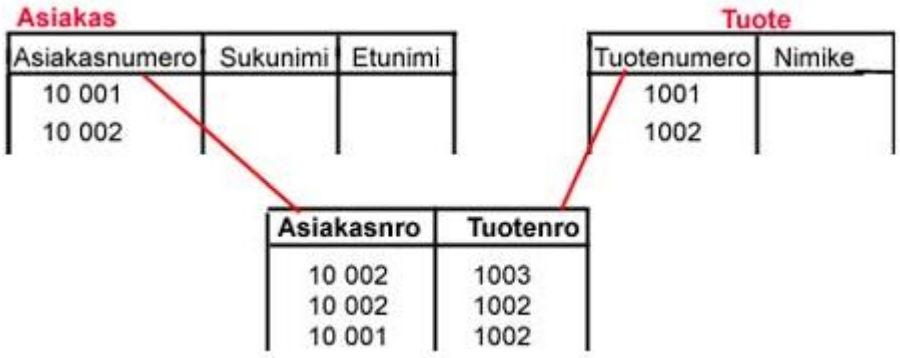
5.2 Tiedonhallinta relaatiotietokannalla

Tiedonkäsittelystä pystytään tekemään huomattavasti yksinkertaisempaa ja helpompaa tietojen varastoinnilla. Ohjelmoijan ei ole tarpeellista muistaa varastoituja tietoja, sillä ne ovat hyvässä tallessa. Tietojenkäsittelyssä joudutaan usein myös keräämään tietoa

uudelleen, mihin kuluu paljon turhaa aikaa. Tiedon varastointi onkin kehitetty sitä varten, että näin ei tarvitsisi menetellä.

Kun puhutaan relaatiotietokannasta, niin siinä monet eri taulukot muodostavat kyseisen tietokannan. Taulukot puolestaan koostuvat riveistä ja sarakkeista, joiden avulla tiedot esitetään. Tietokannan toimintaperiaatteessa tiedot siis jaetaan taulukoihin, joissa yhden tiedon tallennuspaikkana käytetään vain yhtä tiettyä paikkaa. Myös se tieto, miten eri taulukot ovat toisiinsa yhteydessä, tallennetaan relaatiotietokantaan. /13./

Hyvänä ominaisuutena voidaan pitää tietojen helppoa haettavuutta relaatiotietokannasta. Tämä johtuu siitä, että tietoja voidaan käyttää heti sen jälkeen, kun ne on tallennettu tietokantaan. Relaatiolla tarkoitetaan yhteyttä, mikä luodaan eri taulukoiden välille. Sen ansiosta tietojen päivittäminen nopeutuu ja selventyy, koska muutos tehdään ainoastaan yhteen paikkaan. hyvin suunniteltu tietokanta vähentää myös myöhemmin paljon turhan työn määrää, koska kaikkia tietoja ei tarvitse syöttää uudelleen, vaan ne pystytään hakemaan viitatusta taulukosta. /13./



KUVA 7. Asiakastietokanta /13/

Kuva 7 havainnollistaa relaatiotietokannan hyödyllisyyttä sekä kykyä säästää aikaa tiedon muuttuessa. Esimerkissä muutetaan sukunimeä asiakkaan 10 001 kohdalla. Nyt tiedot täytyy muuttaa ainoastaan Asiakas-tauluun. Tiedot olisi täytynyt muuttaa erikseen jokaisen ostetun tuotteen kohdalle, jos käytössä ei olisi ollut relaatioita, vaan pelkkä yksinkertainen luettelo. /13./

5.3 Tietokantojen hyödyntäminen sähkösuunnittelussa

Nykyään sähkösuunnittelutyö tapahtuu käytännössä aina tietokoneen avulla, minkä myötä myös tietokantojen hyödyntäminen sähkösuunnittelussa on lisääntynyt. Täten myös CAD-kuviin lisättyjen objektien sijaintipaikkana toimii tietokanta. Tämä tarkoittaa käytännössä sitä, että kaikki CAD-piirrokset muodostavat kukin oman tietokantansa. Tiedot objekteista saadaan ohjelman avulla tietokannasta ja kyetään sitten lisäämään haluttuun paikkaan kuvaan.

Suunnittelutyön tekemiseen saadaan monia uusia mahdollisuuksia käytettäessä tietokantaa sen tukena. Tietokantoihin pystytään mm. tallentamaan paljon enemmän tietoa verrattuna siihen, että tietoa tallennettaisiin ainoastaan piirustukseen. Tallennettua tietoa on lisäksi mahdollista käyttää jokaisessa sovelluksessa, joka on yhdistetty tietokantaan.

Aikaisemmin kuviin kirjoitettiin informaatiota manuaalisella tavalla, jolloin lisätty tieto oli käyttökelpoinen ainoastaan kyseisessä kuvassa. Esimerkiksi piirikaavioissa esitetään sähkömoottorille ominaiset kilpiarvot aina moottoria kuvaavan piirrosmerkin välittömässä läheisyydessä. Ennen kilpiarvot kirjoitettiin manuaalisesti, mutta nykyään ne lisätään tietokantaan. Täten niitä on mahdollista käyttää muuallakin tarpeen tullen.

6 SÄHKÖSUUNNITTELUN TEHOSTAMINEN

Kaikilla sähkösuunnitteluun erikoistuneilla yrityksillä on jonkin verran toisistaan poikkeavat tavat ja käytännöt suunnittelutyön tekemisessä. Sähkösuunnitelmien lopputulos on kuitenkin aina tietyn standardin mukainen riippumatta suunnittelua tekevästä yrityksestä. Tässä luvussa kerron Etteplanin tämän hetkisestä käytännöstä asiakkaan lähtötietojen selvittämiseen suunnitteluprojektin alussa. Sen jälkeen esittelen uuden käytännön, jonka avulla pystytään tehostamaan suunnittelutyön alkuvaihetta. Esittelen tilaajalle laatimani Excel-taulukon täyttöohjeineen sekä näytän sen, kuinka taulukon sisältämät tiedot saadaan ajettua tietokantaan. Lopuksi kerron vielä ohjelmointityöni eri vaiheista työstäessäni ohjelmakoodia.

6.1 Vanha toimintamalli

Tällä hetkellä Etteplanin toimintamallina on ollut se, että suunnittelutyön alussa on selvitetty lähtötiedot asiakkaalta. Käytännössä se tarkoittaa sitä, että otetaan selvää tulevan suunnitteluprojektin toiminnasta. Tällä viitataan selvityksiin liittyen sähkömoottoreihin, kenttälaitteisiin sekä muihin työn tilaajan toivomuksiin.

Lähtötietoja on saatettu tilanteesta riippuen selvitellä joskus tietynlaisen aloituskokouksen muodossa. Etteplanilla sähkösuunnittelijoiden on täytynyt aina myös omatoimisesti ottaa yhteyttä asiakkaaseen liittyen lähtötietojen selvittämiseen, joiden on pakko olla selvillä ennen kuin suunnittelutyön voi aloittaa tai sitä voidaan jatkaa jonkin työvaiheen jälkeen. Tähän hukkaantuu aina paljon turhaa aikaa, varsinkin jos yhteydenotto tapahtuu väärälle taholle. Asiakas ei nimittäin aina osaa antaa oikeita tietoja, jos kommunikaatio sillä puolen ei ole pelannut vaikka suunnittelukohteen huoltohenkilökunnan tai sen käyttäjien kanssa. Epäselvissä tilanteissa on saatettu saada virheellistä tietoa, mikä on johtanut turhan työn tekemiseen. Tietojen keräämisen jälkeen ne on lisäksi jouduttu aina erikseen työstämään tietokantaan, mikä on vaatinut oman aikansa.

6.2 Asiakkaalle lähetettävä taulukko

Idea uuden toimintamallin luomisesta Etteplanille syntyi siis työnantajani Markku Lakan toimesta. Aluksi kävimme hänen kanssaan yleisesti läpi sitä, millä tavalla suunnitteluprojektien aloitus lähtötietojen keräämisen suhteen on toteutettu. Tulimme siihen tulokseen, että työn tehokkuutta on mahdollista parantaa nykyiseen verrattuna. Uuden toimintamallin avulla asiakas saadaan tekemään työ lähtötietojen keräämisen suhteen. Suunnittelijoiden tarvitsisi enää ainoastaan lähettää tilaajalle laatimani yhtenäinen Excel-taulukko, jonka sisältämät tiedot sitten saataisiin vaivattomasti ajettua tietokantaan saatuani linkin valmiiksi Excelin ja Accessin välille. Excel-taulukosta muodostui seuraavanlainen:

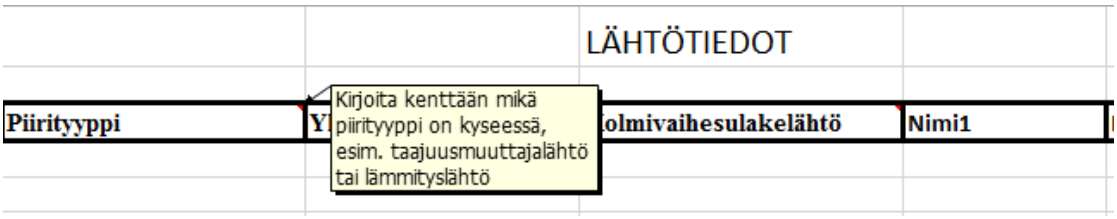
KUVA 8. Asiakkaalle lähetettävä Excel-taulukko

Oikeasti asiakkaalle lähetettävä taulukko (kuva 8) sisältää kaikki otsikkokentät yhdellä rivillä, mutta laitoin ne tähän kahteen riviin tilanpuutteen vuoksi. Ensimmäisen rivin suhteen asiakas kirjoittaa Piirityyppi-kentän alapuolelle, onko kyseessä taajuusmuuttajalähtö, lämmityslähtö, suora lähtö vai suunnanvaihtolähtö. Yksivaihe- sekä kolmivaihesulakelähtökenttiin kirjoitetaan joko ”kyllä” tai ”ei” sen mukaan, kumpi sulakelähtö tarvitaan. Nimi1-kentän alapuolelle tilaaja kirjoittaa mikä laite on kyseessä, esim. pumppu, puhallin tai lämmitin. Nimi2-kenttään tulee sama englannin kielellä. Laitepaikka-kohtaan tulee laitepaikan tunnus, esim. 70PP30. Vastaavasti Keskus-kenttään tulee käytettävän keskuksen tunnus, mikä voi olla vaikka MCC1. Ohjaustapa-kentän

kohdalle asiakas kirjoittaa haluamansa ohjaustavan, mikä voi olla joko paikallisohjaus ilman järjestelmää tai järjestelmän kanssa tai sitten vaihtoehtoisesti etäohjaus.

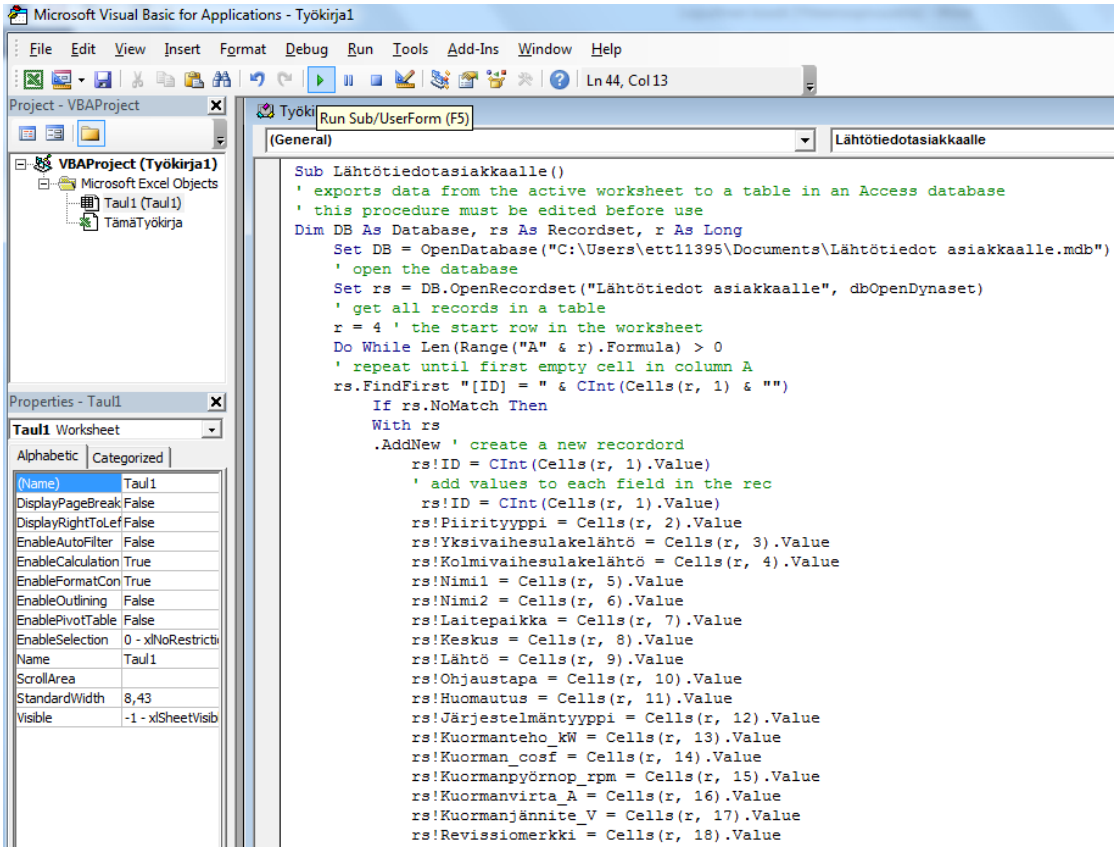
Kuvan toisen rivin tietoihin asiakas voi kirjoittaa Huomautus-kohtaan tarvittaessa, jos taulukon muiden tietojen lisäksi on tarpeellista tuoda jotain informaatiota suunnittelijan tietoon. Järjestelmän tyyppi-kentässä tilaaja ilmoittaa haluamansa järjestelmätyypin valmistajan, mikä voi olla esimerkiksi Metso tai DNA. Kuorman tiedoista asiakas ilmoittaa lukumuodossa tehon (kW), tehokertoimen, pyörimisnopeuden (rpm), virran (A) ja jännitteen (V). Revissiomerkki-kenttään kirjoitetaan tarvittaessa, jos täytyy ilmoittaa jostain muutoksesta.

Excel-taulukko lähetetään asiakkaalle sähköisessä muodossa ja tiedot täytetään siis jokaisen kentän alapuolella olevaan soluun Excelissä. Lisäksi asiakas saa taulukon täyttöohjeet. Ohjeet ovat kenttäkohtaiset jokaisen täytettävän kohdan suhteen, ja ne tulevat näkyviin, kun hiirellä mennään täytettävän kentän kohdalle. Kuva 9 havainnollistaa taulukon täyttöohjetta Piirityyppi-kentän suhteen.

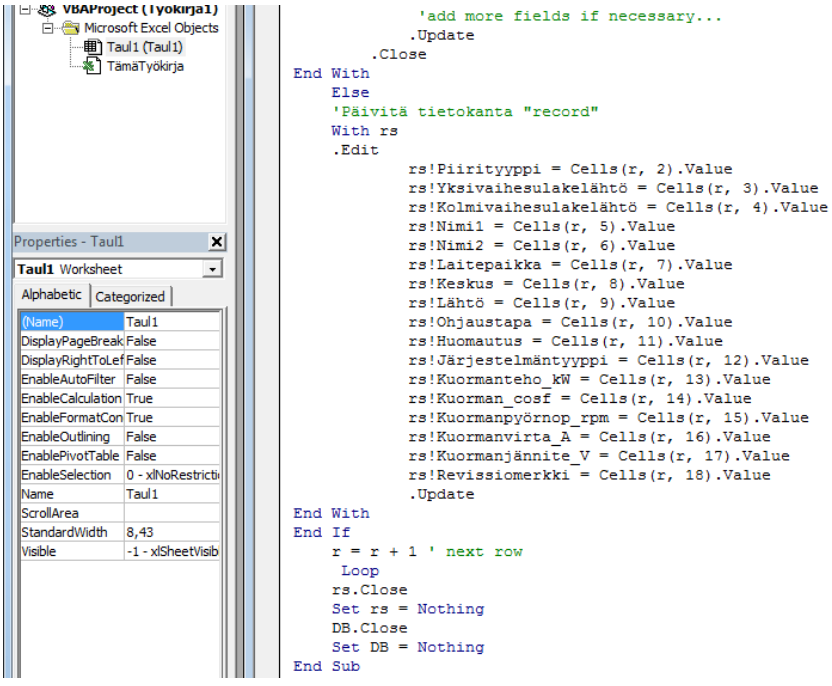


KUVA 9. Esimerkki taulukon kenttäkohtaisesta täyttöohjeesta

6.3 Lähtötietojen siirtäminen tietokantaan



KUVA 10. Tiedonsiirron automatisoinnin alkuosa



KUVA 11. Tiedonsiirron automatisoinnin loppuosa.

Kuva 10 ja kuva 11 havainnollistavat sen, kuinka Excel-taulukon sisältämät lähtötiedot saadaan siirrettyä Access-tietokantaan. Tietojen ajaminen tapahtuu painamalla ylhäällä sijaitsevaa Run Sub-painiketta, kun koodi on saatu kirjoitetuksi Excelin Visual Basic Editoriin. Otin valmiista koodista tilanpuutteen ja selkeyden vuoksi kaksi erillistä kuvaa. Kuvassa 10 on ohjelmakoodin alkuosa ja kuvassa 11 loppuosa.

Nimesin makron yksinkertaisesti käyttäen nimeä ”Lähtötiedotasiakkaalle”. Kirjoitin sanat yhteen, koska välilyöntien käyttäminen on kiellettyä makron nimeämisessä. Koodin vihreällä kirjoitetut tekstit makronauhuri on muodostanut automaattisesti, kun Excelin ja Accessin välillä on tehty toimenpiteitä, eli ne ovat tavallaan Visual Basic Editorin muodostamia kommentteja. Siinä näkyy kommentteina esimerkiksi se, kun tietokanta avataan ja kun nauhoitteeseen luodaan uusia tietueita. Ohjelmaan on mahdollista tehdä kommentteja myös omatoimisesti kirjoittamalla heittomerkin ja sen perään haluamansa kommentin. Sinisellä kirjoitetut tekstin osat ovat kieleen varattuja avainsanoja, jotka kuvaavat niitä toimintoja ja määrittelyitä, joita olen nauhoittamisen jälkeen käyttänyt muokatakseni koodin oikeanlaiseksi. Visual Basic Editorin mustalla tekemäni osat koodissa puolestaan näyttävät ne kohteet, joihin toimenpiteillä sekä määrittelyillä on tarkoitus vaikuttaa.

6.4 Ohjelmointityön vaiheet

Saatuani Excel-taulukon valmiiksi sekä käsityksen tulevasta ohjelmointityöstä, täytyi minun aivan ensiksi perehtyä tietokoneohjelmointiin lukemalla paljon teoriatietoa, mikä käsittelee Excel VBA-ohjelmointia. Siihen piti käyttää paljon aikaa, sillä kyseinen ohjelmointitapa on aiheena hyvin laaja. Internet on onneksi kuitenkin todella kattava tietolähde, joten sieltä löytyi kaikki tarpeellinen teoriatieto asiaan liittyen.

Kun olin käyttänyt omasta mielestäni tarpeeksi aikaa ohjelmointisivujen lukemiseen, aloin soveltaa omaksumaani tietoa käytännössä tekemällä paljon erilaisia yksinkertaisia harjoitustehtäviä liittyen Excel VBA-ohjelmointiin. Tein mm. sellaisia harjoituksia, joissa joistain taulukoista piti saada näkyviin joitain tietoja painamalla tiettyä painiketta tai piti antaa käyttäjälle ilmoitus joidenkin suureiden laskutoimitusten tuloksista.

Kun olin lukenut tarpeeksi teoriatietoa sekä tehnyt harjoituksia Excel VBA-ohjelmointiin liittyen, piti minun alkaa miettiä ohjelmakoodin tekemistä Visual Basic Editoriin. Koska makronauhuriin nauhoitettua sekä lisättyä mahdollisesti muuttuvaa tietoa piti saada siirrettyä tietokantaan, määrittelin Access-tietokannan sekä makronauhoitteen ohjelman muuttujiksi.

Koska muuttujien määrittely tapahtuu aliohjelmassa, käytin niiden nimeämisessä Dim-sanaa. Muuttujien tietotyypin määrityksessä käytin Long-tietotyyppiä, koska se on hyvin yleisesti käytetty ja suosittu tietotyyppi, sillä se mm. vie suhteellisen vähän muistitilaa, ainoastaan neljä tavua. Tietokannan avaamisen sekä tietokanta-objektin esittelyn suoritin Set-lausetta apuna käyttäen. Nauhoitteen luomisessa sekä hakemisessa käytin lisäksi OpenRecordset-menetelmää.

Jotta Excel-taulukon tiedot pystytään ajamaan tietokantaan, täytyy taulukossa olevat tiedot käydä läpi. Täten tulin siihen tulokseen, että silmukkarakenteen käyttäminen olisi hyvin järkevä vaihtoehto tietojen läpikäymiseen. Pidin järkevimpänä vaihtoehtona sitä, että tiedot käydään läpi siinä järjestyksessä, kuin ne ovat Excel-taulukossa vasemmalta oikealle päin mentäessä.

Jotta silmukka toimisi oikealla tavalla, minun piti määritellä sen toiminta lausekkeen avulla. Päätin käyttää siinä ehtolauseketta, jossa silmukka suorittaa kaikki taulukon ensimmäisen rivin solukentät. Lausekkeen vertailussa käytin termiä ”suurempi” kuin nolla, joten lausekkeen ehdot täyttävissä soluissa totuusarvo on aina tosi. Kirjoitin silmukkarakenteen alkuun myös While-ehtosanan, jotta silmukkaa toistettaisiin niin kauan kuin ehtolausekkeen arvo on tosi.

Kun olin saanut silmukkarakenteen kuntoon, minun piti vielä lisätä tietueet, eli Excel-taulukon täytettävät otsikkokentät, jotka silmukan olisi tarkoitus käydä läpi. Tämän päätin suorittaa AddNew-menetelmää apuna käyttäen. Jotta AddNew-menetelmällä lisätyt tietueet saadaan tallennettua nauhoitteeseen sekä ilmaantumaan tietokantaan, käytin Update-menetelmää tietueiden tallentamiseen. Lisättäviä tietueita oli sen verran paljon, että päätin käyttää niiden lisäämisessä With – End With –rakennetta.

Kun ensimmäisen kerran sain monien vaiheiden jälkeen ajettua Excel-taulukon sisältämät tiedot tietokantaan, niin ongelmana oli, että tiedot eivät menneet oikeaan paikkaan Accessiin. Niitä ei aluksi löytynyt sieltä ollenkaan, vaikka Run Sub-käskyn suorittaminen oli sujunut ongelmitta. Sain ongelman ratkaistua lisäämällä ylimääräisen tietueen heti silmukkarakenteen alkuun, jonka avulla identifioiminen onnistuu tietokannan puolella. Nimesin uuden tietueen yksinkertaisesti ID-nimiseksi.

ID:n luomisen jälkeen sain tiedot automatisoitua Accessin puolelle kuitenkin ainoastaan ensimmäisellä ajolla oikeaan kohtaan. Jos nimittäin tein Excel-taulukkoon vaikka jotain muutoksia ja käytin sen jälkeen uudestaan Run Sub-käskyä, uudet päivitetyt tiedot ilmaantuivat vanhojen tietojen alapuolelle. Vanhentuneet tiedot eivät myöskään hävinneet, vaan ne olivat edelleen näkyvissä, mikä olisi täysin turhaa.

Ratkaisin ongelman käyttämällä FindFirst-menetelmää, jonka avulla tiedot päivittyvät aina oikeaan kohtaan Accessiin, heti ensimmäiseltä riviltä alkaen. Sen lisäksi minun piti myös määritellä AddNew-menetelmän toiminta if-rakennetta apuna käyttäen, johon on liitettynä NoMatch-ominaisuus. AddNew-menetelmää siis käytetään, kun NoMatch-ominaisuuden arvoksi tulee tosi. Näin tapahtuu siis aina, kun FindFirst-menetelmän toiminto on epäonnistunut.

Lopputyössäni minun piti myös lisätä kaikki tietueet toiseen kertaan Else-lauseen perään samaan silmukkarakenteeseen, jotta pelkästään tietojen päivittäminen onnistuu, kun mitään uusia tietueita ei lisätä. Else-lauseen jälkeen olevan koodin suorittaminen tapahtuu siis silloin, kun AddNew-menetelmää ei käytetä NoMatch-ominaisuuden arvon ollessa epätosi. Tällöin siis FindFirst-menetelmän toiminto on onnistunut.

7 LOPPUPOHDINNAT

Kun aloin tekemään opinnäytetyötäni, niin aivan ensiksi tutustuin Etteplanilla käytössä oleviin ohjelmistoihin ja otin selvää niiden toiminnasta. Otin myös yleisesti selvää työnteon tavoista sekä käytännöistä. Tutustuin myös tietysti Accessin toimintaperiaatteeseen, sekä myös Etteplanilla käytössä olevaan Elsaan, mikä on kehitetty Accessin tueksi ja se myös käyttää Accessia tietojen tallentamiseen. Perehdyin myös Etteprohon, jonka avulla CAD-kuviin pystytään generoimaan tietoja kaapeleista sekä kojeista.

Työn tekeminen oli hyvin haasteellista, koska minulla ei ollut minkäänlaista aikaisempaa kokemusta tietokoneohjelmoinnista. En ollut aikaisemmin myöskään perehtynyt tietokantoihin tai tiedonhallinnan toimintaperiaatteeseen millään tavalla, joten nekin asiat täytyi opetella alusta alkaen. Minun täytyi käyttää jonkin verran aikaa myös pelkästään Excelin käytön opetteluun. Excelin toiminta oli nimittäin aika hyvin päässyt unohtumaan, sillä en ollut käyttänyt sitä kunnolla moneen vuoteen.

Suunnitteluohjelmistojen lisääntyessä ja kehittyessä tullaan myös tietokantoja käyttämään tulevaisuudessa yhä vain enemmän sähkösuunnittelutyön tukena. Ohjelmistoihin suunnitellaankin tietokannan kanssa yhteistyössä toimivia uusia ominaisuuksia. Sen ansiosta pystytään vähentämään turhan työn määrää manuaalisten kirjoitusvaiheiden jäädessä pois. Täten siis uskon, että omaksumistani taidoista sekä tiedoista on minulle varmasti hyötyä myös tulevaisuuden työelämässä.

LÄHTEET

/1/ Etteplan Oyj. Yrityksen www-sivut. http://www.etteplan.com/about-etteplan.aspx?sc_lang=fi-FI. Luettu 25.2.2014. Päivitetty 25.2.2014

/2/ Jyväskylän yliopisto. Johdatus ohjelmointiin. Www-sivut. http://www.mit.jyu.fi/opetus/Ciao/ciao003.htm. Luettu 26.2.2014. Päivitetty 1.8.2000

/3/ Kerkkänen Jukka-Pekka. Opinnot.net. Www-sivut. http://opinnot.net/tietotekniikka/ohjelmointi/index.php. Luettu 26.2.2014. Päivitetty 13.9.2007.

/4/ Medianurkka. Ohjelmointi. Www-sivut. http://ohjelmointi.medianurkka.com/?p=59. Ei päivitystietoa. Luettu 28.2.2014.

/5/ Hwechtla-tl. Sange.fi. Www-sivut. http://sange.fi/~atehwa/cgi-bin/piki.cgi/olio-ohjelmoinnin%20pointti. Luettu 3.3.2014. Päivitetty 20.4.2013.

/6/ Haaga-helia. Excel VBA-ohjelmointi. PDF-dokumentti. http://myy.haaga-helia.fi/~taaak/vba/vba.pdf. Luettu 3.3.2014. Päivitetty 27.9.2013.

/7/ Ornanet. Excel-ohjelmoinnin sivusto. Www-sivut. http://ornanet.dy.fi/makrotpaasivu.htm. Luettu 5.3.2014. Päivitetty 18.5.2012.

/8/ Salminen Pertti. Edistynyt Excel. Www-dokumentti. http://www.netikka.net/epj/Excel/excel.html. Ei päivitystietoa. Luettu 6.3.2014.

/9/ Ornanet. Excel-ohjelmoinnin sivusto. www-sivut. http://ornanet.dy.fi/koodintuottaminen.htm. Päivitetty 18.5.2012. Luettu 6.3.2014.

/10/ Vesa Lauri. Ohjelmoinnin perusteet. PDF-dokumentti. http://ta.ramk.fi/~ismo.sarajarvi/PATI2007/opintojaksot/4211A/RAMK_VB_opas.pdf. Päivitetty 20.1.2007.Luettu 10.3.2014.

/11/ Lemmetty Sami. AutoCad. Www-sivut. http://piisami.net/tieto/autocad.htm. Päivitetty 10.9.2008. Luettu 3.4.2014.

/12/ Autodesk Oy. Yrityksen www-sivut. http://www.autodesk.fi/products. Päivitetty 3.4.2014. Luettu 3.4.2014.

/13/ Sarja Jari. Verkkopedagogi. WWW-dokumentti. http://verkkopedagogi.net/vanhat/fi/sisalto/materiaalit/access2003/luku0375c6.html?C:D=419702&selres=419702. Päivitetty 2006. Luettu 4.4.2014.