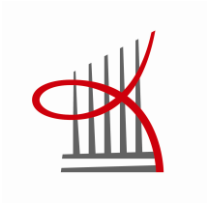


CADS-generointiominaisuuksien kehittäminen

Mikko Konttinen

Opinnäytetyö
Huhtikuu 2014
Sähkötekniikka
Automaatiotekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Sähkötekniikka
Automaatiotekniikka

KONTTINEN, MIKKO:
CADS-generointiominaisuuksien kehittäminen

Opinnäytetyö 63 sivua, joista liitteitä 8 sivua
Huhtikuu 2014

Opinnäytetyön tarkoituksena oli tuottaa Excel-pohjainen työkalu, jota hyödynnetään CADS Planner -suunnitteluohjelmiston version 16 generointintiominaisuuksissa. CADS-generoinnin avulla pystytään luomaan tietokannasta ja piirikaaviopohjista valmiita piirikaavioita. Työkalun avulla opinnäytetyön tilaajayrityksen Insta Automation Oy:n suunnittelutoimintaa saadaan tehostettua.

Tilaajayrityksessä työskentelevien suunnittelijoiden aiemmin tekemissä generointikokeiluissa oli havaittu ongelmia tiedonhaussa. Ongelmat liittyivät aiemmin käytössä olleeseen tilaajayrityksen tuottamaan työkaluun. Tämä työkalu yhdistää tiedot tietolähdetaulukoista generointia varten yhdeksi taulukoksi. Ongelmat olivat korjattavissa Excel-funktioiden ja Visual Basic for Applications -ohjelmakoodin avulla.

Työn alussa oli tarkoitus tutustua CADS-generoinnin toimintaan ja tilaajayrityksessä sovittuihin generointiin liittyviin suunnittelutapoihin. Lisäksi tuli tutustua aiemmin käytössä olleen työkalun toimintaan ja Visual Basic for Applications -ohjelmointikieleen. Näiden perusteella tuli löytää tarvittavat kehitystarpeet työkalua varten. Tavoitteena oli, että mahdollisimman suuri osa tietokannan tärkeimmistä tiedoista haettaisiin automaattisesti ja työkalun käyttö olisi selkeää.

Työssä kehitetyn työkalun avulla on mahdollista hakea tietolähdetaulukoista tietokantaan 32 eri tietoa automaattisesti. Käsin syötettäväksi jää noin 100 tietoa, mutta näistä vain noin kolmasosa on yleisesti käytössä. Työkalu sisältää myös muita tiedonkäsittelyyn liittyviä aputoimintoja. Työkalun toimintaa testattiin erityyppisillä suunnitteluprojekteilla, joiden avulla varmistuttiin siitä, että sen toiminta on luotettavaa ja tehokasta. Työkalun tiedonhakunopeus on vähintään 0,57 sekuntia per rivi ison suunnitteluprojektin suuruisella tietolähdetaulukkomäärällä. Tiedonhakunopeus on kuitenkin riippuvainen PC-laitteiston tehokkuudesta ja tietolähdetaulukoiden määrästä.

Suurin osa opinnäytetyölle asetetuista tavoitteista saavutettiin: joitain toivottuja toimintoja ei ehditty toteuttamaan rajallisen ajan ja monimutkaisen toteutustavan vuoksi. Nämä toiminnot lisätään työkaluun mahdollisesti myöhemmin. Työkalu otetaan käyttöön tulevaisuudessa tilaajayrityksen suunnittelutoiminnassa.

Opinnäytetyö sisältää Insta Automation Oy:lle kuuluvia ratkaisuja. Tästä syystä työ on julistettu salaiseksi viiden vuoden ajaksi julkaisuhetkestä.

Asiasanat: sähkö, automaatio , suunnittelu , ohjelmointi, VBA

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

KONTTINEN, MIKKO:
Development of CADS Generating Features

Bachelor's thesis 63 pages, appendices 8 pages
April 2014

The purpose of the thesis was to produce an Excel-based tool to be used in the generating features of CADS Planner software, version 16. CADS generating features allow creating the circuit diagrams by uniting database and circuit diagram templates. With the help of the tool the client company Insta Automation Oy can improve its performance in engineering.

The engineers of the client company had observed some problems in data import in their previous generating experimentation. These problems related to the tool that was used previously. This tool unites the information from the data source tables for generating into a single table. The problems were fixable with the help of Excel functions and Visual Basic for Application programming code.

In the beginning of the thesis, the purpose was to become familiar with the CADS generating features and the methods of generating engineering in the client company. In addition, the intention was to become familiar with the functions of the tool previous used, as well as with the programming language of Visual Basic for Applications. On the basis of this, the needs for developing the tool were discovered. The objective was that the main part of the most important data would be imported automatically. It was also essential that the use of the tool would be simple.

The developed tool makes it possible to import 32 pieces of information automatically, from the data source tables to the database. About 100 pieces of information still remain to be fed manually, but only about one third of these are in active use. The tool also has other functions to help the handling of data. The tool was tested in different types of engineering projects, which confirmed the reliability and efficiency of the tool. With the number of data source diagrams in a big engineering project, the speed of importing data is about 0.57 seconds per line. However, the speed depends on the capacity of the PC equipment and the number of data source tables.

The goals were achieved even though there was not enough time to accomplish all the functions desired. These functions may be added to the tool outside of the thesis process. The tool will be implemented for the business operations of the client company.

The thesis includes solutions that belong to Insta Automation Oy, which is why the work will be classified for five years since the date of publication.

Key words: electricity, automation, engineering, programming, VBA

SISÄLLYS

LYHENTEET JA TERMIT

VBA	Visual Basic for Applications- ohjelmointikieli
IFC	Rakennusalan ISO/PAS 16739 standardi oliopohjaisen tiedon siirtoon tietokonejärjestelmästä toiseen.
Työkirja	Excel-tiedosto, joka sisältää taulukoita.
Salattu termi	*Tämä termi on asetettu salaiseksi*
Excel-funktio	Apuväline tiedonkäsittelyyn Excel-taulukossa.
PHAKU	Excel-funktio, joka hakee annetun arvon matriisin ensimmäisestä sarakkeesta ja palauttaa matriisin toisen sarakkeen samalla rivillä olevan arvon.

1 JOHDANTO

1.1 Työn tarkoitus

Opinnäytetyö on tehty kehitystyönä Insta Automation Oy:lle. Sen tarkoituksena oli tuottaa tilaajayritykselle Excel-pohjainen työkalu, jota hyödynnetään CADS Planner -suunnitteluohjelmiston version 16 generointintiominaisuuksissa. Työkalun avulla opinnäytetyön tilaajayrityksen suunnittelutoimintaa saadaan tehostettua.

CADS-generoinnin avulla pystytään luomaan tietokannasta ja piirikaaviopohjista valmiita piirikaavioita. Tästä on hyötyä etenkin suuremmissa suunnitteluprojekteissa, jotka sisältävät useita rakenteeltaan samanlaisia piirikaavioita. Jokaista piirikaaviota ei tällöin tarvitse erikseen piirtää ja säästetään huomattavasti aikaa suunnittelussa.

Tilaajayrityksessä työskentelevien suunnittelijoiden aiemmin tekemissä generointikokeiluissa oli havaittu ongelmia tiedonhaussa. Ongelmat liittyivät aiemmin käytössä olleeseen tilaajayrityksen tuottamaan generointityökaluun. Tämä työkalu yhdistää tiedot tietolähdetaulukoista generointia varten yhdeksi taulukoksi. Ongelmat olivat korjattavissa Excel-funktioiden ja Visual Basic for Applications -ohjelmakoodin avulla.

1.2 Tavoitteet

Työn alussa oli tarkoitus tutustua CADS-generoinnin toimintaan ja tilaajayrityksessä sovittuihin generointiin liittyviin suunnittelutapoihin. Lisäksi tuli tutustua aiemmin käytössä olleen generointityökalun toimintaan sekä Visual Basic for Applications -ohjelmointikieleen. Näiden perusteella tuli löytää tarvittavat kehitystarpeet työkalua varten.

Työn päätavoitteet olivat seuraavat:

- Löytää ongelmakohdat aiemmin käytetyn generointityökalun tiedonhausta.
- Ratkaista ongelmat Excel-funktioiden ja Visual Basic for Applications -ohjelmakoodin avulla.

- Kehittää tiedonhakutoimintoja niin, että mahdollisimman pieni osa tiedoista jäisi käsin täytettäväksi.
- Tiedustella toiveita generointityökalun käyttäjiltä työkalun kehitykseen ja toimintoihin, sekä ottaa nämä huomioon kehityksessä.
- Kehittää työkalun käyttö mahdollisimman selkeäksi.
- Tuottaa työkalu, joka toimii mahdollisimman monessa erilaisessa suunnitteluprojektissa.
- Testata ja varmistaa työkalun toiminta luotettavaksi.
- Hyödyntää työkalua johonkin tilaajayrityksessä meneillään olevaan suunnitteluprojektiin.

Opinnäytetyön kehitystyöluonteen vuoksi sen sisältö muuttui hieman työn aikana. Jotta työn sisältö ei paisuisi liian suureksi, tehtiin työn loppupuolella rajaus, mitä toimintoja työkalu tulisi sisältämään. Työn aikana syntyneet välitavoitteet on esitetty generointityökalun ongelmien kartoituksen (kappale 5.1.3) ja suunnittelun (kappale 5.2) yhteydessä.

1.3 Sisältö

Opinnäytetyö käsittelee yleisesti, mitä on CADS-generointi, sekä antaa perustiedot Visual Basic for Applications -ohjelmointiin. Näiden pohjalta se esittelee työn tuloksena saadun työkalun toimintaa ja käyttöä. Opinnäytetyössä on myös esitelty, miten valmista työkalua testattiin ja miten sitä voitaisiin vielä kehittää.

2 INSTA AUTOMATION OY

Insta Automation Oy on osa Insta Group Oy (entinen Instrumentointi Oy) konsernia, joka on perustettu vuonna 1960. Insta Group Oy konserniin kuuluvat puolustus- ja turvallisuusteknologiaa kehittävä Insta DefSec Oy, ilmailun elinkaaripalveluita tuottava Insta ILS Oy, virtuaaliteknologioihin ja innovatiivisiin palveluihin keskittyvä Insta Innovation Oy sekä teollisuusautomaatioon erikoistunut Insta Automation Oy. Yhtiön tärkeimpiä asiakkaita ovat esimerkiksi prosessi- ja energiateollisuus, materiaalin käsittely, vesilaitokset, näyttämöt, puolustusvoimat, valtionhallinto sekä turvallisuusorientoituneet organisaatiot. (Insta Group Oy 2014.)

Insta Automation Oy on erikoistunut teollisuuden ja prosessien sähköautomaation suunnittelu-, valmistus-, asennus- ja ylläpitopalveluihin sekä kokonaistoimituksina että erillisinä palveluina. Yritys koostuu useasta osastosta, joihin kuuluvat projektisuunnittelu-, ylläpito-, kokonaistoimitukset-, valmistus- ja asennusosasto. Yrityksen suurimpia asiakasryhmiä ovat muun muassa:

- Prosessiteollisuus
- Energian tuotanto
- Elintarviketeollisuus ja meijerit
- Vesilaitokset
- Näyttämöautomaation käyttäjät

(Insta Group Oy 2014)

Insta Group Oy konsernin liikevaihto vuonna 2012 oli 84,4 miljoonaa euroa. Insta Automation Oy:n liikevaihto taas vuonna 2012 oli 47,9 miljoonaa euroa. Instan toimipisteet sijaitsevat Tampereella, Vantaalla, Helsingissä, Muuramessa, Oulussa, Harjavallassa, Porissa ja Varkaudessa. Henkilöstöä on kaikkiaan noin 750 ja Insta Automationilla noin 400. (Insta Group Oy 2014.)

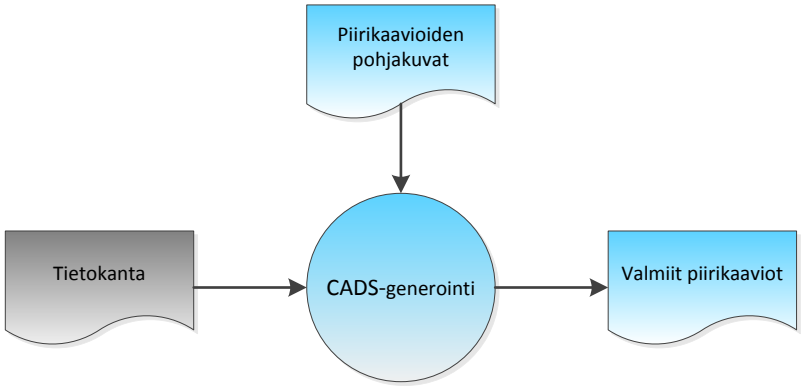
3 CADS PLANNER

3.1 Yleistä

CADS, tai viralliselta nimeltään CADS Planner, on vuonna 1979 perustetun suomalaisen Kymdata Oy:n kehittämä CAD-suunnitteluohjelmisto. Se on yksi Suomen käytetyimmistä suunnitteluohjelmistoista sähkö- ja LVI-suunnittelutoimistoissa sekä sähköurakointiyrityksissä. Suunnittelun lisäksi se sisältää myös ominaisuuksia dokumentointia varten. CADS Planneriin on saatavilla erilaisia alakohtaisia versioita, joita ovat esimerkiksi rakennussuunnitteluun tarkoitettu House, sähkösuunnitteluun tarkoitettu Electric sekä LVI- ja automaatiosuunnitteluun tarkoitettu Hepac. Jokainen versio sisältää alan piirustuksiin liittyviä alisovelluksia. Ohjelmisto on IFC-yhteensopiva (Industry Foundation Classes), joka mahdollistaa muilla IFC-standardin mukaisilla ohjelmistoilla luotujen tiedostojen avaamisen. Myös CADS Plannerilla luodut tiedostot on mahdollista avata muilla IFC-standardin mukaisilla ohjelmistoilla, mikäli tiedostotyyppi on yhteensopiva. (Kymdata Oy 2014a.)

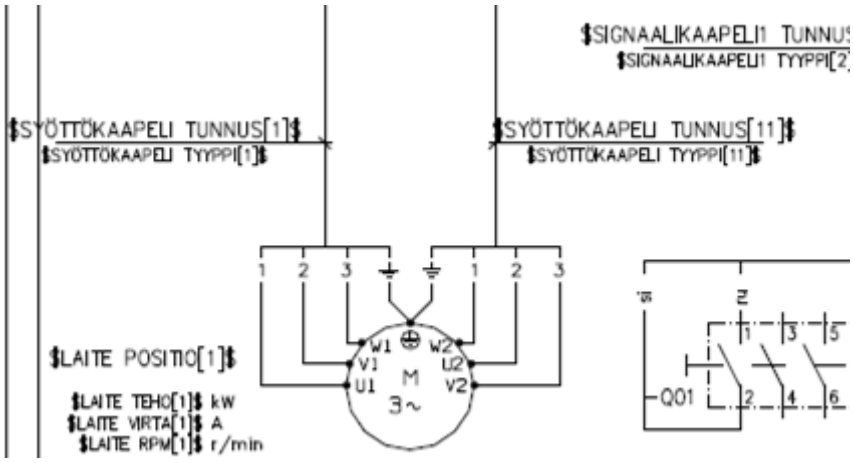
3.2 Generointiominaisuudet

CADS-generoinnin avulla pystytään luomaan tietokannasta ja piirikaaviopohjista valmiita piirikaavioita (KAAVIO 1). Tästä on hyötyä etenkin suuremmissa suunnitteluprojekteissa, jotka sisältävät useita rakenteeltaan samanlaisia piirikaavioita. Jokaista piirikaaviota ei tällöin tarvitse erikseen piirtää, ja säästetään huomattavasti aikaa suunnittelussa. Generointiominaisuus löytyy CADS Planner Electric -version piirikaaviosovelluksesta. CADS 16 -versioon on tullut mukaan ominaisuus, jonka avulla generointitietokantana voidaan käyttää Excel-työkirjaa. Piirikaaviopohjana taas voidaan käyttää mitä tahansa IFC-standardin mukaisella CAD-ohjelmistolla toteutettua dokumenttia. Pääasiana kuitenkin on, että pohjakuvaan on määritetty sarakeviittaukset tietokantataulukkoon tietokantalinkkien avulla. (Kymdata Oy 2014a.)



KAAVIO 1. Piirikaavioiden generointiprosessi.

Pohjakuva on muuten valmis piirikaavio, mutta sen nimiö-, laite-, liitin- ja kaapelitietoja ei ole täytetty. Tietoja varten pohjakuvassa on attribuutit jokaisessa piirikaavion symbolissa sekä nimiössä. CADS-generointi hakee tiedot tietokantataulukosta ja syöttää ne attribuutteihin. Attribuutit sekä tietokantataulukon sarakkeet ja rivit linkitetään toisiinsa tietokantalinkkien avulla. (Kymdata Oy 2014b.) Kuvassa 1 on esitetty esimerkki pohjakuvan attribuutteihin määritetyistä tietokantalinkeistä.



KUVA 1. Piirikaaviopohjan tietokantalinkit.

Kuten kuvasta 1 nähdään, on tietokantalinkki rajattu $-merkeillä molemmin puolin. Näiden merkkien sisäpuolella vasemmalla on tietokantataulukon tietosarakkeen nimi. Tietosarakkeen vieressä hakasulkeiden sisällä sijaitsee tietokantarivinumero.

Tietokantataulukko tulee koostua siten, että kaikki data sijaitsee Excel-työkirjan ensimmäisellä taulukkosivulla. Sen ensimmäisellä rivillä tulee olla pohjakuvien tietokantalinkkejä vastaavat nimet, joiden perusteella tiedetään, mistä sarakkeesta mikäkin tieto haetaan. Sarakkeiden käyttö voidaan toteuttaa muuten vapaasti, mutta niiden määritystä

varten on kolme vakiosaraketta. Pohjakuvasarake kertoo, mitä pohjakuvaa rivin generoinnissa käytetään. Tämä on ainoa generoinnin kannalta pakollinen sarake tietokannassa. Pohjakuvasarakkeen lisäksi voidaan käyttää kuvatiedostosaraketta, joka kertoo, minkä niminen kuvatiedosto rivistä luodaan. Kuvatiedostosarake voidaan jättää myös pois taulukosta, jolloin generoidaan ainoastaan yksi, useampisivuinen kuvatiedosto. Tälle kuvatiedostolle kysytään nimi kuvaa generoitaessa. Kolmantena vakiosarakkeena on tietokantarivinumero. Se on numeerinen määrityskenttä laitteen numerolle, jonka avulla voidaan käyttää useamman taulukkorivin tietoja samassa kuvatiedostossa. Jos tämän sarakkeen arvo on esimerkiksi ”1” ja tietosarakkeen nimenä ”tunnus”, korvataan pohjakuvasta attribuutti, jonka tietokantalinkki on *$tunnus[1]$*. (Kymdata Oy 2014b.)

4 VISUAL BASIC FOR APPLICATIONS

4.1 Yleistä

Visual Basic for Application eli VBA pohjautuu Microsoftin kehittämään Visual Basic-ohjelmointikielen (VB). Sen avulla voidaan automatisoida ja laajentaa Microsoft Office-ohjelmien toimintaa, joita ovat esimerkiksi Excel, Word ja PowerPoint. Näistä jokin toimii niin sanottuna isäntäsovelluksena, jonka päällä VBA-koodi suoritetaan. Sovellusta ohjaavia ohjelmia kutsutaan makroiksi. VBA:n avulla ei voida luoda itsestään suoritettavia ohjelmia. Koodia suorittavasta sovelluksesta voidaan kuitenkin ohjata toista sovellusta. Esimerkiksi Excel-taulukon sisältämistä tiedoista voidaan luoda raportti Word-dokumenttina. VBA on kehitetty nopeuttamaan ja helpottamaan työskentelyä Office-sovellusohjelmilla. VBA-koodi tallennetaan Office-tiedostoon mukaan, eli esimerkiksi Excel VBA -projekti tallentuu Excel-työkirjaan. Tällainen tiedosto on nimeltään makrotyökirja. Kun Excel-makrotyökirjaa siirretään sijainnista toiseen, koodi kulkee koko ajan työkirjan mukana. (Merensalmi 2007, 4-5.)

VBA-ohjelma voidaan luoda yksinkertaisimmillaan nauhoittamalla tarvittavat toimenpiteen makronauhurin avulla. Makronauhuri kirjoittaa tehdyt toimenpiteet VBA -koodiksi. Tällä tavalla voidaan kuitenkin tehdä vain yksinkertaisia toimenpiteitä. Varsinainen hyöty VBA -ohjelmoinnista saadaankin vasta itse kirjoitetulla ohjelmakoodilla. (Taanila 2013, 1.)

Makron suorittamista varten dokumenttiin voidaan lisätä esimerkiksi painike tai luoda sille oma pikanäppäin. Se on myös mahdollista suorittaa automaattisesti, kun tietty toimenpide tehdään isäntäsovelluksessa.

VBA-kielen opettelu ei ole yhtä vaikeaa kuin esimerkiksi C++- tai java-kielen opettelu, sillä se on nimensäkin mukaisesti visuaalista. Kohtuullisten englannin kielen taitojen ja päättelyn avulla ohjelman toiminnallisuudesta pääsee ymmärrykseen. Perustoiminnot on helppo opetella nauhoittamalla toimenpiteet makronauhurilla ja tutkimalla tämän perusteella luotua koodia. VBA-editori sisältää myös objektikirjaston, josta voi tarkastella kaikkia VBA-kielen luokkia, moduuleita, tietotyyppejä ja vakioita. Objektikirjastosta näkee myös ohjeet edellä mainituille asioille.

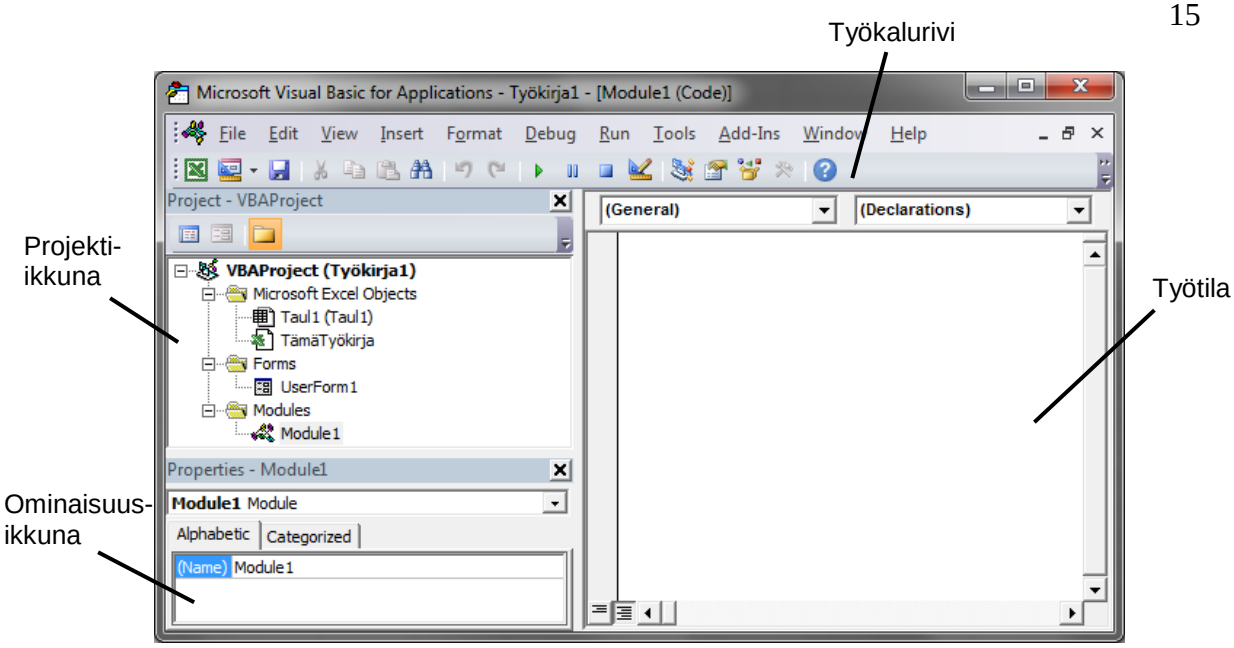
Suurin hyöty VBA-ohjelmoinnista saadaan, kun käsiteltävänä on suuri määrä tietoa. Jos esimerkiksi halutaan lihavoida vain joka toinen fontti monia tuhansia rivejä sisältävälle Excel-taulukolle, on tämä ihmisen tekemänä todella hidasta, ellei jopa mahdotonta. VBA-koodilla luodun makron avulla kyseinen toimenpide on mahdollista tehdä muutamissa sekunneissa.

VBA-ohjelmointi tuo kuitenkin eteen myös haasteita. Excelin asetukset saattavat vaikuttaa ohjelman suoritukseen siten, että kahdella eri tietokoneella ohjelma toimii eri tavalla. Esimerkiksi maa-asetukset saattavat sekoittaa ohjelman suoritusta päivämäärien ja desimaalierottimien eri esitystapojen vuoksi. Päivämäärät voidaan esittää esimerkiksi *3.4.2014* tai *4/3/2014*, ja desimaalierottimena voidaan käyttää joko pilkkua tai pistettä. Ohjelmakoodin kirjoittaja ei siis voi olettaa, että ohjelmaa käytettäisiin aina samoilla asetuksilla, jotka hänen tietokoneellaan ovat voimassa. Hänen pitää varautua kaikkiin mahdollisiin tilanteisiin tai keksiä jokin toinen ratkaisu. (Merensalmi 2007, 7.)

4.2 Excel VBA

4.2.1 Visual Basic -editori

Ohjelmakoodia käsitellään erillisessä Visual Basic -editorissa. Editorin ulkoasu on samanlainen riippumatta sovellusohjelmasta tai -versiosta. Editorissa määritellään projektissa käytettävät ohjelmakoodit, lomakkeet ja muut tarvittavat komponentit. Se tarjoaa myös apuvälineitä koodin hallintaan. Jotta Excelin kehitystyökaluja (joihin VBA -editori mukautuu) päästään käyttämään, tulee kehitystyökalut olla aktivoituna. (Merensalmi 2007, 19-21.)



KUVA 2. VBA -editori.

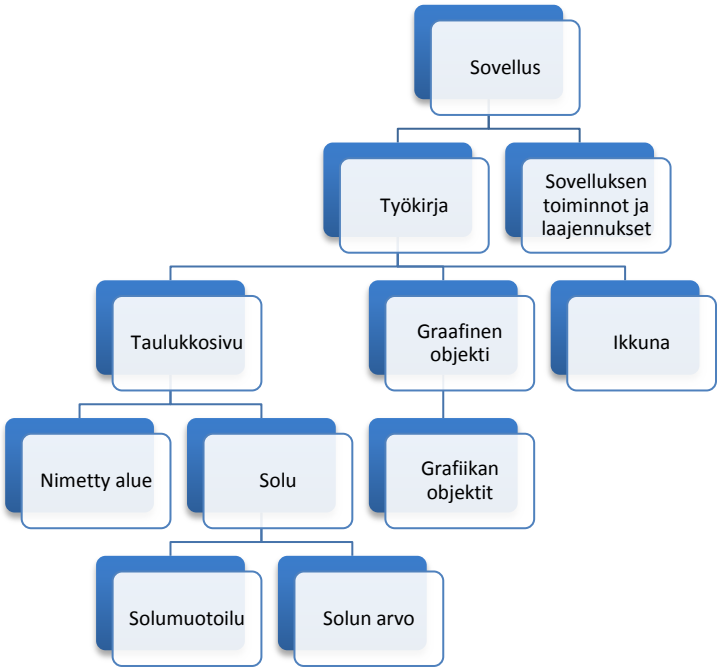
VBA-editori (KUVA 2) on englanninkielinen riippumatta Officen kieliasetuksista. Työkaluriviltä löytyy pikanäppäimet kaikkiin yleisimmin käytettyihin toimintoihin, kuten tallentamiseen, ohjelman ajamiseen, asetuksiin ja objektikirjastoon. Projekti-ikkunassa on listattu kaikki projektiin liitetyt komponentit. Kuvassa 2 olevassa projektissa on Excel-työkirjan lisäksi yksi käyttäjälomake ja moduuli. Ohjelmakoodi kirjoitetaan pääasiassa moduuleihin. Jos kuitenkin esimerkiksi taulukkosivu sisältää painikkeita tai valintoja, voidaan näitä varten kirjoittaa toimintoja itse taulukkosivuun. Sama pätee myös koko työkirjaan ja käyttäjälomakkeisiin. Kaikkia projekti-ikkunan komponentteja on mahdollista lisätä käyttäjän tarpeiden mukaisesti. Ominaisuudet-ikkunassa näytetään valitun komponentin ominaisuudet. Esimerkiksi moduulin voi uudelleen nimetä kirjoittamalla uuden nimen ominaisuusikkunassa. Työtilaan kirjoitetaan ohjelmakoodi ja luodaan käyttäjälomakkeet. Editori tekee automaattisesti erotinviivan aliohjelmien väliin. (Merensalmi 2007, 19-21.)

Koodi voidaan testiajaa työkalurivin Run-painikkeella. Jotta koodi voidaan ajaa, tekee VBA -editori syntaksitarkastuksen koodille. Toisin sanoen se tarkistaa koodin ulkoasun oikeellisuuden. Se ilmoittaa, mikäli esimerkiksi jokin komento on puutteellinen tai sisältää kirjoitusvirheitä. Jos itse ohjelman suorituksen aikana tulee virheitä, antaa ajo virheilmoituksen. Virheilmoituksessa on mukana virheen tunnusnumero ja tämä kuvausteksti. Virheilmoituksesta voidaan joko lopettaa ohjelman ajo tai Debug-painiketta painamalla nähdä koodin rivi, jossa virhe ilmaantui. Koodia voidaan ajaa myös rivi kerrallaan F8-näppäimen avulla. Tästä on hyötyä, kun halutaan nähdä tarkemmin, miten

ohjelma suoritetaan. Koodin suoritus voidaan myös keskeyttää tietylle riville tekemällä keskeytysmerkintä. Tällöin ennen keskeytystä oleva koodi voidaan suorittaa Run-painikkeella, ja keskeytysmerkinnän kohdasta eteenpäin voidaan jatkaa joko askel kerrallaan tai Run-painikkeella.

4.2.2 VBA-projektin tietorakenne

Excel VBA:n tietorakenne on hierarkinen. Kun viitataan rakenteeseen, puhutaan yleensä objekteista. Esimerkiksi itse Excelin makrotyökirja on yksi objekti. Makrotyökirja taas koostuu muista objekteista, joita ovat esimerkiksi taulukkosivut. Tietorakenteeseen kuvataan kaikki Excelin työkirjapuolella olevat objektit. Tämän lisäksi rakenteesta löytyy sellaisia objekteja, joita työkirjapuolella ei ole käytettävissä. Kaaviossa 2 on esitetty yleisimmät tavallisessa VBA-projektissa tarvittavat objektit. (Merensalmi 2007, 41.)



KAAVIO 2. Excel VBA:n tietorakennehierarkia (Merensalmi 2007, 41.)

Ylimmällä tasolla viitataan Exceliin eli sovellukseen (*Application*). Sovelluksen alla olevilla objekteilla viitataan sovelluksen sisällä auki oleviin ikkunoihin (*ActiveWindow* ja *Windows*) tai työkirjoihin (*ActiveWorkbook*, *ThisWorkbook* ja *Workbooks*). Työkirjojen alla voidaan viitata taulukkosivuihin (*Sheets* tai *Worksheets*) tai kaavioihin (*Chart*). Taulukkosivut koostuvat soluista (*Range* tai *Cells*) ja kuvaajat taas grafiikan komponenteista (esim. *Floor* ja *Legend*). Näillä on puolestaan omat ominaisuutensa, kuten esi-

merkiksi solun fontin väri ja lihavointi. Otetaan esimerkki, jossa työkirjan ”Tyokirja1” taulukon ”Taul2” solun *B5* fontti halutaan lihavoida. (Merensalmi 2007, 41-42.) Asetuslause voidaan muodostaa esimerkiksi seuraavasti objektit pisteellä erottaen:

```
Workbooks("Tyokirja1").Worksheets("Taul2").Range("B5").Font.Bold = True
```

Jos taas ohjelman suorituksen aikana ei haluta päivittää tapahtumia näytössä, voidaan tämä toteuttaa sovellukseen viitaten seuraavasti:

```
Application.ScreenUpdating = False
```

Useat objektit ovat kokoelmia. Objekti, joka toistuu samanlaisena useaan kertaan, on kokoelma. Esimerkiksi työkirjan taulukkosivut yhdessä muodostavat taulukkosivujen (*Worksheets*) kokoelman. Kokoelmatyyppisten objektien nimi esitetään aina monikossa. (Merensalmi 2007, 42.)

Objekteilla voi olla myös ominaisuuksia. Esimerkiksi työkirjalla on ominaisuuksina muun muassa nimi tai taulukkosivujen kappalemäärä. Solun ominaisuuksia taas ovat esimerkiksi osoite, arvo ja fontin muotoilu. Yleensä ominaisuudet ovat sellaisia, että niiden arvon voi noutaa, ja nykyisen arvon päälle voi asettaa uuden arvon. On olemassa kuitenkin sellaisiakin ominaisuuksia, jotka voidaan ainoastaan lukea. (Merensalmi 2007, 42.)

Metodi käsittää objektiin liittyvän toiminnallisuuden. Työkirjaan liittyviä metodeja ovat muun muassa avaaminen, sulkeminen ja tallentaminen. Solun metodeja taas ovat esimerkiksi aktivointi, tyhjennys ja kopiointi. Kun metodia käytetään, muutetaan objektin jotain toista ominaisuutta. Kun esimerkiksi työkirja aktivoidaan, se merkitään samalla aktiiviseksi. (Merensalmi 2007, 42.)Seuraavassa esimerkissä viitataan työkirjat-kokoelmaan ja käytetään avausmetodia avaamaan työkirja ”esimerkki.xlsx”, joka sijaitsee *C:*-asemalla:

```
Workbooks.Open ("C:\esimerkki.xlsx")
Workbooks("alkuperainen.xlsx").Activate
```

Esimerkin toisella rivillä aktivoidaan takaisin ennen työkirjan avausta esillä ollut työkirja ”alkuperainen.xlsx”.

4.2.3 Ohjelmien yleinen rakenne

VBA-ohjelmia on pää- ja aliohjelmia. Pääohjelma on yleensä suoraan käyttäjälle tarkoitettu ohjelma, joka voidaan suorittaa esimerkiksi painikkeen avulla. Aliohjelmat taas ovat yleensä sellaisia, että niiden suorittaminen vaatii pääohjelman kutsun sekä mahdolliset esiparametroinnit. Aliohjelmien käytön avulla saadaan vähennettyä koodin määrää. Jos jokin toimenpide tehdään useasti ohjelman suorituksen aikana, kannattaa se tehdä aliohjelmaksi. Tällöin kyseisen toimenpiteen kohdalla tarvitsee vain kutsua aliohjelmaa, jolloin koodi lyhenee useammasta rivistä yksiriviseen kutsukäskyyn. (Merensalmi 2007, 57-58.)

Ali- ja pääohjelmat voivat olla joko funktio- tai proseduurityyppisiä. Funktiotyyppisessä (*Function*) ohjelmassa sitä kutsuvalle ohjelmalle palautetaan arvo funktion lopputuloksesta tai vastauksesta. Proseduurityyppisestä (*Sub*) ohjelmasta lopputulosta taas ei palauteta, vaan se tekee ainoastaan siihen kirjoitetut toiminnot. Toimintojen jälkeen ohjelmakoodin suoritus jatkuu pääohjelman proseduurikutsua seuraavalta riviltä. (Merensalmi 2007, 57-58.) Proseduurityyppinen ohjelman rakenne on seuraava:

```
Sub ohjelman_nimi()
'toimenpiteet
End Function
```

Funktiotyyppisen ohjelman rakenne on samanlainen kuin proseduurityyppisen, mutta otsikkorivin *Sub*-sanan tilalla on *Function*-sana. Proseduurityyppistä aliohjelmaa voidaan kutsua pääsovelluksesta käskyllä *Call aliohjelman nimi*. Funktion käyttöön riittää pelkkä funktion nimen kirjoitus. (Merensalmi 2007, 57-58.)

Käytännössä ohjelmointikoodin kirjoittamisessa yhden rivin pituudelle ei ole asetettu maksimirajaa, eli käyttäjän ei itse tarvitse rivittää koodia. Kuitenkin hallinnan ja luettavuuden kannalta kannattaa pidemmät rivit jakaa useammalle riville. Tämä onnistuu lisäämällä rivin loppuun välilyönnin ja ”_” -merkin. Tämän jälkeen tehdään rivinvaihto ja jatketaan koodin kirjoittamista. Koodia voidaan myös kommentoida aloittamalla kommenttiteksti puolilainausmerkillä. (Merensalmi 2007, 22-23.)

4.2.4 Muuttujat ja parametrit

Muuttujiin tallennetaan koodin suorituksen aikana tietoa. Nämä ovat väliaikaisia muistipaikkoja, jolloin esimerkiksi taulukon solun arvo voidaan ottaa talteen muuttujaan. Tällöin arvoa ei tarvitse toistuvasti hakea solusta. Tarvittavat muuttujat on hyvä esitellä koodin alussa sekä määrittää niiden tietotyyppi. Se ei ole välttämätöntä, mutta määrittelemätön muuttuja vie muistitilaa suurimman tietotyypin eli Variant-tyypin verran (16 tavua tai enemmän. ks. taulukko 1). Jos määrittelemättömiä muuttujia alkaa olla useita kymmeniä, vie ohjelma paljon muistia. Tietotyypit kannattaa siis määritellä muuttujille niiden käyttötarpeen mukaan, jotta muistin käyttö olisi mahdollisimman vähäistä. Käytetyt muistityypit on esitetty taulukossa 1. Tietotyyppejä on totuusarvojen, erilaisten numeroarvojen, merkkijonojen ja päivämäärien tallentamista varten. (Merensalmi 2007, 65.)

TAULUKKO 1. muistityypit (Lähde: 1)

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

Seuraavaksi on esitetty esimerkki muuttujan ”muuttuja” esittelystä, joka koodin suorituksen aikana tulee sisältämään lukuja 0 - 1000:

```
Dim muuttujan_nimi As Integer
```

Muuttujan esittely aloitetaan sanalla *Dim*, jonka jälkeen kirjoitetaan muuttujan nimi. Lopuksi kirjoitetaan sana *As*, sekä itse muuttujan tietotyyppi.

Pääohjelman ja aliohjelman välillä voidaan siirtää tietoa parametrien avulla. Nämä määritellään aliohjelman otsikkorivillä olevien sulkeiden sisälle. Kutsuvan ohjelman pitää antaa aliohjelmaa kutsuttaessa parametreille jokin arvo. Jos näin ei tapahdu, ei aliohjel-

ma osaa toimia. Myös parametrien tietotyyppi voidaan esitellä samaan tapaan kuin muuttujilla. (Merensalmi 2007, 91-94.)

4.2.5 Ohjausrakenteet

Ohjausrakenteiden avulla pystytään luomaan itse ohjelman toiminta eli automatisoimaan asioita. Yleisiä ohjausrakenteita ovat asetuslause, ehtolause, silmukkarakenne ja hyppyrakenne. Makronauhurilla nauhoitettu ohjelmakoodi käyttää ainoastaan asetuslausetta, sillä tällöin ainoastaan tehdään asioita. Kun asioita halutaan ehdollistaa tai toistaa, ei ohjelmakoodia voida toteuttaa muuten kuin käsin kirjoittamalla. (Merensalmi 2007, 76-77.)

Asetuslauseen avulla muuttujiin voidaan asettaa arvoja tai muuta tietoa. Sen avulla muuttujan arvoksi voidaan esimerkiksi asettaa taulukon tietyn solun arvo, jotta sitä ei koodin suorituksen aikana tarvitse noutaa toistuvasti. Muuttuja, jonka arvoa muutetaan, sijaitsee aina yhtäsuuruusmerkin vasemmalla puolella. Yhtäsuuruusmerkin oikealla puolella oleva komentosarja tai toinen muuttuja määrittelee uuden arvon vasemman puoleiseen muuttujaan. (Merensalmi 2007, 77.) Seuraavaksi on esitetty esimerkki asetuslauseesta, jossa muuttuja1:n arvoksi asetetaan *muuttuja2*:n arvo, johon on lisätty luku 4:

```
muuttuja1 = muuttuja2 + 4
```

Asetuslauseissa tulee huomioida myös muuttujien tietotyypit. Muuttujaan voidaan asettaa ainoastaan sille määritetyn tietotyypin mukaista tietoa. Myös asetuslauseen komentosarjassa tulee käyttää oikeanlaisia operointitapoja; merkkijonomuuttujien operointi merkkijono-operaattoreilla ja lukuarvomuuttujien operointi matemaattisilla operaattoreilla. (Merensalmi 2007, 78.)

Ehtolauseissa ja silmukkarakenteissa voidaan käyttää loogisia operaattoreita, joita ovat *And, Or* ja *Not*. Niiden avulla sidotaan kaksi ehtoa toisiinsa. Jos ehtolause riippuu esimerkiksi sekä ehdosta 1 että ehdosta 2, nämä kaksi ehtoa pitää sitoa keskenään loogisella operaattorilla. Kun ehdot ovat sidottu *And*-operaattorilla, tulee molempien ehtojen toteutua, jotta ehtolause toteutuu. *Or*-operaattorilla, jomman kumman ehdon tulee toteu-

tua, jotta ehtolause toteutuu. *Not*-operaattorin avulla ehtolause toteutuu, jos esimerkiksi ehto 1 ei toteudu. Toteutumista kuvataan totuusarvoilla Tosi (*True*) ja Epätosi (*False*). (Merensalmi 2007, 79-80.)

Kun ohjelman toimintaa halutaan ehdollistaa, käytetään yleensä ehtolausetta *If*. Ehtolauseen avulla ohjelman toimintaa voidaan jakaa eli voidaan toimia eri tavalla erilaisissa tilanteissa. (Merensalmi 2007, 80.) Seuraavaksi on esitetty esimerkki ehtolauseesta, jossa *muuttuja2*:n arvo asetetaan arvoksi 10, jos *muuttuja1*:n arvo on 5:

```
If muuttuja1 = 5 Then
muuttuja2 = 10
End If
```

Ehtolauseessa määritellään ensin ehto, jonka tulee toteutua, jotta määritetty toimenpide tehdään. Tämän jälkeen tulee kirjoittaa sana *Then*, minkä jälkeen kirjoitetaan ehtolauseen toteuduttua tehtävä toimenpide. Ehtolause voidaan kirjoittaa myös yksiriviseksi, jolloin viimeistä *End If* -käskyä ei tarvita. Ehtolauseeseen voidaan lisätä ehto, joka tehdään silloin, jos ehto ei toteudu. Tämä tehdään *Else*-sanan avulla. Jos ehtoja taas halutaan enemmän kuin yksi, tulee seuraavat ehdot *If*-ehdon jälkeen määritellä *ElseIf*-sanan avulla. (Merensalmi 2007, 81-82.)

Pitkä ehtolause on mahdollista korvata myös monivalintalauseella *Select Case*. Tämä valitsee sopivan vaihtoehdon tilanteen mukaan. (Merensalmi 2007, 83.) Seuraavassa esimerkissä valitaan toiminta muuttuja1:n mukaan:

```
Select Case muuttuja1
  Case 5: muuttuja2 = 7
  Case Is > 10: muutuja2 = 11
  Case Else: muuttuja2 = 1
End Select
```

Jos esimerkin *muuttuja1*:n arvo on 5, on *muuttuja2*:n arvo 7, ja jos *muuttuja1*:n arvo taas on suurempi kuin 10, on *muuttuja2*:n arvo 11. Muuten *muuttuja2*:n arvo on 1. Monivalintalause päättyy käskyyn *End Select*.

Ohjelmissa on yleensä tarvetta toistaa sama toimenpide useita kertoja, jotta tietty toiminto saadaan suoritettua. Tällöin koodi kirjoitetaan silmukkamuotoon. Yleisin silmukkarakenne on *While*-silmukka, joka aloitetaan sanayhdistelmällä *Do While*. Se suorittaa

silmukkaa niin kauan, kunnes haluttu ehto ei enää toteudu. Vastakkainen toiminta tällä on *Do Until* -silmukkarakenne, eli silmukkaa suoritetaan niin kauan, kunnes haluttu ehto toteutuu. *Do*-sanalla alkavat silmukkarakenteet päätetään lauseella *Loop*. Silmukkaa suoritettaessa *Loop*-käskystä palataan aina silmukan ensimmäiselle riville, kunnes ehto täyttyy. (Merensalmi 2007, 84-85.) Seuraavassa esimerkissä *Do Until* -silmukkaa suoritetaan niin kauan, kunnes *muuttuja1*:n arvo on suurempi kuin 5:

```
Do Until muuttuja1 > 5
'toimenpiteitä
muuttuja1 = muuttuja1 + 1
Loop
```

Esimerkissä toimenpiteiden jälkeen *muuttuja1*:n lisätään arvo 1, jolloin jossain vaiheessa ohjelman suorituksen aikana *muuttuja1*:n arvo on 5. VBA:ssa on myös *For*-silmukkarakenne. Sitä suoritetaan kunnes silmukan sisällä oleva muuttuja on yhtä suuri tai suurempi kuin silmukan aloitusrivillä määritetty loppuarvo. Jokainen silmukan kierros kasvattaa muuttujan arvoa yhdellä. *For*-silmukkarakenne päätetään käskyllä *Next*. (Merensalmi 2007, 87-88.)

Jos silmukan ehdon arvoa unohdetaan päivittää suorituksen aikana tai asetettu ehto ei muuten voi toteutua, ajautuu se ikuiseen silmukkaan. Tällöin suoritus ei pääty koskaan ilman voimakeinoja. Koodin kirjoittaja voi keskeyttää ikuisen silmukan CTRL+Break -näppäinyhdistelmällä tai Run/Reset-painikkeella. Ohjelman käyttäjä ei tätä kuitenkaan voi tehdä, jolloin Excel kaatuu. Tällaisen tilanteet pitää huomioida ohjelmaa kirjoitettaessa. (Merensalmi 2007, 89.)

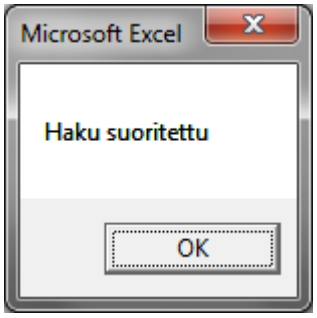
Hyppyrakenteen avulla voidaan hypätä ohjelmakoodissa ulos silmukasta tai aliohjelmasta (*Exit*). Se avulla voidaan hypätä myös erikseen määritettyyn leimaan (*GoTo*). Hyppyrakenteesta on hyötyä esimerkiksi virhetilanteiden hallinnassa, jolloin virheen ilmaantuessa poistutaan normaalista toiminnasta ja tehdään virheen mukainen toiminta. *Exit*-käskyn avulla poistutaan silmukasta esimerkiksi sanayhdistelmällä *Exit Do* tai aliohjelmasta *Exit Sub*. *GoTo*-tyyppisen hyppykomennon leima määritellään siten, että *Goto*-lauseen jälkeen kerrotaan hypyn leima. Käytetty leima kirjoitetaan siihen kohtaan koodia, johon halutaan hypätä. Virheen numero löytyy aina Excelin oman muuttujan ominaisuudesta *Err.number*. Toiminta tietyllä virhenumerolla voidaan toteuttaa ehto- tai monivalintalauseen avulla. (Merensalmi 2007, 89-91.)

4.2.6 Käyttäjän kanssa kommunikointi

Käyttäjää on mahdollista informoida esimerkiksi ohjelman suorituksen tilasta tai toiminnon onnistumisesta. Tällainen toiminta on mahdollista toteuttaa *MsgBox*-funktion avulla. (Merensalmi 2007, 128.)Seuraavassa esimerkissä on esitetty ohjelma, jossa käyttäjälle annetaan ilmoitus ”Haku suoritettu”, kun muut toimenpiteet on suoritettu:

```
Sub esimerkki()
'toimenpiteitä
MsgBox ("Haku suoritettu")
End Sub
```

Kuvassa 3 on esitetty ruutuun näkyviin tuleva sanomaikkuna. Se sisältää myös OK-painikkeen, jolla sanomaikkuna kuitataan ja suljetaan.



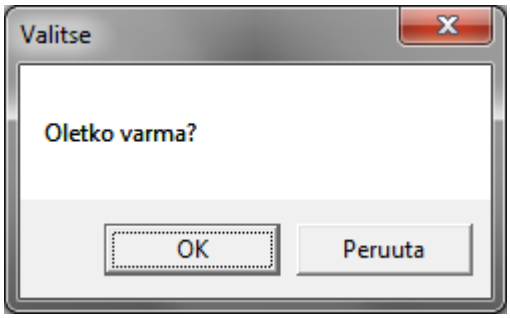
KUVA 3. Sanomaikkuna

Ohjelman käytössä saattaa olla myös tarpeellista, että sanomaikkuna sisältää OK-painikkeen lisäksi esimerkiksi Peruuta-painikkeen (Cancel). Tällainen tilanne esiintyy esimerkiksi silloin, jos käyttäjän halutaan hyväksyvän tai hylkäävän seuraavaksi suoritettava toimenpide. Seuraavassa esimerkissä on esitetty sanomaikkuna, joka kysyy käyttäjältä ”Oletko varma?” ja pyytää häntä joko hyväksymään tai hylkäämään ikkunan kysymyksen:

```
vastaus = MsgBox("Oletko varma?", vbOKCancel, "Valitse")
If vastaus = vbOK Then
ActiveCell.Value = 3
Else
Exit Sub
End If
```

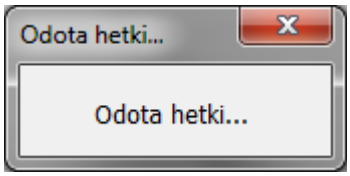
Esimerkissä aktiivisena olevaan soluun syötetään arvo 3, mikäli painetaan OK-painiketta. Peruuta-painiketta painettaessa ohjelman suoritus lopetetaan. Käyttäjän te-

kemä valinta sanomaikkunassa tallennetaan muuttujaan *vastaus*, jota käytetään ehtolauseessa. Sanomaikkunan otsikoksi on myös asetettu ”Valitse”, joka näkyy MsgBox komennon sulkeiden sisällä viimeisenä. Kuvassa 4 on esitetty ruutuun näkyviin tuleva sanomaikkuna.



KUVA 4. Sanomaikkuna OK ja Peruuta painikkeella.

Jos näytettävään ikkunaan halutaan lisätä useita painikkeita, valintoja tai kuvia, toteutetaan ikkuna käyttäjälomakkeella. Sen ulkoasua on myös mahdollista muokata haluamansa näköiseksi. Käyttäjälomake ovat tarkoitettu pääasiassa käyttöliittymän tai lomakkeiden rakentamiseen (Merensalmi 2007, 14.). Sen avulla on kuitenkin mahdollista toteuttaa myös esimerkiksi ohjelman suorituksen aikana ohjelman suorituksen tilasta kertovia informaatioikkunoita. Yksinkertaisimmillaan tällainen voi olla ikkuna, jossa lukee ”Odota hetki” ohjelman suorituksen ajan (KUVA 5). Teksti-ikkunaan on kirjoitettu käyttäjälomakkeen tekstikontrollin avulla.



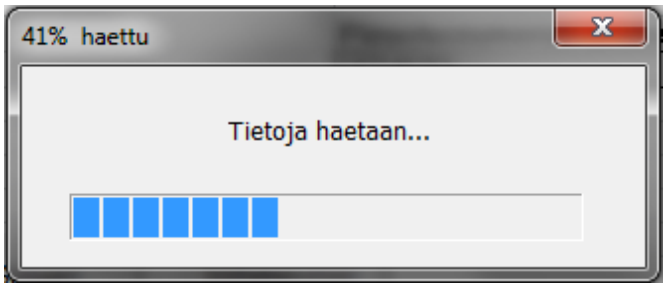
KUVA 5. Käyttäjälomakkeella toteutettu informaatioikkuna.

Jotta käyttäjälomake saadaan näkyviin koodin suorituksen aikana, pitää tätä kutsua ohjelmakoodissa. Seuraavaksi on esitetty esimerkki, jossa ennen toimenpiteitä kutsutaan käyttäjälomaketta ”Odota” ja suljetaan se toimenpiteiden jälkeen:

```
Odota.Show vbModeless
DoEvents
'toimenpiteitä
Odota.Hide
```

Käyttäjälomakkeen avaaminen tapahtuu siis lisäämällä käyttäjälomakkeen nimen perään metodi *Show* ja sulkeminen metodilla *Hide*. Avaamisen yhteydessä on myös vakio *vbModeless*, joka antaa ohjelman suorittua vaikka käyttäjä ei tekisi mitään. Jos tämän tilalla olisikin *vbModal*, odotettaisiin käyttäjältä jonkinlaisia toimenpiteitä ennen ohjelman jatkamista. Saman asian voi esittää myös vakioilla *True* (*vbModal*) ja *False* (*vbModeless*). Jotta toimenpiteet suoritetaan, tulee käyttäjälomakkeen näyttämisen jälkeen kirjoittaa ohjelmakoodiin komento *DoEvents*.

Käyttäjälomakkeessa olevien komponenttien eli kontrollien ominaisuuksia on mahdollista muokata myös koodin suorituksen aikana itse koodista. Tästä on hyötyä esimerkiksi silloin, kun käyttäjää halutaan informoida ohjelman tilasta prosenttitiedolla ja tilapalkilla. Esimerkki tällaisesta ikkunasta on esitetty kuvassa 6.



KUVA 6. Käyttäjän informointi ohjelman tilasta.

Seuraavaksi on esitetty kuvan 6 mukaisen tilaikkunan ohjaus ohjelmakoodista:

```
With Tila
    .ProgressBar1.Min = 0
    .ProgressBar1.Max = tietomaara
    .Show vbModeless
haettu = 1
Do Until (haettu > tietomaara)
        .ProgressBar1.Value = haettu
        .Caption = VBA.Format(haettu / Tila.ProgressBar1.Max, "0%") & "  haettu"
        DoEvents
        'toimenpiteitä
        haettu = haettu + 1
Loop
End With
Unload Odota
```

Esimerkin alussa esiintyy yhdistäjälause *With*, jonka avulla samaan objektiin liittyviä ominaisuuksia ja metodeja voidaan kirjoittaa ilman objektin kirjoittamista uudelleen. *With* päätetään komennolla *End With*. Koodin alussa tilapalkin ”ProgressBar1” minimiarvoksi asetetaan nolla ja maksimiarvoksi muuttujan *tietomaara* sisältö. Ennen *Do*

Until -silmukkarakennetta muuttuja *haettu* alustetaan arvoon yksi. Tilapalkkia ja prosenttitietoa päivitetään silmukkarakenteen kierrosten perusteella, jota suoritetaan niin kauan, kunnes muuttuja *haettu* on suurempi kuin muuttuja *tietomaara*. Silmukan alussa tilapalkin arvoksi päivitetään muuttujan *haettu* arvo. Tilapalkissa tämä näytetään skaalattuna tilapalkin min- ja max-arvojen välille. Ikkunan otsikkokentässä (*Caption*) näytetään prosenttitieto, joka on laskettu jakamalla muuttujan *haettu* arvo tilapalkin maksimilla. Vaikka tuloksen arvo on esitetty desimaaleina, osataan tämä tulkita prosentteina metodin *VBA.Format* avulla. Prosenttiluvun perässä on teksti ”haettu”. Tämän jälkeen tehdään toimenpiteet ja lopuksi lisätään muuttujaan *haettu* luku yksi. Silmukan jälkeen päätetään yhdistäjälause *With* komennolla *End With*. Lopuksi tilapalkki poistetaan näkyvistä, mutta tässä tapauksessa sitä ei tehdä metodilla *Hide.* Kun kyseessä on käyttäjälomake, jota on päivitetty ohjelman suorituksen aikana, voidaan tämä poistaa näkyvistä ainoastaan *Unload*-käskyn avulla. Samalla ohjelman suorituksen aikana muokatut käyttäjälomakkeen tiedot palautuvat oletusarvoihinsa.

4.2.7 Virheet ja niiden käsittely

Ohjelmakoodin suorituksen aikana saattaa ilmaantua virheitä, joista VBA-editori ilmoittaa käyttäjälle virheilmoituksella. Virheilmoituksessa on nähtävissä virheen tunnusnumero ja kuvausteksti virheen aiheuttajasta. Tyypillisesti ohjelman suoritus pysähtyy tällaisessa tilanteessa eikä voi jatkua. Voi kuitenkin olla ennalta tiedossa, että tietty virhe saattaa ilmaantua ohjelman suorituksen aikana. Tällöin halutaan tehdä toimenpide virheen korjaamiseksi tai muuten ohjelman jatkamiseksi. Virheiden käsittely saadaan aikaiseksi hyppyrakenteen avulla. Esimerkiksi *On Error GoTo virhe:* -komento määritellään koodin alussa, jolloin virheen ilmaantuessa missä tahansa vaiheessa koodin suoritusta hypätään leimaan virhe. (Merensalmi 2007, 108-109.) Esimerkiksi virhe 9 ilmaantuu, jos työkirjaa, johon tietty käsky viittaa, ei ole auki. Seuraavaksi on esitetty esimerkki tilanteesta, jossa virheellä 9 valittuna olevaan soluun kirjoitetaan sana ”VIRHE”:

```
On Error GoTo virhe:
'toimenpiteitä
virhe:
If Err.Number = 9 Then ActiveCell = "VIRHE"
```

5 GENEROINTITYÖKALU

5.1 Lähtötilanne

Tämän kappaleen sisältö on julistettu salaiseksi.

5.1.1 Generointitaulukon rakenne

Tämän kappaleen sisältö on julistettu salaiseksi.

5.1.2 Lähdetaulukot

Tämän kappaleen sisältö on julistettu salaiseksi.

5.1.3 Ongelmien kartoitus ja kehitysideat

Tämän kappaleen sisältö on julistettu salaiseksi.

5.2 Suunnittelu

Tämän kappaleen sisältö on julistettu salaiseksi.

5.3 Toiminta ja käyttö

Tämän kappaleen sisältö on julistettu salaiseksi.

5.3.1 Tiedonhaun perusrakenne

Tämän kappaleen sisältö on julistettu salaiseksi.

5.3.2 Yleinen tiedonhakualiohjelma

Tämän kappaleen sisältö on julistettu salaiseksi.

5.3.3 IO-tiedonhakualiohjelma

Tämän kappaleen sisältö on julistettu salaiseksi.

5.3.4 Virheet ja varoitukset

Tämän kappaleen sisältö on julistettu salaiseksi.

5.3.5 Erityispiirteet tiedonhaussa

Tämän kappaleen sisältö on julistettu salaiseksi.

5.3.6 Solujen mustaus

Tämän kappaleen sisältö on julistettu salaiseksi.

6 GENEROINTITYÖKALUN TESTAUS

6.1 Testauksen suunnittelu

Tämän kappaleen sisältö on julistettu salaiseksi.

6.2 Testaus

6.2.1 Yleinen testaus

Tämän kappaleen sisältö on julistettu salaiseksi.

6.2.2 Tiedonhakusnopeus

Tämän kappaleen sisältö on julistettu salaiseksi.

6.3 Työkalun hyödyntäminen suunnitteluprojektissa

Tämän kappaleen sisältö on julistettu salaiseksi.

7 POHDINTA

Tämän kappaleen sisältö on julistettu salaiseksi.

LÄHTEET

Insta Automation Oy. 2014. Generointitietokantaohje. Tulostettu 12.2.2014

Insta Goup Oy. 2014. Insta. Tulostettu 3.3.2014. http://www.insta.fi

Kymdata Oy. 2014a. CADS Planner. Tulostettu 3.3.2014. http://www.cads.fi

Kymdata Oy. 2014b. CADS Planner Electric Piirikaaviot. Ohjelman sisäinen käyttöohje. Tulostettu 9.4.2014

Merensalmi, J. 2007. Excel VBA yrityskäytössä. 1. painos. Jyväskylä: WSOY-pro/Docendo-tuotteet.

Microsoft. 2014a. PHAKU. Tulostettu 7.4.2014
http://office.microsoft.com/fi-fi/excel-help/phaku-HP005209335.aspx (Phaku)

Microsoft. 2014b. Timer Function. Tulostettu 11.4.2014.
http://msdn.microsoft.com/en-us/library/office/gg264416(v=office.15).aspx (Timerf)

Taanila, A. 2013. Excel VBA-ohjelmointi. Tulostettu 12.2.2014.
http://myy.haaga-helia.fi/~taaak/vba/vba.pdf

LIITTEET

Liite 1. Generointitietokantanumerot

Tämän liitteen sisältö on julistettu salaiseksi.

(Insta Automation Oy 2014.)

Liite 2. Tiedostosijainnit-taulukkosivu

Tämän liitteen sisältö on julistettu salaiseksi.

Liite 3. Tiedonhaun perusrakenne

Tämän liitteen sisältö on julistettu salaiseksi.

Liite 4. Sarakekohtaisen tiedonhaun koodin perusrakenne

Tämän liitteen sisältö on julistettu salaiseksi.

Liite 5. *MHAKU*-tiedonhakualiohjelma

Tämän liitteen sisältö on julistettu salaiseksi.

Liite 6. Mustaustoiminnon ohjelmakoodi

Tämän liitteen sisältö on julistettu salaiseksi.

Liite 7. Pohjakuvien rakenne 1

Tämän liitteen sisältö on julistettu salaiseksi.

Liite 8. Pohjakuvien rakenne 2

Tämän liitteen sisältö on julistettu salaiseksi.