AUTOMAATION TOIMINTAKAAVIOIDEN LAATIMINEN MICROSOFT VISIO 2010 -OHJELMALLA

Kalle Hyvärinen

Opinnäytetyö
Lokakuu 2011

Automaatiotekniikka
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t)	Julkaisun laji	Päivämäärä
HYVÄRINEN, Kalle	Opinnäytetyö	04.10.2011
	Sivumäärä 35	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
AUTOMAATION TOIMINTAKAAVIOIDEN LAATIMINEN MICROSOFT VISIO 2010 -OHJELMALLA

Koulutusohjelma
Automaatiotekniikan ko.

Työn ohjaaja(t)
HÄKKINEN, Veli-Matti

Toimeksiantaja(t)
JEEC Oy
PEKKANEN, Jarmo

Tiivistelmä

Opinnäytetyön tavoitteena oli luoda generointiohjelma automatisoimaan toimintakaavioiden laatimisprosessia Microsoft Visio 2010-ohjelmalla. Generointiohjelman lisäksi opinnäytetyössä tuli luoda toimintakaavioiden pohjakuvien piirtämistä helpottavia, alalla sovellettaviin standardeihin perustuvia symbolikirjastoja sekä kirjoittaa käyttöohje helpottamaan suunnittelijoiden työskentelyä opinnäytetyön tuloksena syntyneiden työkalujen kanssa. Työn toimeksiantajana toimi jyväskyläläinen automaatio- ja sähköalan suunnitteluyritys JEEC Oy.

Generointiohjelma on toteutettu tässä opinnäytetyössä Visual Basic for Applications eli VBA -ohjelmointikielellä kirjoitetulla makro-ohjelmalla. Opinnäytetyössä luodut symbolikirjastot ovat puolestaan suunniteltu yhdenmukaisiksi toimeksiantajan CAD-suunnittelussa käyttämien kirjastojen kanssa. Valmiiden kaavioiden tulostamista varten syntyi myös massatyökalu, jonka avulla suuri määrä kaavioita on mahdollista tulostaa yhdellä napin painalluksella.

Ohjelmalliset ratkaisut ovat osoittautuneet koekäytössä helppokäyttöisiksi ja suunnittelutyötä huomattavasti nopeuttaviksi työkaluiksi. Generointi- ja tulostusohjelmien ansiosta manuaalisesti suoritettavan tiedonsiirron ja työn tarve on vähentynyt murto-osaan verrattuna tilanteeseen, jossa suunnittelija joutuu tekemään kaiken alusta loppuun ilman automatisoinnin tarjoamaa apua. Tehostunut ajankäyttö vähentää luonnollisesti toimeksiantajan kustannuksia ja mahdollistaa tehokkaamman toiminnan suunnitteluprosesseissa.

Opinnäytetyön tuloksena syntyneen ohjelman koodi on kommentoitu siten, että sen laajennus ja kehitystyö jatkossa on mahdollista kenelle tahansa VBA -ohjelmoinnin hallitsevalle suunnittelijalle. Myös symbolikirjastojen laajennus tilaajien vaatimusten mukaan onnistuu helposti opinnäytetyön yhtenä lopputuotteena syntyneen käyttöohjeen opastamana.

Avainsanat (asiasanat)
Microsoft Visio 2010, toimintakaavio, Visual Basic for Applications, ohjelmointi, symbolikirjasto

Muut tiedot
Liitteenä ohjelmien vuokaaviot sekä esimerkkejä VBA -ohjelmakoodista, 12 sivua.

Author(s)
HYVÄRINEN, Kalle

Type of publication
Bachelor´s Thesis

Date
04.10.2011

Pages
35

Language
Finnish

Confidential

( ) Until

Permission for web publication
(X)

Title
DRAFTING FUNCTION DIAGRAMS FOR AUTOMATION WITH MICROSOFT VISIO 2010 SOFTWARE

Degree Programme
Automation engineering

Tutor(s)
HÄKKINEN, Veli-Matti

Assigned by
JEEC Ltd.
PEKKANEN, Jarmo

Abstract

The aim of the thesis was to create a programmatic solution to automate the drafting of function diagrams with the Microsoft Visio 2010 software. Creating symbol libraries based on commonly used standards in automation industry was also included in the thesis, as well as writing the user guide to aid automation designers to work with the end products of the thesis process. The thesis was assigned by JEEC Ltd, which offers engineering and consultant services in the fields of automation and electrification.

Automation of the drafting of function diagrams in Visio 2010 has been implemented in this thesis with the help of a macro program, written in the Visual Basic for Applications (VBA) programming language. The program generates diagrams based on templates and specific user defined data inputs. Templates can be drawn easily with the help of symbol libraries. The symbols are similar to those which have been in use in the CAD development environment in the company. The VBA macro program for printing vast number of diagrams with a single button click was also created as a result of the thesis process.

Testing of macro programs have proved that they provide significant saving of time and effort by reducing the demand of manual work in the designing process, enabling the designers to work more efficiently. The above-mentioned naturally helps the company to exploit their resources better.

The program codes are documented in the codes' comments the way that enables further development of macro programs to be made by any software developer familiar to VBA programming. With the help of the instructions of the user guide, symbol libraries are also well extensible in the future for example based on the demands of the clients.

Keywords
Microsoft Visio 2010, function diagram, Visual Basic for Applications, software development, symbol library

Miscellaneous
Flow charts of macro programs and examples of VBA codes as attachments, 12 pages.

SISÄLTÖ

KUVIOT

TAULUKOT

1 TOIMEKSIANTAJA JA TAVOITTEET

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi JEEC Oy, jonka toimitilat sijaitsevat Jyväskylän Palokassa. Vuonna 2009 perustettu JEEC tarjoaa korkealaatuisia suunnittelu- ja konsultointipalveluja automaatio- ja sähkötoimialoilla. Yrityksen ydinosaamista ovat automaatioalalla prosessiautomaatiojärjestelmien suunnittelu ja käyttöönotto, sähkötoimialoilla prosessi- ja rakennussähkösuunnittelu sekä tele- ja turvajärjestelmien suunnittelu. Automaatiotoimituksissa JEEC pystyy toteuttamaan hankkeiden koko elinkaaren mittaisen palvelun esisuunnittelusta aina ylläpitotehtäviin asti. (JEEC Oy 2011.)

1.2 Toimeksiantajan tavoitteet

Opinnäytetyön aihe syntyi toimeksiantajan tarpeesta luoda automaation perussuunnitteluvaiheessa tuotettavia toimintakaavioita Microsoft Visio 2010 -ohjelmalla. Lähtökohtana oli pyrkiä automatisoimaan toimintakaavioiden laatimisprosessia siten, että manuaalisesti suoritettavan työn ja tiedonsiirron tarve vähenisi ja suunnitteluprosessi nopeutuisi sitä mukaa huomattavasti.

Opinnäytetyöprosessin käynnistyessä tavoitteeksi asetettiin generointiohjelma, joka vertailisi pohjakuvan tunnisteita ja korvaisi nämä tunnisteet niitä vastaavilla lähtötiedoilla. Toiminnalliselta näkökannalta identtisesti toimivia kohteita varten tarvitsisi siis piirtää vain yksi pohjakuva, jonka jälkeen kaavioon sisältyvät tunnukset ja muu informaatio pystyttäisiin täydentämään ohjelmallisesti. Valmiin generoidun kaavion tulisi tallentua automaattisesti kovalevylle. Toimeksiantajalla oli käytössä vastaavalla toimintaperiaatteella toteutettu CAD-ympäristössä toimiva ohjelmistoratkaisu, jonka tarjoamaa mallia tässä opinnäytetyössä pyrittiin soveltuvin osin seuraamaan.

Generointiohjelman toiminnallisessa puolessa tuli kiinnittää huomiota helppokäyttöisyyteen ja ohjelman resurssivaatimuksiin. Tavoitteeksi asetettiin käyttöliittymältään mahdollisimman yksinkertainen ja helposti omaksuttava ratkaisu. Ohjelman toiminta pyrittiin puolestaan saamaan mahdollisimman nopeaksi, jotta sen käyttö osana suunnitteluprosesseja olisi järkevää ja perusteltua. Valmiiden kaavioiden tiedostokoko oli pyrittävä pitämään mahdollisimman pienenä ja Vision oman tiedostoformaatin lisäksi toimeksiantaja halusi mahdollisuuden tallentaa kaavioita XML-tiedostoina.

Kaavioiden tulostamista varten tuli myös kehitellä ohjelmallinen ratkaisu, sillä jopa satojen kaavioiden tulostaminen yksitellen ei ole ajankäytön kannalta viisasta saati mielekästä toimintaa. Generointiohjelman tulevia käyttäjiä varten oli puolestaan kirjoitettava käyttöohje, joka perehdyttää ohjelman käyttöön ja opastaa uusien pohjakuvien piirtämisessä.

Visio 2010 -ohjelma sisältää runsaasti valmiita symbolikirjastoja, mutta näiden visuaalinen ilme ei vastaa Euroopassa automaatio-alalla yleisesti sovellettavia standardeja. Näin ollen osaksi opinnäytetyön tavoitteita asetettiin toimeksiantajan tarkoituksiin soveltuvien symbolikirjastojen luominen. Edellä mainittujen kirjastojen avulla uusien pohjakuvien piirtäminen helpottuisi ja nopeutuisi. Symbolikirjastojen käyttö pohjakuvia piirrettäessä takaisi myös valmiiden toimintakaavioiden yhdenmukaisuuden. Myös symbolikirjastojen osalta päädyttiin noudattamaan toimeksiantajan CAD-suunnittelussa käyttämiä symboleja.

1.2 Henkilökohtaiset tavoitteet

Henkilökohtaisesti pyrin opinnäytetyöprosessin alusta alkaen kohti lopputulosta, joka olisi mahdollisimman käyttökelpoinen toimeksiantajan tulevissa projekteissa. Myös generointiohjelman ja kirjastojen sekä itse generoinnin ohjelmallisen toteutuksen mahdollisimman helppo laajennettavuus jatkossa oli mielestäni tärkeää. Tämä opetti itseäni opinnäytetyöprojektin aikana riittävän dokumentoinnin tärkeydestä sovellussuunnittelussa ja ylipäätään kehitysprojekteissa. Lisäksi tavoitteena oli opinnäytetyössä käytettävien ohjelmointikielten ymmärryksen ja hallinnan syventäminen sekä

Microsoft Visio 2010 -ohjelman kokonaisvaltainen hallinta. Henkilökohtaisesti tärkeäksi koin myös opinnäytetyöprosessin ajankäytöllisen hallinnan, mikä opetti työelämässä tarvittavaa ajanhallintaa projekteissa ennalta määriteltyjen aikataulullisten tavoitteiden saavuttamiseksi.

2 TIETOPERUSTA JA TYÖKALUT

2.1 Automaation perussuunnittelu ja toimintakaaviot

Automaation perussuunnitteluvaiheessa määritellään järjestelmän toiminnot, kuten mittaukset, ohjaukset ja säädöt. Perussuunnittelun tuloksena syntyvät automaation perusdokumentit, jotka kuvaavat järjestelmän toiminnan, mutta eivät vielä ole laitesidonnaisia. Edellä mainittuja dokumentteja ovat muun muassa:

- toimintakuvaukset
- PI-kaaviot
- toiminta- ja säätökaaviot
- mittapisteluettelot
- hälytysluettelot.

Ohjausautomaation yksityiskohtainen toiminta esitetään toiminta- ja säätökaavioissa sekä niitä tukevissa toimintakuvauksissa. Perussuunnittelun aikana koko automaation tehtävä tulee määriteltyä ja näin luodaan pohja toteutussuunnittelulle. (Häkkinen n.d., 12–14)

Automaation perussuunnitteluvaiheessa tuotettuihin dokumentteihin kuuluvan toimintakaavion tulee kuvata kohde toiminnalliselta näkökannalta toteutuksesta riippumattomalla tavalla. Toimintakaaviossa tulee esittää kohteen kaikki suhteet sen rakenneosien keskuudessa. Signaalien ensisijainen kulkusuunnan tulisi olla vasemmalta oikealle ja ylhäältä alas. Lisäksi toimintakaavio voi sisältää askeleiden ja siirtymien esityksiä. (SFS-EN 61082-1 2006, 51).

2.2 Visual Basic

Visual Basic on Microsoftin kehittämä ohjelmointikieli, joka on johdettu BASIC -kielestä. BASIC (Beginners All-Purpose Symbolic Instruction Code) on ohjelmoijien eniten käyttämä kieli tietokoneiden historiassa. ”Visual” puolestaan viittaa kielen tapaan rakentaa graafisia käyttöliittymiä. Siinä missä käyttöliittymän luominen monella muulla kielellä vaatii sivukaupalla ohjelmakoodia, Visual Basicissa ohjelmoija asettaa esirakennettuja olioita haluamalleen paikalle näytöllä. Näin ollen jo jonkin peruspiirto-ohjelman, kuten Paint, hallinta riittää antamaan tarpeelliset taidot käyttöliittymän suunnittelua varten. (Ohjelmoijan käsikirja 1999, 3)

Visual Basic -ohjelmointikielellä kirjoitetut ohjelmat ovat tapahtumaohjattuja. Tapahtumaohjatulla ohjelmalla tarkoitetaan tietokoneohjelmaa, joka käynnistyttyään odottaa ennalta määrättyä tapahtumaa ja tekee jotakin vastauksena sattuneeseen tapahtumaan. Tapahtuma voi olla esimerkiksi käyttäjän suorittama hiiren painikkeen napsautus. Tapahtumien järjestys määrää ohjelmakoodien suoritusjärjestyksen, joten suoritusjärjestys voi vaihdella periaatteessa ohjelman jokaisella suorituskerralla. Edellä kuvattu ohjelmakoodin suoritustapa erottaa tapahtumaohjatun ohjelmamallin perinteisistä, proseduraalisista ohjelmamalleista, joissa ohjelma suoritetaan joka kerta ennalta määrättyä polkua pitkin, aliohjelmia tarpeen mukaan kutsuen (Ohjelmoijan käsikirja 1999, 12).

Microsoft julkaisi Visual Basicin ensimmäisen version vuonna 1991. Se helpotti merkittävästi aikaisemmin hankalana pidettyä Windows-ohjelmointia, mahdollistaen toimivien ja tyylikkäiden ohjelmakokonaisuuksien luonnin varsin vähäisellä kokemuksella. Uusien versioiden myötä Visual Basiciin on tullut runsaasti parannuksia, jotka mahdollistavat laajojenkin ohjelmien tekemisen. (Laaksonen 2002.)

2.3 Visual Basic for Applications

Visual Basic for Applications eli VBA on Microsoftin Visual Basic 6 -ohjelmointikielen pohjalta kehittämä, alun perin Microsoft Office -tuotepaketin makro-ohjelmointia varten suunniteltu ohjelmointikieli (VBA Developer's Guide 2006, 2). Makrolla puolestaan tarkoitetaan tietokoneohjelmaa, joka toimii isäntäohjelman sisällä ja jonka avulla ohjataan ja täydennetään isäntäohjelman toimintoja (Ekonoja 2003).

VBA -kielisten makro-ohjelmien kirjoittamisessa on hyödyllistä huomioida kielen sisäiset ominaisuudet, jotta kirjoitettavista ohjelmista tulisi mahdollisimman suorituskykyisiä. Esimerkiksi numeeristen muuttujien tallentamiseen käytetyistä tietotyypeistä integer vaatii vähemmän muistia kuin long, mutta koska uusimmat VBA -versiot muuntavat integer -muuttujat ohjelmaa ajettaessa joka tapauksessa long -tyyppisiksi, on perusteltua ja ohjelman toiminnan nopeuttamisen kannalta järkevää määritellä pienempienkin lukujen datatyypiksi long (The Integer, Long and Byte Data Types 2011).

2.4 Visual Basic Editor ja debuggeri

Microsoft Visio 2010 -ohjelmaan sekä Microsoft Office -paketin ohjelmiin sisällytetty Visual Basic Editor eli VBE on VBA makro-ohjelmien kehitykseen tarkoitettu ohjelmointiympäristö. Siinä yhdistyvät useat eri toiminnot, kuten suunnittelu, ohjelmankirjoitus, käännös ja virheidenetsintä. Tällaista, eri toiminnot yhdistävää ohjelmointiympäristöä kutsutaan usein IDE:ksi. Lyhennys IDE perustuu englannin kielen sanoihin Integrated Developing Environment. (Ohjelmoijan käsikirja 1999, 13)

Debuggerilla tarkoitetaan ohjelmointiympäristön työkalua, jonka avulla on mahdollista analysoida koodin toimintaa ja etsiä siitä mahdollisesti löytyviä virheitä. Visual Basic Editoriin sisältyvä debuggeri tarkastaa koodin jokaisen rivin heti sen kirjoittamisen jälkeen ja ilmoittaa mikäli havaitsee syntaksivirheitä. Debuggerin ajon aikaisten virheiden paikantamiseen tarkoitetut toiminnot ovat puolestaan suunniteltu siten, että niiden avulla on mahdollista nähdä tarkalleen, miten koodin tietty kohta toimii

ohjelman ajon aikana. Suorittamalla koodia rivi kerrallaan ja monitoroimalla samalla muuttujien arvoja, on mahdollista löytää tarkalleen se ohjelmalause, jossa virhe tapahtuu. (Debugging Code 2011.)

2.5 Microsoft Visio 2010

2.5.1 Ohjelman esittely

Visio 2010 on uusin versio Microsoftin erilaisten kaavioiden piirtoa varten kehittämästä ohjelmasta. Visio tarjoaa sekä dynaamisia että datapohjaisia visualisointityökaluja sekä jakamistoimintoja, jotka mahdollistavat kaavioiden jakamisen reaaliaikaisesti tietoverkoissa. Ohjelman sisäänrakennettujen työkalujen ja valmiiden mallien avulla on mahdollista luoda muun muassa prosessi-, organisaatio- ja verkkokaavioita sekä mallintaa liiketoimintaprosesseja. (10 tärkeintä syytä tutustua Visio 2010 -ohjelmaan 2011.)

Visual Basic for Applications tulee sisäänrakennettuna Visio 2010 -ohjelman mukana. Sen avulla on mahdollista kirjoittaa makro-ohjelmia automatisoimaan Vision toimintoja. Vision avoin arkkitehtuuri paljastaa objektimallin VBA:lle ja tarjoaa näin tarvittavan rajapinnan automatisoinnin toteutukselle. Visio tallentaa makrot osaksi piirrosdokumenttia ja makro käyttää ajonaikaisena muistina samaa muistialuetta Vision kanssa. Edellä mainittu muistinvaraustapa on useimmissa tapauksissa tehokkaampi ohjelman toiminnan kannalta kuin ulkoisen, oman muistialueen varaavan, Visual Basic -ohjelman kirjoittaminen. (Walker 2007, 686–688)

2.5.2 Kirjastot ja masterit

Visio 2010 sisältää runsaasti erilaisiin käyttötarkoituksiin suunniteltuja symbolikirjastoja. Vision maailmassa näistä kirjastotiedostoista käytetään nimeä ”Stencil”, suomennettuna kaavain, ja niiden sisältämistä symboleista nimeä ”Master Shape”. Kun symbolia halutaan käyttää piirroksessa, se raahataan piirtoalueelle. Tällöin syntyy kopio, ”Shape”, joka perii kaikki masterin ominaisuudet (Walker 2007, 184). Koska

mikä tahansa Vision piirtoalueelle piirretty objekti on mahdollista tallentaa kirjastoon, käyttäjän on mahdollista luoda omia kirjastoja ja tallentaa niihin itse luomiaan objekteja. Tällöin käyttäjän piirtämistä objekteista tulee mastereita, joita on mahdollista käyttää myöhemmin tarpeen niin vaatiessa. Omien kirjastojen luominen on hyödyllistä silloin, kun käyttäjä tarvitsee toistuvasti jotakin tiettyä symbolia, eikä kyseistä symbolia sisälly Vision tarjoamiin kirjastoihin.

2.5.3 Masterien parametrointi

Monet Vision tarjoamien kirjastojen sisältämistä mastereista sisältävät parametreja. Visio käyttää näistä parametreista nimitystä ”Shape data”. Raahattaessa master piirtoalueelle nämä parametrit periytyvät masterista syntyvään kopioon, samoin kuin muutkin masterin ominaisuudet. Parametri voi olla piilotettu tai sen avulla voidaan esimerkiksi pyrkiä vaikuttamaan objektin ulkonäköön, jolloin parametri tulee kiinnittää kyseisen objektin grafiikkaan. Edellä mainitussa tapauksessa parametroinnin hyöty ilmenee siinä, että useita hyvin toistensa kaltaisia symboleja ei tarvitse tallentaa erillisinä mastereina, vaan piirrokseen liitetyn symbolin ulkonäkö konfiguroidaan halutuksi parametrin tai parametrien arvoja muuttamalla. (Walker 2007, 181–184)

3 TOTEUTUS

3.1 Microsoft Visio 2010 -ohjelmaan tutustuminen

Koska Microsoft Visio 2010 oli itselleni täysin tuntematon ja muutama Vision aikaisempi versio ainoastaan pintapuolisesti tuttu ohjelmisto, aloitin opinnäytetyöprosessin tutustumalla Vision toimintoihin. Tutustumisvaiheessa harjoittelin pääasiassa Vision piirtotyökalujen käyttöä, Vision valmiiden kirjastojen sisältämien symbolien käyttöä ja näiden ohjelmallista manipulointia VBA -koodin avulla. Viimeksi mainittu toimi hyvänä perehdyttäjänä Vision käyttämään objektimalliin, jonka tuntemuksen ja

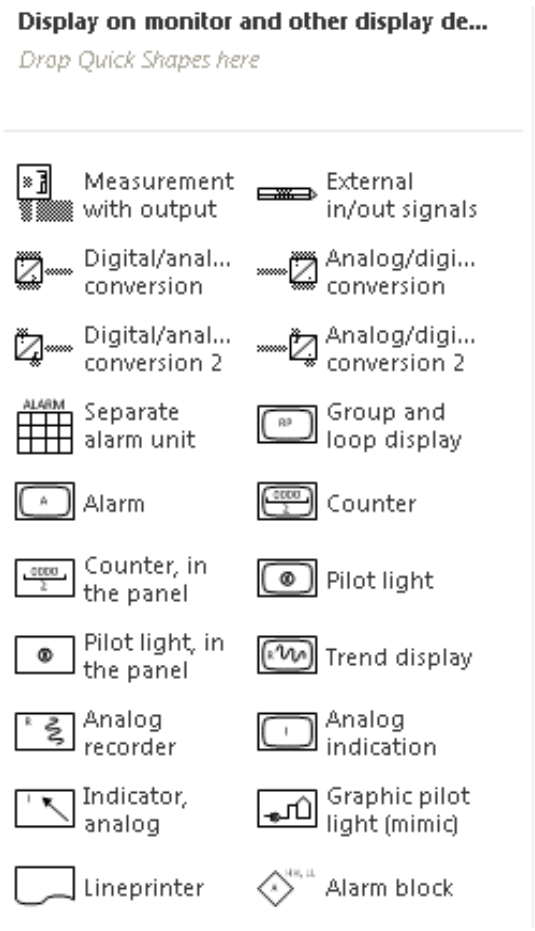
ymmärtämisen tiesin astuvan perustavan tärkeään rooliin opinnäytetyöhön sisältyneen ohjelmointityön aikana.

3.2 Symbolikirjastojen luominen

Perusasioiden harjoittelun jälkeen ryhdyin piirtämään symbolikirjastoja. Piirtämäni symbolit perustuvat standardeihin IEC 61131-3, IEC 60617, SFS 61082-1 ja ISO 81714-1. Toimeksiantaja on käyttänyt aikaisemmin samoihin standardeihin perustuvia symboleja muissa suunnitteluympäristöissä toteuttamissaan projekteissa.

Piirtämäni symbolit jaottelin siten, että tallensin samankaltaisia toimintoja kuvaavat symbolit samaan kirjastoon (ks. kuvio 1). Valmiita kirjastoja syntyi yhteensä seitsemän. Kirjastot nimesin siten, että nimi kuvaisi niiden sisältämiä symboleja. Kirjastoihin tallennettujen symbolien käyttö on jatkossa erittäin helppoa. Haluttu symboli raahataan hiiren avulla piirtoalueelle.

Kaikissa piirtämissäni symboleissa on liittimet helpottamassa signaaleita kuvaavien viivojen piirtämistä. Liittimet toimivat siten, että Visio liimaa piirrettävät yhdysviivat automaattisesti liittimiin, kun ohjelman asetuksista on optio ”Glue to Connection Points” valittuna. Tämä auttaa suuresti suunnittelijaa piirtotyön suorittamisessa, kun lopullisena tavoitteena on luoda yhdenmukaisia, siistejä ja helposti tulkittavia kaavioita.



KUVIO 1. Esimerkki opinnäytetyössä luodusta symbolikirjastosta.

3.3 Pohjakuvien luominen ja tagit

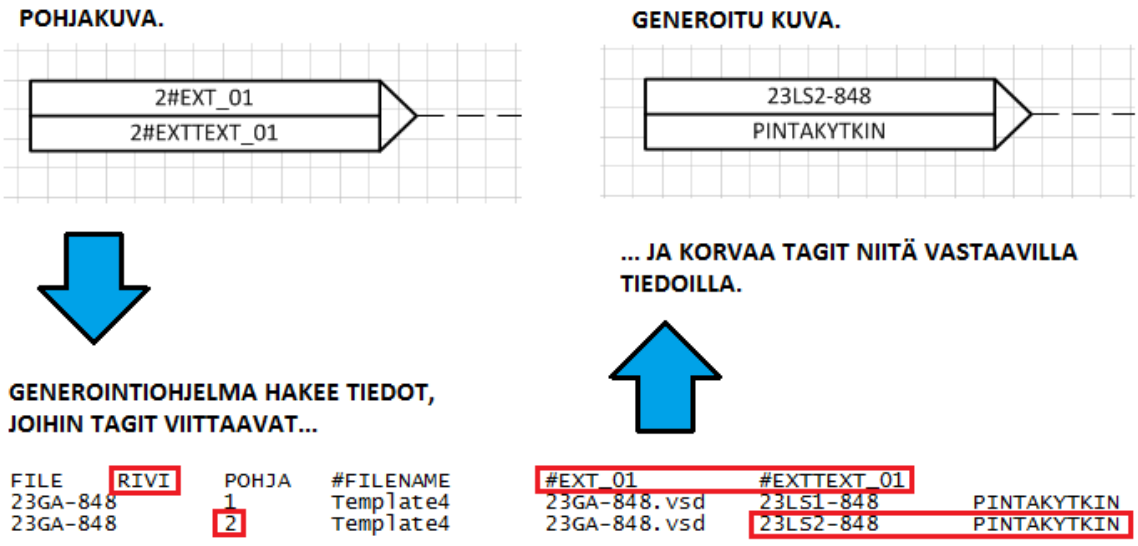
Jotta toimintakaavioiden generoiminen olisi mahdollista, on ensin piirrettävä tavoitteena olevaa kaaviota visuaaliselta ilmeeltään vastaava pohjakuva. Pohjakuvan piirtämiseen käytetään Vision piirtotyökaluja sekä opinnäytetyössä luotujen kirjastojen sisältämiä symboleja. Kaikki muuttuva informaatio, kuten esimerkiksi ulkoisten tulojen ja lähtöjen tunnukset ja otsikkotaulun tiedot merkitään pohjakuvaan tageilla. Tagit ovat tämän opinnäytetyön toteutusmallissa tekstikenttiä, joiden muoto on ennalta määrätty siten, että niiden avulla on generointivaiheessa mahdollista hakea haluttu lähtötieto.

Kokeilin tagien toteutusta pohjakuvan piirtämiseen käytettävien symbolien parametreina, mutta se johti lopullista toteutusta huomattavasti suurempiin tiedostokokoihin niin pohjakuvien kuin myös generoitujen kaavioiden osalta. Monessa tapauksessa

yhden kaavion koko kasvoi jopa kymmenkertaiseksi verrattuna ilman parametroituja objekteja tehtyyn kuvaan. Näin suuri ero tiedostokoossa aiheuttaa valtavaa tallennustilan tuhlausta käytännön työelämän projekteissa, joissa kaavioita tuotetaan tavallisesti nelinumeroinen lukumäärä.

Kokeilu paljasti myös muita heikkouksia, sillä generoinnin toiminta oli parametroitujen objektien mallilla tässä opinnäytetyössä esiteltyä käytäntöä huomattavasti hitaampaa. Uusien symbolien ja pohjakuvien luominen olisi myös ollut huomattavasti toteuttamaani käytäntöä monimutkaisempaa ja siten myös hitaampaa. Kaikkien mainitsemieni argumenttien perusteella tarkasteltuna tekstikenttien käyttäminen tageina on huomattavasti parametrointia järkevämpi ja kustannustehokkaampi toimintamalli.

Tagi toimii siten, että generointiohjelma korvaa sen lähtötiedolla, johon kyseinen tagi viittaa (ks. kuvio 2). Toimeksiantajan CAD-suunnittelussa vakiintuneiden käytäntöjen mukaan kaikki pohjakuvan tagit ovat muotoa X#YYY, jossa X on rivi, jolta lähtötietoja haetaan ja #YYY lähtötiedot sisältävän taulukon sarakkeen nimi. Siten esimerkiksi 2#EXT_01 tarkoittaa, että tagi korvataan sen kentän arvolla, joka löytyy lähtötiedoista kyseisen kaavion kohdalta riviltä 2 sarakkeesta #EXT_01.



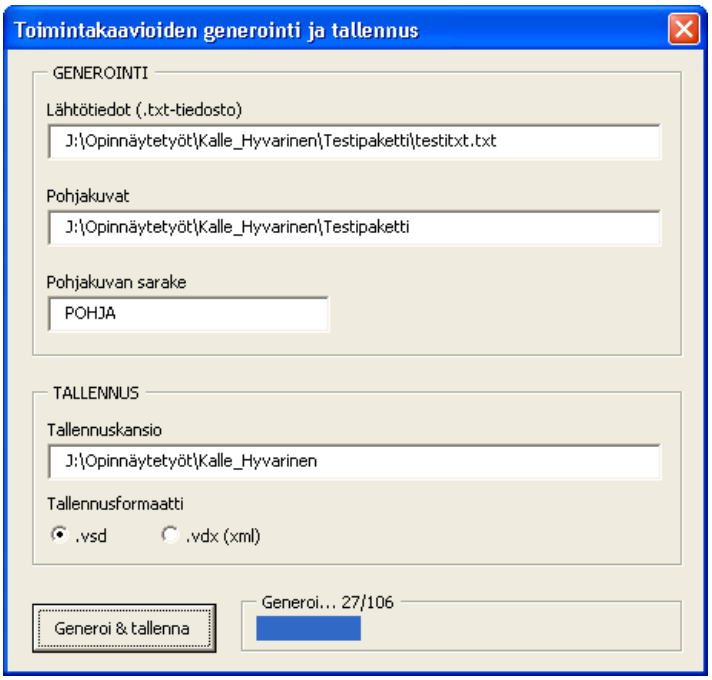
KUVIO 2. Esimerkki tagien toiminnasta kuvia generoitaessa.

Laadin opinnäytetyöprosessin aikana toimeksiantajaa varten valmiita mallipohjakuvia, joita käytin myös opinnäytetyöhön sisältyneiden ohjelmallisten toimintojen testaamisessa. Näitä pohjakuvia muokkaamalla uusien pohjakuvien luominen on jatkossa helpompaa ja ennen kaikkea nopeampaa. Sisällytin mallipohjiin myös yhden ainoastaan otsikkotaulut sisältävän pohjan, koska tilanne, jossa mikään olemassa olevista mallipohjista ei muistuta alkuunkaan tavoitteena olevaa pohjakuvaa, on käytännössä väistämätön.

3.4 Toimintakaavioiden generointi

3.4.1 Generointimakron toteutus

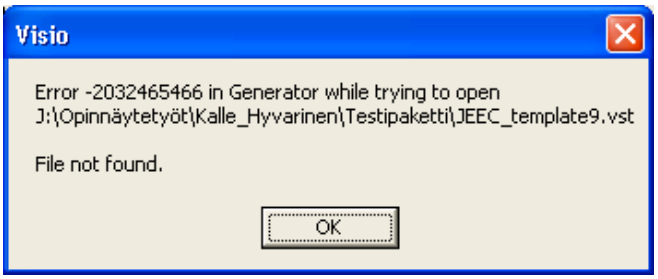
Toimintakaavioiden generointi on toteutettu tässä opinnäytetyössä VBA -kielellä kirjoitetun makro-ohjelman avulla. Generointimakro sisältyy VBA-projektiin, joka on tallennettu Vision piirrosdokumenttiin. Käyttäjä avaa ensin Microsoft Visio 2010 -ohjelman, jonka jälkeen hän avaa projektin sisältävän dokumentin. Kaikki muut mahdollisesti auki olevat dokumentit on suljettava, jotta makro toimisi oikein. Makron käyttöliittymä (ks. kuvio 3) toimii rajapintana käyttäjän ja ohjelmakoodin välillä.



KUVIO 3. Generointimakron käyttöliittymä ohjelman suorittaessa generointia.

Itse ohjelmointiprosessin aloitin laatimalla vuokaavion (ks. liite 1) ohjelmakoodin kirjoittamistyön helpottamiseksi ja ongelmanratkaisun jäsentelemisen tueksi. Vuokaaviolla tarkoitetaan kaaviomallia, joka on semaattinen esitys siitä, miten prosessin eri osat ovat yhteydessä toisiinsa (SuomiSanakirja 2011). Vuokaavio laaditaan lähes poikkeuksetta jokaisen ohjelmointiprojektin suunnitteluvaiheessa ja se sisällytetään tavallisesti myös projektin loppudokumentaatioon.

Ohjelmointiprosessin lopputuotteena syntyi makro, jonka toiminnallisuus on toteutettu aliohjelmissa, joita kutsutaan pääohjelmassa (ks. liite 2). Virheenkäsittely suoritetaan kussakin aliohjelmassa, mikäli ajonaikaisen virheen syntyminen on mahdollinen kyseisen aliohjelman suorittamisen aikana. Tällaisia ovat esimerkiksi kaikki tiedostoja käsittelevät aliohjelmat. Mikäli havaitaan virhe, siitä ilmoitetaan käyttäjälle (ks. kuvio 4), jonka jälkeen virheen aiheuttaneen aliohjelman suoritus lopetetaan. Seuraavissa luvuissa kuvataan aliohjelmien suorittamia toimintoja.



KUVIO 4. Esimerkki ajonaikaisen virheen aiheuttamasta virheilmoituksesta.

3.4.2 Lähtötiedot

Lähtötietojen lukeminen ohjelman muistiin

Ohjelma lukee lähtötiedot käyttäjän sille syötteenä antamasta Unicode -koodatusta tekstitiedostosta. Ensimmäiseksi käyttäjän syötteet tarkastetaan. Mikäli tarkastuksessa ilmenee puuttuvia tai virheellisiä syötteitä, tästä annetaan käyttäjälle samankaltainen ilmoitus kuin virheenkäsittelyn yhteydessä, mutta nyt koko ohjelman suoritus lopetetaan. Kun syötteiden asianmukaisuus on todettu, lähtötiedot luetaan ohjelman ajonaikaisen työmuistin kaksiulotteiseen, String-datatyypin alkioita sisältävään taulukkoon (ks.liite 3). Näin kovalevyn käsittelyn tarve ohjelman ajon aikana vähenee, mikä puolestaan nopeuttaa ohjelman toimintaa huomattavasti.

Lähtötietojen käsittely

Lähtötietojen käsittely on toteutettu aliohjelmissa, jotka suorittavat hakuja työmuistiin talletetusta taulukosta. Näiden hakutoimintojen avulla taulukosta palautetaan arvoja, joita tarvitaan kaavioiden generoinnissa.

3.4.3 Generointiprosessi

Aliohjelmien kutsuminen toistolauseessa

Lähtötiedot luetaan ohjelman ajon aikana vain kerran. Generointi on puolestaan toteutettu vakiomodulien aliohjelmilla, joita kutsutaan toistolauseessa niin monta kertaa kuin lähtötiedoissa on generoitavia kaavioita. Generointiprosessin eteneminen näytetään käyttäjälle käyttöliittymän alalaitaan avautuvassa palkissa, joka ilmoittaa prosessin etenemisen sekä numeerisesti että graafisesti. Palkin esittämä informaatio rauhoittaa käyttäjää ja vakuuttaa tämän siitä, että ohjelma ei ole kaatunut, vaikka sen suorittaminen kestäisikin kauan. Edellä mainittu tilanne on ajankohtainen etenkin silloin, kun generoitavien kaavioiden lukumäärä on suuri. Palkin päivittävää aliohjelmaa kutsutaan toistolauseen jokaisen kierroksen alussa. Seuraavissa kappaleissa kuvataan muiden toistolauseessa kutsuttavien aliohjelmien toiminta.

Pohjakuvan avaaminen

Ohjelma etsii pohjakuvan käyttäjän antamasta, pohjakuvatiedostot sisältävästä kansiosta. Käyttäjä antaa syötteenä myös sen lähtötietotaulukon sarakkeen nimen, josta pohjakuvatiedoston nimi löytyy. Näiden tietojen avulla ohjelma etsii kansiosta generoitavaa kaaviota vastaavan pohjakuvan ja avaa pohjakuvatiedoston generointia varten.

Generointi

Koska Vision piirroksissa voi olla useampi välilehti, generoinnin suorittava aliohjelma (ks. liite 4) käy läpi kaikki pohjakuvan välilehdet. Edelleen aliohjelma tutkii kaikki välilehdeltä löytyvät objektit. Visiossa tekstikenttä on objekti, johon ei sisälly mitään muuta grafiikkaa kuin merkkijono. Mikäli tämän merkkijonon arvo vastaa jotakin lähtötietotaulukon saraketta, korvataan merkkijono generoitavan kaavion tiedot sisältävältä riviltä löytyvällä vastaavan sarakkeen arvolla. Näin siis luvussa 3.3 esitellyt tagit

toimivat käytännössä. Kun kaikki pohjakuvan välilehdet ja kaikki niiltä löytyneet objektit on käyty aliohjelman toimesta läpi, on syntynyt valmis toimintakaavio.

Tallennus

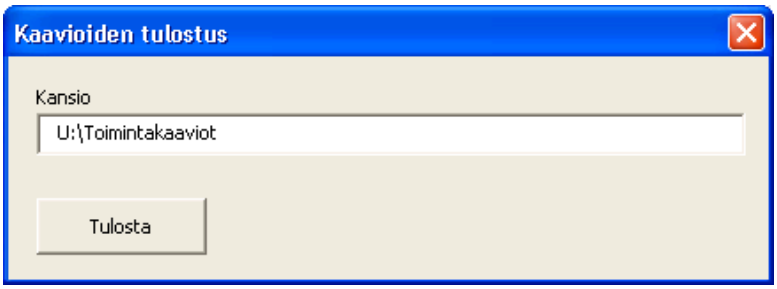
Valmiin toimintakaavion tallennus on toteutettu aliohjelmalla, joka tallentaa kaavion käyttäjän syötteenä antamaan kansioon. Tiedosto nimetään lähtötietotaulukon ensimmäisen sarakkeen arvon mukaan. Sarakkeen arvo luetaan luonnollisesti siltä riviltä, miltä kaavioon haettiin informaatiota generoinnin yhteydessä. Tallennettavan kaavion mahdolliset makrot poistetaan ohjelmallisesti, jotta tiedostokoko saataisiin pienemmäksi, eivätkä pohjakuvaan mahdollisesti tallennetut makrot lähtisi vahingossa dokumentin mukana tilaajalle. Tallennetun kaavion tiedostoformaatti määräytyy käyttäjän valinnan mukaan. Vaihtoehtoina ovat joko vsd tai vdx. Ensin mainittu tarkoittaa, että kaavio tallennetaan Vision omassa kuvatiedostojen tallentamiseen tarkoitetussa formaatissa. Jälkimmäinen puolestaan, että kaavio tallennetaan XML-formaatissa.

3.4.3 Testaus

Testasin generointimakron toimintaa koko ohjelmointiprosessin ajan rakentamani testipaketin avulla. Testipakettiin kuului kaksi erilaista kuvitteellisia lähtötietoja sisältävää tekstitiedostoa ja muutama pohjakuva. Kirjoitettuani pienen pätkän ohjelmakoodia, tarkastin sen toiminnan testipaketin ja ohjelmointiympäristön debuggerin monitorointitoiminnon avulla. Mikäli koodi ei toiminut toivotulla tavalla, debuggeri auttoi virheiden etsimisessä ja korjaamisessa. Vaikka testaaminen muodostaa hyvin pienen osan opinnäytetyön kirjallisesta raportoinnista, se haukkasi käytännön tasolla hyvin suuren prosentin työhön käyttämästäni ajasta.

3.5 Toimintakaavioiden tulostus

Tulostusmakro on tallennettu samaan dokumenttiin generointimakron kanssa ja kirjoitettu myös VBA -ohjelmointikielellä. Makron käyttö tapahtuu aivan samoin kuin generointimakron tapauksessa. Ohjelmointiprosessin tukena käytin myös tässä tapauksessa piirtämääni vuokaaviota (ks. liite 4). Makrolla pystytään tulostamaan kaikki Vision tukemissa tiedostoformaateissa tallennetut dokumentit käyttäjän syötteenä antamasta kansiosta. Käyttöliittymä (ks. kuvio 5) toimii rajapintana käyttäjän ja ohjelmakoodin välillä.



KUVIO 5. Tulostusmakron käyttöliittymä.

Makron toiminnallisuus on toteutettu aliohjelmassa, jota kutsutaan pääohjelmasta käsin. Generointiin verrattuna tulostus on perin yksinkertaisesti toteutettu. Käyttäjä antaa syötteenä kansion, jonka sisältämät dokumentit hän haluaa tulostaa. Makro hakee ensin kaikkien kansiossa olevien tiedostojen nimet ja tarkastaa tiedostopäätteestä, että kyseessä on Vision tukema tiedostomuoto. Tämän jälkeen tarkastuksessa hyväksytyt tiedostot avataan yksi kerrallaan, tulostetaan välilehti kerrallaan ja lopuksi tiedosto suljetaan. Näin menetellen makro käy läpi kaikki kansiossa olevat, tarkastuksessa hyväksytyt tiedostot.

Tulostusmakron ohjelmointiprosessin aikaisen testauksen suoritin debuggerin avulla. Tällöin ohitin koodirivin, joka lähettää kulloinkin käsiteltävän välilehden järjestelmän oletusprintterin tulostettavaksi ja seurasin ainoastaan debuggerin monitorointitoiminnon avulla, että koodi toimii halutulla tavalla. Käytin edellä kuvattua testaustapaa tarpeettoman paperin tuhlauksen välttämiseksi testauksen yhteydessä. Kun olin havainnut koodin toimivaksi debuggerin avulla, testasin sen toimintaa käytännössä te-

kemällä testipakettiin kansion, johon sijoitin kolme toimintakaavioiden pohjakuvaa. Makro suoriutui testistä moitteettomasti tulostaen pohjakuvat järjestelmän oletus-printteriä käyttäen.

3.6. Käyttöohje

Opinnäytetyön toimeksianto sisälsi käyttöohjeen kirjoittamisen. Laadin käyttöohjeen siten, että sen avulla ohjeen lukijan olisi mahdollista löytää kaikki Vision toiminnot, joita käyttämällä uusien pohjakuvien ja symbolien piirtäminen on mahdollista. Käyttöohje opastaa tavallisimpien piirtotyökalujen käytössä ja tagien nimeämisessä, perehdyttää pintapuolisesti Vision objektimalliin sekä neuvoo makrojen käytössä siltä osin, kun se on tarpeellista. Käyttöohjetta kirjoittaessa pyrin rajaamaan aiheet siten, että se palvelisi parhaiten tulevaa kohderyhmää, automaatiosuunnittelijoita. Koska kohderyhmän kokemus ja monimutkaistenkin tietokoneohjelmien hallinta on korkealla tasolla, käyttöohjeessa oli turhaa puuttua niin sanottuihin itsestäänselvyyksiin tai Windows -ohjelmien käytön alkeisiin.

4 TULOKSET JA KEHITYSKOHTEET

4.1 Tulokset

Opinnäytetyön tuloksia arvioitaessa ei ole mahdollista suorittaa vertailua aikaisempiin projekteihin, sillä toimeksiantaja ei ole aikaisemmin suorittamissaan projekteissa tuottanut automaation toimintakaavioita Microsoft Visio 2010 -ohjelmalla. Generointi- ja tulostusmakrojen aiheuttama suunnitteluprosessin nopeutuminen on sitä vastoin mahdollista arvioida teoriassa. Tällöin vertailukohtana voidaan pitää tilannetta, jossa kaikki makrojen ohjelmallisesti suorittamat toimenpiteet tehtäisiin suunnittelijan toimesta manuaalisesti.

Generointimakron vaikutuksia suunnitteluprosessin ajankäytön tehostajana arvioitaessa tarkastelen tapauksia, joissa piirrettäisiin 106 kaaviota käsin tai tehtäisiin yksi pohjakuva, jonka pohjalta generoitaisiin 106 kaaviota. Jokainen kaavio kuvaisi samalla periaatteella tapahtuvaa toimintaa, esimerkiksi mittausta, joten jokaisen kaavion grafiikka olisi sama, ainoastaan positiotunnukset muuttuisivat. Tunnusten muuttamisen ja siihen liittyvän tiedonhaun suunnittelija joutuisi siis manuaalisessa toimintamallissa suorittamaan kaavio kerrallaan käsipelillä.

Arvion tuloksia (ks. taulukko 1) tarkasteltaessa on otettava huomioon, että se on hyvin optimistinen manuaalisesti suoritetun piirto- ja tiedonhakunopeuden suhteen. Arviossa esitettyihin arvoihin päästäkseen suunnittelijan tulee olla harjaantunut Visio on ja tiedonhakuprosessiin liittyvien ohjelmien tai muiden tietolähteiden käyttäjä. Käytännössä aikaa kuluu eri vaiheisiin suurella todennäköisyydellä enemmän. Generointiprosessiin kuluva aika on puolestaan mitattu kokeellisesti makron toiminnan testauksen yhteydessä.

TAULUKKO 1. Arvio toimintakaavioiden (106 kaaviota) luomiseen käytetystä ajasta manuaalisesti piirrettynä ja generointimakrolla suoritettuna.

Kuvat	Manuaalisesti	Makrolla
Ensimmäinen	1 h	1 h
Loput yhteensä	0,5 h · 105	~ 3 min
Kaikki yhteensä	53,5 h	~ 1 h 3 min

Taulukossa 1 esitetyn arvion perusteella voidaan todeta, että opinnäytetyön tuloksena syntyneen generointimakron käyttäminen laadittaessa toimintakaavioita Microsoft Visio 2010 -ohjelmalla, nopeuttaa suunnitteluprosessia huomattavasti. Arvio osoittaa, että makron avulla voidaan suorittaa sama työ reilussa tunnissa, minkä tekemiseen suunnittelijalta kuluisi ilman makroa lähes seitsemän työpäivää, mikäli oletetaan yhden työpäivän olevan kahdeksan tunnin mittainen.

Toimintakaavioiden tulostuksen osalta makron aikaansaama ajansäästöä tulee verrata tilanteeseen, jossa suunnittelija avaa tulostettavan tiedoston Visio 2010 -ohjelmassa ja tulostaa sen ohjelman tulostustoimintoa käyttäen. Tulostusmakro lä-

hettää nopeammin kaikki 106 kaaviota tulostimelle kuin mitä suunnittelijalta kuluu yhden kaavion avaamiseen ja tulostamiseen. Rajoittavia tekijöitä tulostusmakron toimintanopeudelle ovat pääasiassa tiedonsiirtonopeus ohjelmaa pyörittävästä koneesta tulostimelle sekä tulostustyöhön käytettävän järjestelmän oletustulostimen kapasiteetti.

4.2 Kehityskohteet

Opinnäytetyön tuloksina syntyneiden VBA -makro-ohjelmien toimintojen jatkokehitykselle on olemassa mahdollisuudet kenellä tahansa toimeksiantajayrityksen osoittamalla, Visual Basic -ohjelmoinnin hallitsevalla henkilöllä. Koodin kommentoinnissa olen pyrkinyt dokumentoimaan koodin toiminnan siten, että osaava Visual Basic ohjelmoija saa nopeasti kiinni koodin toimintamallista ja pystyy kehittämään sitä haluttuun suuntaan.

Pohjakuvien ja symbolikirjastojen suhteen kehitykselle ei ole olemassa mitään rajoja, vaan niitä voidaan laatia lisää tapauskohtaisesti. Opinnäytetyössä luotujen symbolikirjastojen sisältämät symbolit noudattavat alalla sovellettuja standardeja, mutta mikäli ja kun niiden ulkoasu ei vastaa tilaajan vaatimuksia, uusia symbolikirjastoja on helppo piirtää lisää tulevaisuudessa.

5 POHDINTA

Opinnäytetyö tarjosi hienon mahdollisuuden perehtyä ja syventyä automaation suunnittelua tukevien työkalujen suunnitteluun. Saavutettu lopputulos vastaa mielestäni hyvin niin toimeksiantajan asettamia kuin myös itse itselleni asettamiani henkilökohtaisia tavoitteita. Symbolikirjastot ovat helposti käytettävissä ja laajennettavissa. Makrot puolestaan ovat käyttöliittymältään pelkistettyjä ja siten helppokäyt-

töisiä. Mikäli tulevaisuudessa ilmenee ongelmia, niin käyttöohje, jos ei tarjoa valmista ratkaisua, ohjaa ainakin haluttuun suuntaan.

Opinnäytetyöhön sisältynyt sovellussuunnittelu vahvisti puolestaan Visual Basic -ohjelmointikielen osaamistani ja opetti itselleni aiemmin tuntemattoman VBA -kielen tarjoamia ohjelmakohtaisia VB6 -kielen laajennuksia. Sovellussuunnittelussa vastaan tulevien ongelmien ratkaisukykyä tuli puolestaan kehitettyä päivittäin ohjelmointityön aikana. Kohdatut ongelmat opettivat kärsivällisyyttä ja pakottivat etsimään omia ratkaisumalleja, sillä vaikka Visual Basic -aiheista materiaalia ja jopa valmista koodia on runsaasti saatavilla, sen soveltuvuus käsillä olevaan ongelmaan on aina iso kysymysmerkki. Vastauksen löytäminen juuri kyseiseen ongelmaan vaatii käytännössä aina tapauskohtaista soveltamista, toisinaan enemmän, toisinaan vähemmän. Ja kun toimiva ratkaisu on löydetty, prosessi ei pysähdy vielä siihenkään, sillä lähes aina on mahdollista löytää nopeammin toimiva ja ohjelmaa pyörittävää konetta vähemmän kuormittava ratkaisu.

Kaiken kaikkiaan koen opinnäytetyön saavuttaneen sille asetetut tavoitteet ja toivon sen tuloksista olevan hyötyä toimeksiantajayritykselle tulevaisuudessa. Henkilökohtaisesti olen vahvasti sitä mieltä, että opinnäytetyöprosessi tuki ammatillista kasvuani siltä edellytetyllä tavalla.

LÄHTEET

10 tärkeintä syytä tutustua Visio 2010 -ohjelmaan. 2011. Office.com verkkosivut. Viitattu 10.9.2011. http://office.microsoft.com/fi-fi, Tuotteet, Visio, 10 tärkeintä syytä tutustua Visio 2010 -ohjelmaan

Debugging Code. 2011. MSDN Library. Viitattu 11.9.2011. http://msdn.microsoft.com/library, Office Development, Microsoft Office XP, Office XP Developer, Microsoft Office XP Developer, Developing Office Developer Applications, Debugging and Error Handling

Ekonoja, A. 2003. Makrot. Henkilökohtaisen tiedonhallinnan perusteet kurssimateriaali. Jyväskylän yliopiston IT-tiedekunta ja avoin yliopisto. Julkaisuja. Viitattu 9.9.2011. http://appro.mit.jyu.fi, henkilökohtaisen tiedonhallinnan perusteet, moniste, makrot

Häkkinen, V-M. n.d. Automaatiosuunnittelu, johdanto. Jyväskylän ammattikorkeakoulun automaatiosuunnittelu opintojakson luentomateriaali.

JEEC Oy. 2011. Kotisivut. Viitattu 8.9.2011. http://www.jeec.fi

Laaksonen, A. 2002. Visual Basic -opas. Viitattu 12.9.2011. http://www.ohjelmointiputka.net, Oppaat, Visual Basic -opas

Microsoft Visual Basic 6.0 Ohjelmoijan käsikirja. 1999. Toim. Microsoft Corporation. Jyväskylä: Gummerus Kirjapaino Oy.

SFS-EN 61082-1. 2006. Sähkötekniikassa käytettävien dokumenttien laatiminen. 2. p. Helsinki: Suomen Standardisoimisliitto SFS.

SuomiSanakirja. 2011. Suomen kielen sanakirja netissä. Viitattu 12.9.2011. http://suomisanakirja.fi

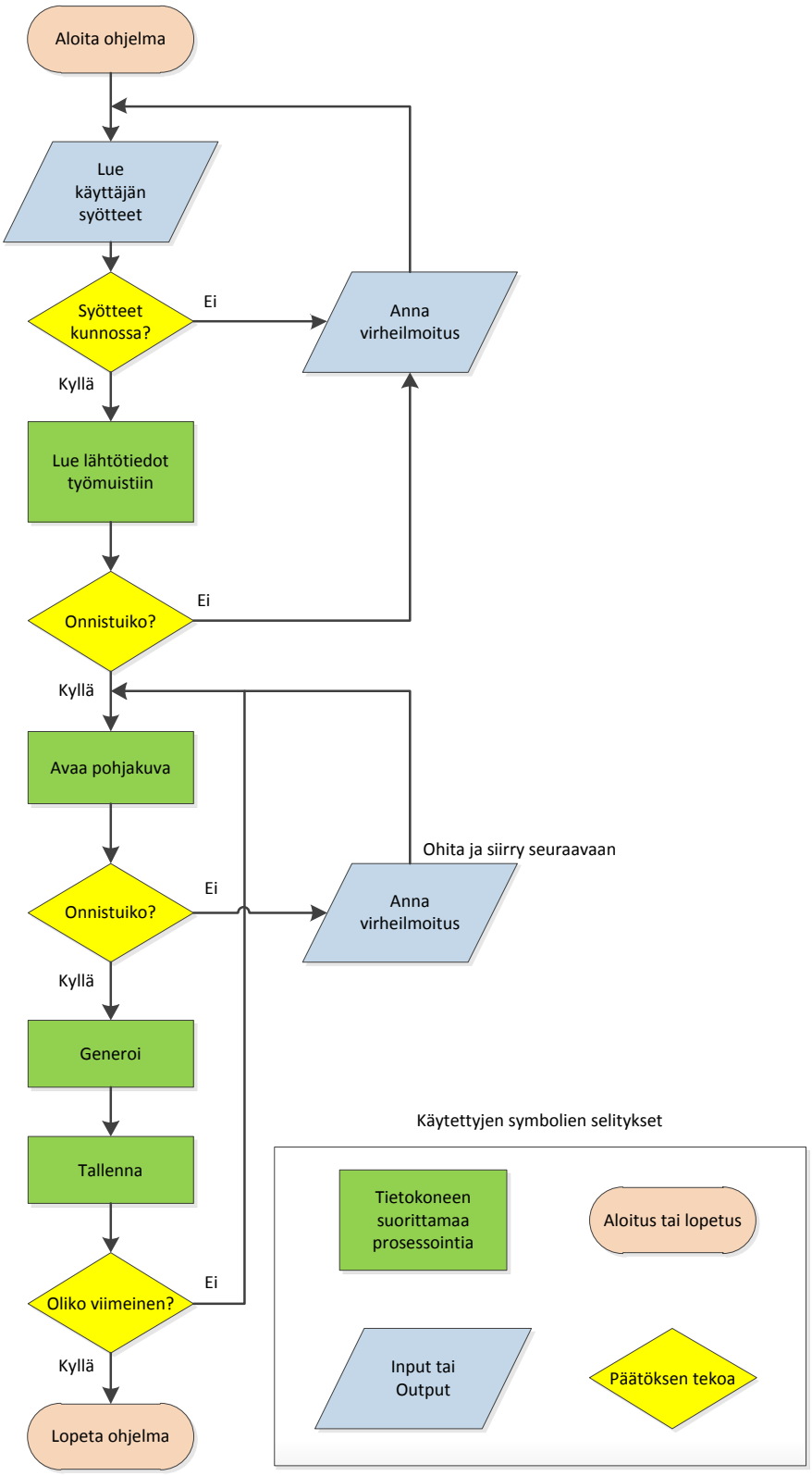
The Integer, Long and Byte Data Types. 2011. MSDN Library. Viitattu 11.9.2011. http://msdn.microsoft.com/library, Office Development, Microsoft Office XP, Office XP Developer, Microsoft Office XP Developer, Developing Office Developer Applications, Developing Applications with Microsoft Office, Developing Office Applications Using VBA, Getting the Most Out of Visual Basic for Applications, Working with Numbers

VBA Developer's Guide Release 9.0. 2006. Microsoft.

Walker, M. 2007. Microsoft Office Visio 2007 Inside Out. Redmond, Washington: Microsoft Press.

LIITTEET

LIITE 1. Generointimakron vuokaavio.



LIITE 2. Generointimakron pääohjelman VBA-koodi

```
Option Explicit
Option Compare Text

Private Sub GenAndSaveButton_Click()

    Dim fileReaderObj As New FileReader
    Dim searchObj As New Search
    Dim checkerObj As New Checker
    Dim progressBarObj As New ProgressBar
    Dim generatorObj As New Generator

    Dim txtFilePath As String
    Dim theArray() As String
    Dim theRows As Long
    Dim theCols As Long
    Dim arrayOfFiles() As String
    Dim nbrOfFiles As Long
    Dim firstRowAsArray() As String
    Dim templatesPath As String
    Dim templateIdtf As String
    Dim savefolder As String
    Dim saveAsVsd As Boolean
    Dim saveAsVdx As Boolean
    Dim i As Long

    On Error GoTo ErrorHandler

    ' luetaan muuttujille arvoja käyttöliittymästä
    txtFilePath = TxtFilePathTextBox.Text
    templatesPath = TemplatesTextBox.Text
    templateIdtf = TemplateColTextBox.Text
```

```
savefolder = SaveFolderTextBox.Text
saveAsVsd = SaveAsVsdOptionButton.Value
saveAsVdx = SaveAsVdxOptionButton.Value

' tarkastetaan, että ohjelman tarvitsemat kentät on täytetty
If checkerObj.checkStr(txtFilePath) = False Then
   MsgBox ("Tekstitiedoston polkua ei ole annettu.")
   Exit Sub
End If

If checkerObj.checkStr(templatesPath) = False Then
   MsgBox ("Pohjakuvien kansiota ei ole annettu.")
   Exit Sub
End If

If checkerObj.checkStr(templateIdtf) = False Then
   MsgBox ("Pohjakuvan saraketta ei ole annettu.")
   Exit Sub
End If

If checkerObj.checkStr(savefolder) = False Then
   MsgBox ("Tallennuskansiota ei ole annettu.")
   Exit Sub
End If

If checkerObj.checkExts(saveAsVsd, saveAsVdx) = False Then
   MsgBox ("Tallennusmuotoa ei ole valittu.")
   Exit Sub
End If

' luetaan lähtötiedot Filereader-luokan avulla
With fileReaderObj
   .readFile (txtFilePath)
```

```
    theArray = .getMyArray
    theRows = .getMyRows
    theCols = .getMyCols
End With

' luetaan lisää arvoja lähtötiedoista Search-luokan avulla
With searchObj
    arrayOfFiles = .getFILEs(theArray, theRows)
    nbrOfFiles = .getNbrOfFILEs(arrayOfFiles)
    firstRowAsArray = .getFirstRow(theArray, theCols)
End With

' tarkastetaan, että pohjakuvan tunniste löytyy lähtötiedoista
If checkerObj.checkTemplateIdtf(firstRowAsArray, templateIdtf) = False Then
    MsgBox ("Pohjakuvan saraketta ei löydy.")
    Exit Sub
End If

' progress-barin käynnistys
progressBarObj.showGenProgressBar

' generointi ja tallennus kuva kerrallaan
For i = 0 To UBound(arrayOfFiles)
    Dim RIVIT As Long

    ' progress-palkin päivitys
    progressBarObj.updateGenProgressBar nbrOfFiles, i

    ' generoitavat rivit
    RIVIT = searchObj.getRIVIT(theArray, theRows, arrayOfFiles(i))

    ' generointi Generator-luokan avulla
    With generatorObj
```

```
            .openTemplate theArray, theRows, theCols, templateIdtf, _
            templatesPath, arrayOfFiles(i)
            .generate theArray, firstRowAsArray, arrayOfFiles(i), theRows, _
            theCols, RIVIT, saveAsVsd, saveAsVdx
            .saveAsDrawing savefolder, arrayOfFiles(i), _
            saveAsVsd, saveAsVdx
        End With
    Next i

    progressBarObj.closeGenProgressBar

    Set fileReaderObj = Nothing
    Set searchObj = Nothing
    Set checkerObj = Nothing
    Set progressBarObj = Nothing
    Set generatorObj = Nothing

    MsgBox ("Valmis!")

ExitProcedure:
    Exit Sub

ErrorHandler:
    Resume ExitProcedure

End Sub
```

LIITE 3. Lähtötietojen lukemisen suorittavan aliohjelman VBA-koodi

```
Option Explicit

Private myArray() As String
Private myRows As Long
Private myCols As Long

'**********
' Lukee UNICODE txt-tiedoston rivi kerrallaan 1-ulotteiseen
' tmpTaulukkoon,jonka jälkeen pilkkoo rivit tabulaattorin lyöntien
' mukaan ja sijoittaa pilkotut rivit 2-ulotteiseen taulukkoon.
'**********
Public Sub readFile(ByVal filePath As String)

   On Error GoTo ErrorHandler

   Dim oFSO As New FileSystemObject
   Dim oFSTR As scripting.TextStream
   Dim tmpArray() As String
   Dim j As Long

   myRows = 0
   j = 0

   '**********
   ' Avataan tiedosto, mikäli se löytyi parametrina syötetystä
   ' kansiosta, luetaan se rivi kerrallaan ja sijoitetaan taulukkoon.
   ' Suljetaan tiedosto.
   '**********
   If oFSO.FileExists(filePath) = True Then
      Set oFSTR = oFSO.OpenTextFile(filePath, 1, 0, -1)
```

```
    Do While Not oFSTR.AtEndOfStream
        ReDim Preserve tmpArray(myRows)
        tmpArray(myRows) = oFSTR.ReadLine
        DoEvents
        myRows = myRows + 1
    Loop

    oFSTR.Close
Else
    MsgBox ("Lähtötietoja ei löydy.")
    Exit Sub
End If

'**********
' 1-dim taulukon rivien purku tabulaattori merkkien mukaan
' ja sijoitus 2-dim taulukkoon
' välivaiheiden suorittamisessa käytetty apumuuttujia
' row ja tmpArray2()
'**********
For j = 0 To myRows - 1
    Dim row As String
    Dim tmpArray2() As String
    Dim k As Long

    myCols = 0
    k = 0
    row = tmpArray(j)
    tmpArray2 = Split(row, vbTab)
    myCols = UBound(tmpArray2)

    ReDim Preserve myArray(myRows, myCols)
```

```
        For k = 0 To myCols
            myArray(j, k) = tmpArray2(k)
        Next k

    Next j

    Set oFSO = Nothing
    Set oFSTR = Nothing

ExitProcedure:
    Exit Sub

ErrorHandler:
    MsgBox ("Error " & Err.Number & ", " & Err.Description)
    Resume ExitProcedure

End Sub
```

LIITE 4. Toimintakaavion generoinnin suorittavan aliohjelman VBA-koodi

```
Public Sub generate(ByRef theArray() As String, ByRef firstRowAsArray() As String, _
ByVal rowidtf As String, ByVal rows As Long, ByVal cols As Long, _
ByVal RIVIT As Long, ByVal saveAsVsd As Boolean, ByVal saveAsVdx As Boolean)

    Dim appVisio As Visio.Application
    Dim docObj As Visio.Document
    Dim pagObj As Visio.Page
    Dim shpObj As Visio.Shape

    Dim searchObj As New Search
    Dim i As Long
    Dim j As Long
    Dim k As Long

    On Error GoTo ErrorHandler

    Set appVisio = GetObject(, "Visio.Application")
    Set docObj = appVisio.Documents.Item(2)

    ' käydään läpi dokumentin kaikki auki olevat sivut
    For i = 1 To docObj.Pages.Count
        Set pagObj = docObj.Pages.Item(i)

        ' käydään läpi kaikki sivun muodot
        For j = 1 To pagObj.Shapes.Count
        Set shpObj = pagObj.Shapes.Item(j)

            ' sivunumerojen asetus
            If shpObj.Text = "1#SHEET" Then
```

```
    shpObj.Text = i & "/" & docObj.Pages.Count
End If

' tiedoston nimen asetus
If shpObj.Text = "1#FILE_NAME" Then
    If saveAsVsd = True Then
        shpObj.Text = rowidtf & ".vsd"
    Else
        shpObj.Text = rowidtf & ".vdx"
    End If
End If

'**********
' Käydään läpi taulukon ensimmäinen rivi sarake kerrallaan
' ja tarkastetaan löytyykö sivulla olevasta muodosta sama string
' kuin taulukon sarakkeesta. Jos löytyy, asetetaan halutulta
' riviltä ko sarakkeen arvo vastaavan stringin arvoksi.
'**********
For k = 0 To cols
    Dim identifier As String
    identifier = firstRowAsArray(k)
    Dim iRows() As Long
    Dim iRow As Long
    Dim iCol As Long
    Dim strToAdd As String
    Dim RIVI As Long

    For RIVI = 1 To RIVIT
        If shpObj.Text = RIVI & identifier Then
            iRows = searchObj.getRows(theArray, rowidtf, rows)
            iRow = iRows(RIVI)
            iCol = searchObj.getCol(theArray, identifier, cols)
            strToAdd = theArray(iRow, iCol)
```

```
                shpObj.Text = strToAdd
            End If
        Next RIVI

    Next k

  Next j

 Next i

 Set appVisio = Nothing
 Set docObj = Nothing
 Set pagObj = Nothing
 Set shpObj = Nothing
 Set searchObj = Nothing

ExitProcedure:
 Exit Sub

ErrorHandler:
 MsgBox ("Error " & Err.Number & " in Generator" & vbCrLf _
 & "while trying to generate " & rowidtf & "." & Err.Description)
 Resume ExitProcedure

End Sub
```

LIITE 5. Tulostusmakron vuokaavio.