Oskari Mörsky

Instrumenttilaitteiden sijaintitietojen käsittely graafisnumeerisesti

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Energia- ja ympäristötekniikka

Insinöörityö

18.12.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Oskari Mörsky Instrumenttilaitteiden sijaintitietojen käsittely graafisnumeerisesti 33 sivua 18.12.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Energia- ja ympäristötekniikka
Ammatillinen pääaine	Energiatekniikka
Ohjaajat	Lehtori Tuomo Heikkinen Projektipäällikkö Jouni Paloniitty

Opinnäytetyön tarkoituksena oli suunnitella ja tuottaa työkalu, joka mahdollistaa instrumenttien sijaintitietojen graafisen ja numeerisen käsittelyn instrumenttikaapeliverkon optimoimiseksi. Työkalun avulla voidaan myös tarkastella instrumenttien sijoitusta erilaisin suodattimin, asettaa CAD-kuvaan position Z-koordinaatin ja laskea laitekaapeleiden pituudet. Tietoa kerätään sijoituskuvasta erilaisin kysymyksin jokainen instrumentti yksitellen, jonka jälkeen kerätty tieto siirtyy automaattisesti Excel-taulukkoon.

Työn tilaaja tarvitsee työkalun, jolla voidaan tarkastella kenttäkotelojakoa visuaalisesti.

Ohjelma tehtiin Visual Basic for Applications -ohjelmointikielellä. Kaikki opinnäytetyölle asetetut tilaajan vaatimukset ominaisuuksista saatiin toteutettua.

Avainsanat	AutoCAD, Excel, VBA, Ohjelmointi,

Author Title Number of Pages Date	Oskari Mörsky Instrument location data handling graphico-numerically 33 pages 18.12.2018
Degree	Bachelor of Engineering
Degree Programme	Energy and Environmental engineering
Professional Major	Energy Engineering
Instructors	Principal Lecturer Tuomo Lehtinen Project Manager Jouni Paloniitty

The aim of the thesis was to design and produce a program, which allows viewing graphically the location images of the instruments in AutoCAD. The tool also allows you to study the positioning of instruments with different filters, set the position of the Z coordinate in the CAD image and dimension the instrument cables. Information collected in reflecting a variety of questions, each instrument one by one, after which the collected data is transferred automatically to an Excel spreadsheet.

The commissioning company needs a tool that makes the field connection cabins more effective and easier to notice from drawings.

The program was executed in the Visual Basic for Applications programming language. Commissioning company requirements set for the thesis characteristics were achieved.

Keywords	AutoCAD, VBA, Programming, Excel

Sisällys

Lyhenteet

1 Johdanto 1

2 Insta Group 2

3 Instrumentointisuunnittelussa tuotetut dokumentit 2

3.1 Piirikaaviot 2
3.2 Kojeluettelo 3
3.3 Kenttäkaapeliluettelo 4
3.4 Instrumenttien sijoituspiirustus 5
3.5 Ristikytkentäkaappien kaaviot 5
3.6 PI-kaaviot 6

4 AutoCAD ja ohjelmointikielet 7

5 Visual Basic for Application 8

5.1 Visual Basic for Applicationsin perusteita 8
5.2 Ohjelmien perusfunktio ja rakenne 12
Arvojen määrittely muuttujille 13
5.3 Datatyypit 14
5.4 VBA:n perusfunktioita ja ohjausrakenteet 15
5.4.1 For-silmukka 15
5.4.2 Do Until/While 16
5.4.3 GoTo-rakenne 16
5.4.4 If-lause 17
5.4.5 Select Case 17
5.4.6 With... End 18
5.5 Virheiden käsittely 18
5.6 Koodiesimerkki 19

6 AutoLisp 20

6.1 Makrot ja skriptit 21

7 ObjectARX® 22

8 Opinnäytetyö 24

8.1 Lähtötilanne ja opinnäytetyön suunnittelu 24
8.2 AutoCADin ja Excelin välinen dataintegraation toteutus 24
8.3 Työkalun koodin perustoteutus 28

9 Työkalun kehitysajatukset, yhteenveto ja pohdinta 30

Lähteet 31

Lyhenteet ja käsitteet

AutoLisp	Ohjelmointikieli
ActiveX	COM:a mahdollistava käyttävä teknologia
C#	Ohjelmointikieli
C++	Ohjelmointikieli
CAD	Tietokoneavusteinen suunnittelu
COM	Component Object Model
Entity	Ryhmä erilaisista Autocadin objekteista
Exd	Räjähdyksen kestävä laite
Exi	Luonnostaan vaaraton laite
Handle	Yksilöivä tunnus objekteissa
K-kieli	Ohjelmointikieli
ObjectARX	Ohjelmointirajapinta
ObjectID	Väliaikainen yksilöivä tunnus
OLE-Automation	Ohjelmien välinen kommunikaatioprotolla
PHAKU	Excel-hakufunktio
SelectionSet	Objektien valinta
VB.Net	Ohjelmointikieli
VBA	Visual Basic for Applications lyhennettynä
Visual Basic for Applications	Ohjelmointikieli

1 Johdanto

Suunnitteluyhtiöissä on jatkuva tarve tehostaa suunnittelua, poistaa turhaa manuaalista ja toistuvaa työtä, sekä tehdä toistuva työ tehokkaammin ja laadukkaammin. Kilpailu on kovaa suunnittelutalojen välillä, ja kilpailukykyä pitää tehostaa. Tämän opinnäytetyön tarkoituksena oli toteuttaa suunnittelun työkalu, joka tuo instrumenttien sijoituskuvista tiedot Exceliin ja tarkastelee tietoja graafisnumeerisesti. Ohjelma toteutettiin Visual Basic for Applications -ohjelmointikielellä. Työ tehtiin Insta Automation Oy:lle. Ohjelman toiminnan ymmärtämiseksi opinnäytetyössä kuvataan perustoiminnot ja funktiot VBA-ohjelmoinnista. Opinnäytetyössä pohdittiin muiden ohjelmointikielien mahdollisuudesta työkalun toteuttamiseksi.

Opinnäytetyössä suunnitellun työkalun vaatimukset ovat seuraavat:

- hakea ja siirtää mahdollisimman helposti instrumenttien sijoituskuvasta tarvittavat tiedot Exceliin
- lisätä suunnitteluun olennaisia raja-arvoja, suodattaa tulosta graafisesti ja lisätä kuvaan Z-koordinaatti.
- laskea laitekaapeleiden pituudet.

Työkalun tavoitteena nopeuttaa instrumenttien sijoituskuvien käsittelyä ja suunnitteluprosessia poistamalla manuaalisesti tehtäviä asioita. Työkalu on tehty suunnittelun tarpeisiin, eikä työkalua voi käyttää hyväksi kunnossapidossa.

Sijoituskuvat kerran käsittelemällä lasketaan kaapelin mitta, muutetaan sijoituskuvaan sijoituspositioiden nimiä nopeammin ja helpommin. Ilman työkalun käyttämistä laitekaapeleiden mitta lasketaan manuaalisesti käsin mittaamalla AutoCAD-kuvasta, kotelojaon tarkastelu visuaalisesti on hankalaa, instrumenttien korkeutta on hankalaa tarkastella kuvissa ja sijoituskuvissa muutetaan yksi instrumentti kerrallaan. Työkalulla optimoidaan kaapeliverkon suunnittelua. Työkalu myös parantaa laatua vähentämällä käyttäjän virheitä käsittelemällä tietoa graafisnumeerisesti, jolloin kuvien tarkastaminen ja tarkastelu muuttuu helpommaksi. Sijoituskuvia voidaan tarkastaa Excelillä erilaisin funktioin, esimerkiksi PHAKU-funktiolla, jolloin kuvasta tiedot keräämällä sarakkeisiin voidaan vertailla kuvassa olevaa tietoa ja ulkopuolista lähdettä sekä keskenään. Vastaavaa työkalua ei ole toteutettu Insta Automation Oy:lla, joten työkalu toteutettiin perusteista alkaen. Työn kaikki osiot ja ominaisuudet ohjelmoitiin tilaajan vaatimusten mukaisesti.

2 Insta Group

Insta Group on perheyritys, joka toimii automaatio- ja ilmailu-alalla sekä kyberturvallisuudessa. Yritys perustettiin vuonna 1960, yrityksen nimellä Automatiikka-Asentajat Mattsson & K:nit. Yritys laajeni nopeasti kasvaen vuosikymmenen loppuun mennessä 220 henkilön työpaikaksi. Vuonna 1970 yrityksen kasvu jatkui ja laajeni suunnittelupalveluihin. Samana vuonna käynnistettiin myös lentoteknisten mittareiden korjauspalvelu. Vuonna 1980 käynnistyi puolustusvoimien strateginen tekninen kehitystyö. Vuoden 1990 jälkeen, yritys tarjosi myös informaatioturvallisuuden ratkaisuja asiakkaille. (Insta Group Oy. 2018.)

Nykyisin Insta Groupin tärkeimpiä toimialoja ovat prosessiteollisuus, energiantuotanto, materiaalien käsittely, vesilaitokset, valtionhallinto sekä turvallisuusorientoituneet organisaatiot. Insta Group Oy on Puolustusvoimien strateginen kumppani. (Insta Group Oy 2018.)

3 Instrumentointisuunnittelussa tuotetut dokumentit

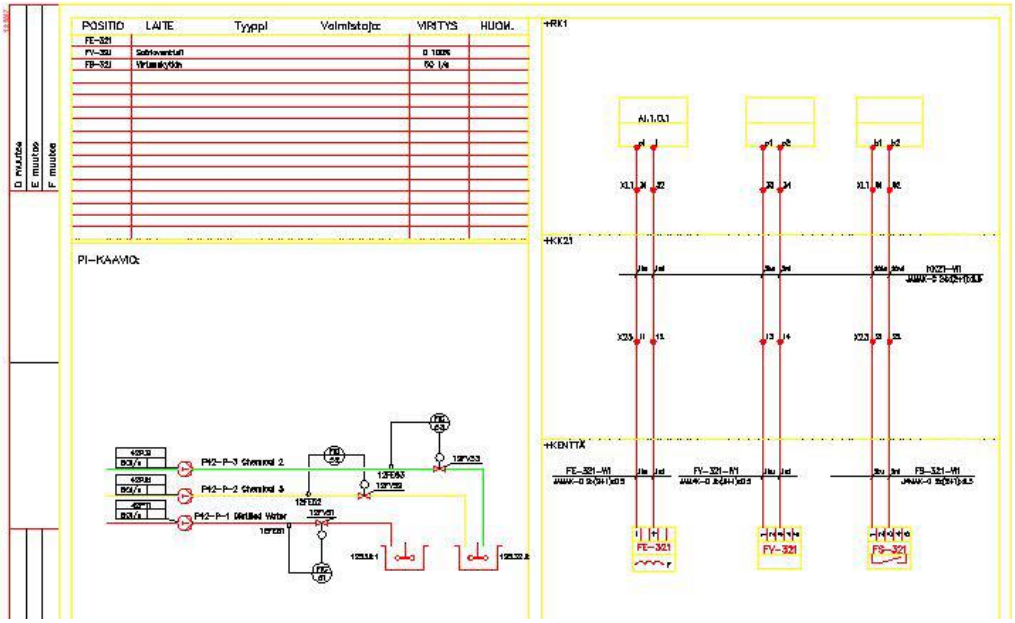
Instrumentointia aloittaessa esisuunnitteluvaiheessa kerätään asiakkaalta tarvittava tieto talteen, ja tietoja vertaillaan keskenään, jotta ristiriitaisuudet alkutiedosta saadaan selvitettyä. Kun tarvittavat alkutiedot on kerätty talteen, tietoa kerätään ja suunnitteluprosessi aloitetaan.

Instrumentoinnin suunnitteluprosessin lopputuloksena on erilaisia dokumentteja asennusta, kunnossapitoa ja jatkosuunnittelua varten. Instrumentoinnin suunnittelun toteuttamia dokumentteja jaotellaan eri dokumenttityyppeihin. Seuraavassa on jaoteltu muutamia erilaisia instrumentoinnin dokumentteja jaoteltu ja kerrottu, mihin niitä käytetään.

3.1 Piirikaaviot

Piirikaavio on kytkentäpiirustus, jossa on esitetty instrumenttien kytkentäreitti instrumentilta automaation prosessiasemaan asti. Kytkentäpiirustuksessa esitetään laitteen reitti kytkentäliittimineen. Piirikaaviota käytetään kunnossapidossa, sekä suunnitteluprosessissa. Se on tärkein dokumentti kunnossapidossa, koska siitä kunnossapito näkee koko laitteen reitin. Piirikaavioita toteutetaan nykyään tietokantapohjaisesti ja manuaalisesti

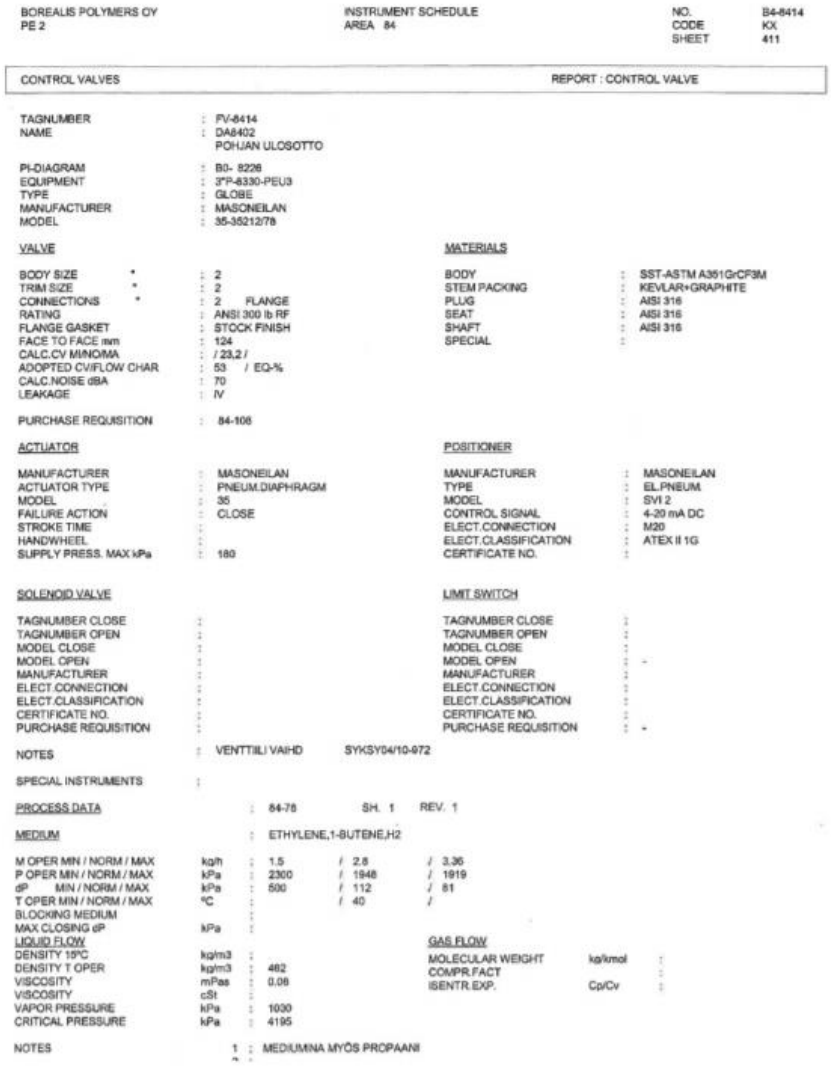
AutoCADilla piirtäminen on väistynyt tehokkuuden ja virheiden minimoimiseksi. Piirikaavioita toteutetaan tietokantapohjaisesti ALMA Consulting Oy:n toteuttamalla suunnittelujärjestelmällä tai CADS Electric -ohjelmalla. Kuvassa 1 piirikaavio on luotu CADS Electric –ohjelmalla tietokannan pohjalta.



Kuva 1. Esimerkki piirikaaviosta (Kymdata Oy.)

3.2 Kojeluettelo

Eri yrityksillä on tarpeita erilaisiin asioihin kojeluettelossa. Esimerkiksi kojeluetteloissa mainitaan mainittu instrumentin tagi, prosessidataa, signaalin tyyppi, prosessiin liitynnän tyyppi, EX-luokitus sekä valmistaja. Kojeluettelosta nähdään laitteen erilaiset suunnitteluun vaikuttavat arvot nopeasti ilman monien eri dokumenttien tarkastelua. Kojeluetteloa käytetään kunnossapidossa, kun tarvitaan tietoa laitteista ja prosessista nopeasti kuten kuvassa 2 nähdään.



BOREALIS POLYMERS OY
PE 2

INSTRUMENT SCHEDULE
AREA 84

NO. B4-8414
CODE KX
SHEET 411

CONTROL VALVES — REPORT : CONTROL VALVE

TAGNUMBER : FV-8414
NAME : DA8402 POHJAN ULOSOTTO

PI-DIAGRAM : B0- 8226
EQUIPMENT : 3"P-8330-PEU3
TYPE : GLOBE
MANUFACTURER : MASONEILAN
MODEL : 35-35212/78

VALVE

BODY SIZE * : 2
TRIM SIZE * : 2
CONNECTIONS * : 2 FLANGE
RATING : ANSI 300 lb RF
FLANGE GASKET : STOCK FINISH
FACE TO FACE mm : 124
CALC.CV MIN/NOMA : / 23,2 /
ADOPTED CV/FLOW CHAR : 53 / EQ-%
CALC.NOISE dBA : 70
LEAKAGE : IV

PURCHASE REQUISITION : 84-106

MATERIALS

BODY : SST-ASTM A351GrCF3M
STEM PACKING : KEVLAR+GRAPHITE
PLUG : AISI 316
SEAT : AISI 316
SHAFT : AISI 316
SPECIAL :

ACTUATOR

MANUFACTURER : MASONEILAN
ACTUATOR TYPE : PNEUM.DIAPHRAGM
MODEL : 35
FAILURE ACTION : CLOSE
STROKE TIME :
HANDWHEEL :
SUPPLY PRESS. MAX kPa : 180

POSITIONER

MANUFACTURER : MASONEILAN
TYPE : EL.PNEUM
MODEL : SVI 2
CONTROL SIGNAL : 4-20 mA DC
ELECT.CONNECTION : M20
ELECT.CLASSIFICATION : ATEX II 1G
CERTIFICATE NO. :

SOLENOID VALVE

TAGNUMBER CLOSE :
TAGNUMBER OPEN :
MODEL CLOSE :
MODEL OPEN :
MANUFACTURER :
ELECT.CONNECTION :
ELECT.CLASSIFICATION :
CERTIFICATE NO. :
PURCHASE REQUISITION :

LIMIT SWITCH

TAGNUMBER CLOSE :
TAGNUMBER OPEN :
MODEL CLOSE :
MODEL OPEN : -
MANUFACTURER :
ELECT.CONNECTION :
ELECT.CLASSIFICATION :
CERTIFICATE NO. :
PURCHASE REQUISITION : -

NOTES : VENTTIILI VAIHD SYKSY04/10-972

SPECIAL INSTRUMENTS :

PROCESS DATA : 84-78 SH. 1 REV. 1

MEDIUM : ETHYLENE,1-BUTENE,H2

M OPER MIN / NORM / MAX	kg/h	: 1.5	/ 2.8	/ 3.36	
P OPER MIN / NORM / MAX	kPa	: 2300	/ 1948	/ 1919	
dP MIN / NORM / MAX	kPa	: 500	/ 112	/ 81	
T OPER MIN / NORM / MAX	°C	:	/ 40	/	
BLOCKING MEDIUM		:			
MAX CLOSING dP	kPa	:			

LIQUID FLOW

DENSITY 15°C : kg/m3
DENSITY T OPER : kg/m3 : 462
VISCOSITY : mPas : 0.08
VISCOSITY : cSt :
VAPOR PRESSURE : kPa : 1030
CRITICAL PRESSURE : kPa : 4195

GAS FLOW

MOLECULAR WEIGHT kg/kmol :
COMPR.FACT :
ISENTR.EXP. Cp/Cv :

NOTES 1 : MEDIUMINA MYÖS PROPAANI

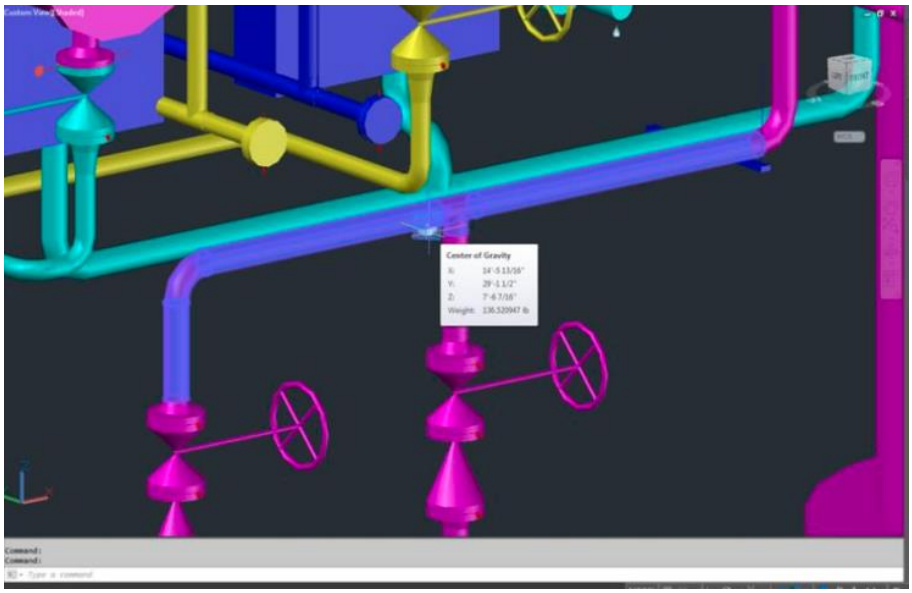
Kuva 2. Kojeluettelon esimerkki (Oraviita 2017: Liite 2)

3.3 Kenttäkaapeliluettelo

Kenttäkaapeliluettelo on lista kaikista kentällä olevista kaapeleista. Kaapelikilpien malli riippuu laajasti, eikä standardia kenttäkaapeleiden merkitsemisestä ei ole. Kaapelit merkitään yksilöivällä tunnuksella molempiin päihin, jotta kaapelit pysyvät tunnistettavissa kentällä.

3.4 Instrumenttien sijoituspiirustus

Instrumenttien sijoituspiirustukseen on piirretty instrumenttien, kytkentäkoteloiden sijoitus ja tärkeät lähipiirin tunnistukseen liittyvät asiat, esimerkki säiliöt, pumput ja kolonnit. Sijoituspiirustuksia käyttää esimerkiksi suunnittelija, kunnossapidon henkilöstö laitesijoitusten tarkasteluun kentällä. Sijoituspiirustus voi olla kuvasta riippuen joko 3D-mallina tai 2D-tasopiirustus. 3D-mallit ovat alkaneet korvata tavallisia 2D-tasopiirustuksia tarkkuuden vuoksi, ympäröiviä asioita ja mahdollisia ongelmakohtia 3D-kuvasta erotetaan paremmin, esimerkiksi kuvassa 3 nähdään venttiilien ympäryksen paikka tarkemmin, kuin 2D-tasokuvassa.

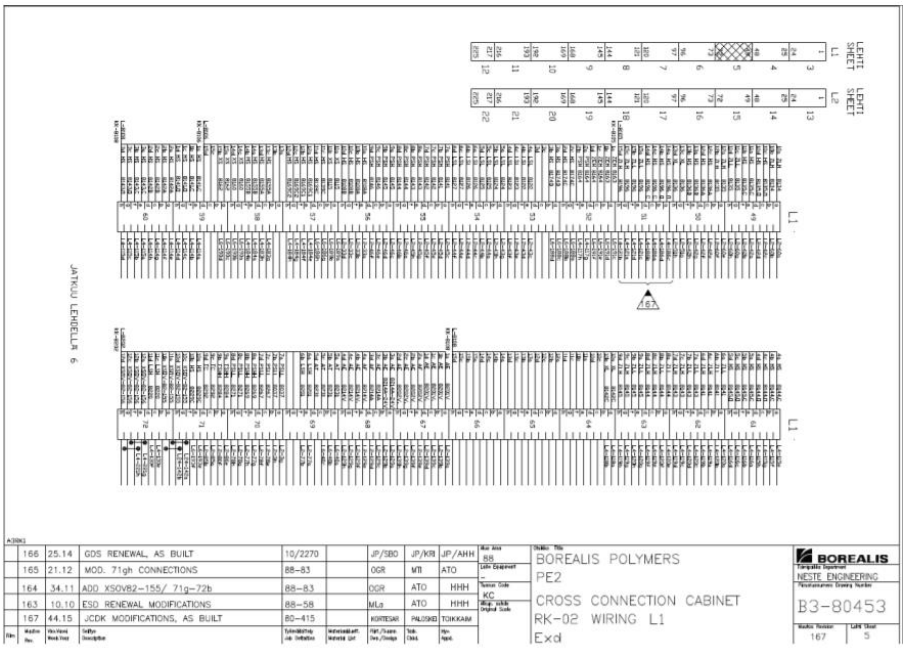


Kuva 3. Sijoituspiirustuksen 3D-malli (Autodesk Inc.)

3.5 Ristikytkentäkaappien kaaviot

Ristikytkentäkaappien kaavioista selviää, mistä kytkentäkoteloilta tulee tiettyyn ristikytkentäkaappiin signaaleja ja mitkä ovat riviliittimet, johon kyseinen signaali menee. Ristikytkentäkaappien dokumentit luodaan usein, kuten piirikaaviotkin, tietokantapohjaisesti.

Ristikytkennän dokumentteja käytetään esimerkiksi kunnossapidossa ja asennuksessa.

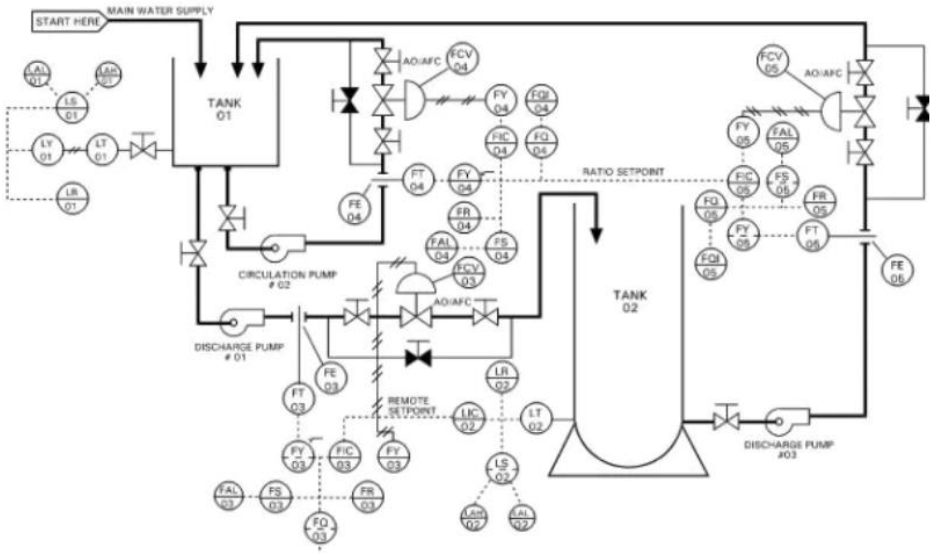


Kuva 4. Esimerkkikuva ristikytkentäkaapista (Oraviita 2017: 20)

Ristikytkentäkaapit ovat jaettu Exd- ja Exi-räjähdysluokitusten mukaisesti. EXI-räjähdysluokitus tarkoittaa luonnostaan vaaratonta laitetta, eli energiaa ei ole tarpeeksi edes viallisena tuottaa kipinää. Exd-luokitus tarkoittaa räjähdyksen kestävää laitetta. (Oraviita 2017: 20). Yleensä ristikytkentäkaappien dokumenteissa on mainittu EX-luokitus, kuten kuvassa 4 nähdään.

3.6 PI-kaaviot

PI-kaaviot eli putkisto- ja instrumenttikaavio on instrumentointisuunnittelun lähtötieto, jonka pohjalta suunnittelua tehdään. Kaavioissa kuvataan putkisto ja instrumentit säiliöineen ja muine prosessilaitteineen. PI-kaavioista on ilmoitettu prosessiteollisuuden paineluokat ja muut suunnitteluun vaikuttavia raja-arvoja, esimerkiksi rajahälytyksiä ja säätöjä, jota kuvassa 5 nähdään katkoviivoina.



Kuva 5. Esimerkki PI-kaaviosta

4 AutoCAD ja ohjelmointikielet

AutoCAD on Autodesk Inc:n perustaman vektorigrafiikkaohjelma. Yhtiön perusti John Walker ja hänen 15 yhtiökumppaniaan. Heidän tarkoituksenaan oli myydä viisi eri tuotetta, joista uskottiin vain yhden menestyvän. Näistä tuotteista onnistunein tuote oli AutoCAD, vektorigrafikkaohjelma, joka julkaistiin vuonna 1982. Yhtiö onnistui markkinoinnissa, sillä vain neljä vuotta myöhemmin AutoCAD oli maailmanlaajuisesti käytetty suunnitteluohjelmisto. (Weisberg 2014.) AutoCAD-kuvat toimitetaan tavallisesti DWG-tiedostona, joka on AutoCADin luoma ja omistama tiedostostandardi. (Kennedy 2014).

Nykyään AutoCAD myydään kahtena eri versiona, AutoCAD ja AutoCAD LT. LT-versio on karsittu versio, josta ei löydy muun muassa ohjelmointimahdollisuuksia eikä 3D-ominaisuuksia (Autodesk Inc. 2018a). AutoCAD:n lisäksi Autodesk on julkaissut useita eri ohjelmistoja, mekaniikkasuunnittelusta 3D-animointiin.

Varhaisista ohjelmaversioista alkaen lisäominaisuuksia on voitu aina tehdä, koska AutoCAD on tukenut laajasti eri ohjelmointikieliä. Ensimmäisenä ohjelmointikielenä on ollut AutoLisp. Kieli otettiin käyttöön ensimmäisistä versioista alkaen. Visual Basic for Application on ollut integroituna versiosta R14.01 alkaen. Versiosta 2010 lähtien VBA-moduuli

on mahdollista ladata ja asentaa ohjelmaan. Ilman lisämoduulia VBA:ta ei voida käyttää AutoCADissa. (Ambrosius 2015: 839).

VBA:n ja AutoLispin lisäksi AutoCAD tukee myös muita ohjelmointikieliä ObjectARX-ohjelmointiympäristön avulla. ObjectARX:n avulla voidaan toteuttaa lisäosia, komentoja ja ohjelmia VB.Net, C# sekä C++-ohjelmointikielillä käytettäväksi.

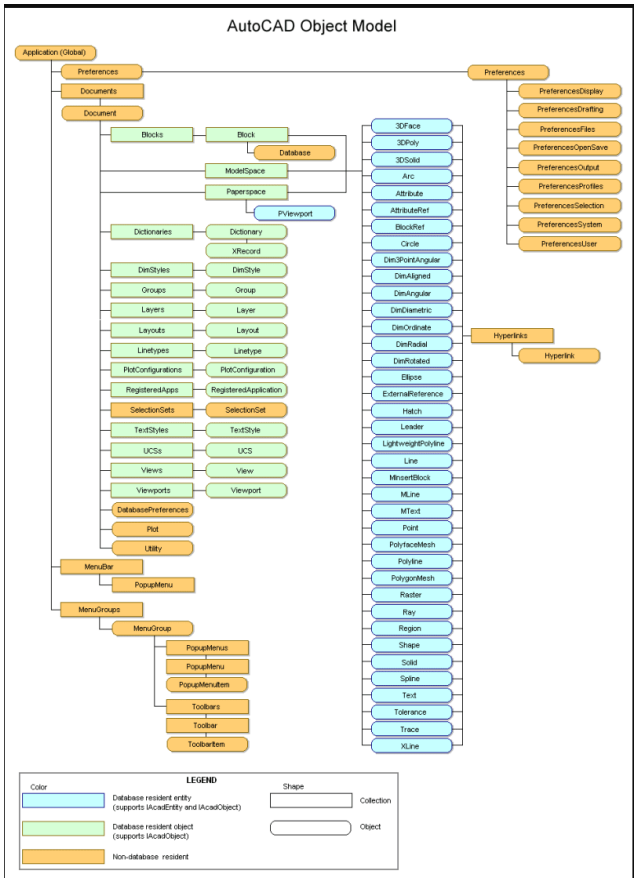
5 Visual Basic for Application

5.1 Visual Basic for Applicationsin perusteita

Visual Basic for Applications, lyhennettynä VBA, on Microsoftin vuonna 1998 kehittämä ohjelmointikieli, joka on alkuperäisesti muunnos Visual Basic 6 (VB6) -ohjelmointikielestä. VBA mahdollistaa makrojen ja ohjelmien yksinkertaisen teon Microsoftin ohjelmiin. VBA:lla ei pystytä toteuttamaan yksin ajettavia ohjelmia, vaan jokin sovellus on aina oltava ohjelman isäntäsovellus. Suoritettava koodi tallentuu .dvb-tiedostoksi. VB6:lla tehdyt ohjelmat ovat erikseen ajettavia exe-ohjelmia. (Ambrosius 2015: 839.)

Visual Basic for Application on nimensä mukaisesti visuaalista ohjelmointikieltä, ja se on helposti opeteltavissa. VBA:n kanssa täysin kokematonkin käyttäjä pystyy nopeasti oppimaan uuden ohjelmointikielen. Ohjelmointikielessä on vahva integraatio Windowsiin sekä Office-ohjelmistoihin. Se on yksinkertaisempi muihin ohjelmointikieliin verrattuna. (Caputo 2017a.)

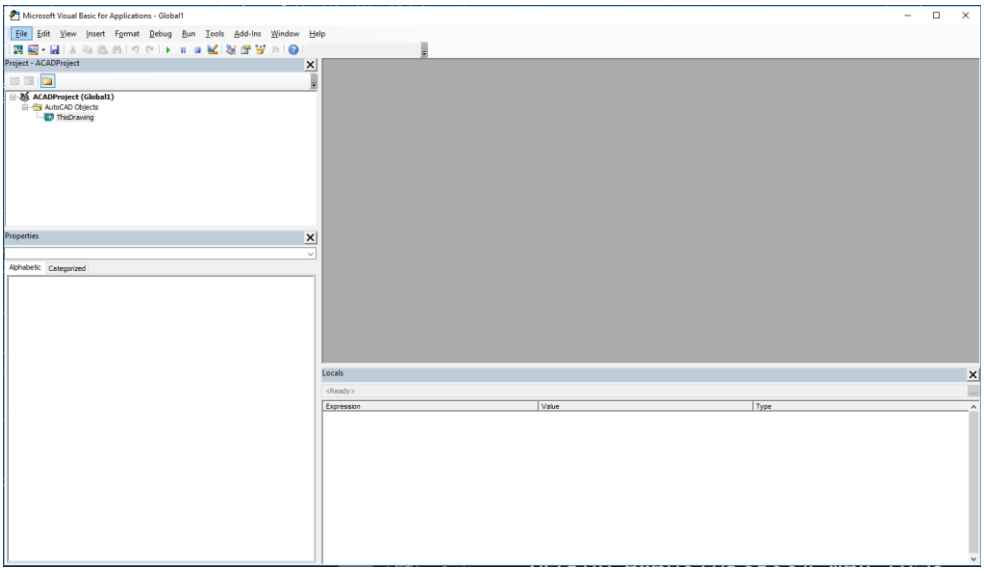
VBA ohjelmointikielenä on objektipohjainen ohjelmointikieli. Esimerkiksi yhden ympyrän piirtämiseksi kuvaan ohjelmaan on kerrottava, missä dokumentissa ja missä piirustusmaailmassa objekti sijaitsee. Näiden toimenpiteiden jälkeen voidaan objekteja piirtää ja muokata piirustuksessa ja määritellä ympyrän koordinaatit sekä halkaisija (Autodesk Inc. 2017a).



Kuva 6. AutoCAD:n VBA-objektikuvaus

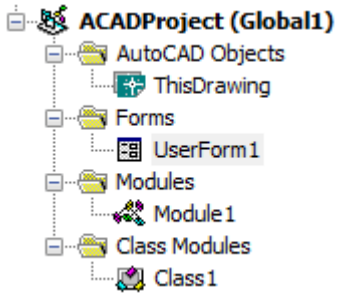
AutoCADissa objektit rajautuvat objekteihin, jotka ovat ja eivät ole tietokannassa. Objektit ovat jakautuneet funktioihin, jotka toimivat AutoCadin kahdella sisäisillä funktioilla, ja funktioilla, joita voidaan käyttää ainoastaan yhdellä funktiolla kuten kuvassa 6 nähdään.

VBA:ta koodataan Visual Basic Editorissa. Editorissa toiminto jakaantuu muutamaan eri osioihin. Editori aukaistaan Manage-palkista tai syöttämällä komentoriville komento ***vbaide***.



Kuva 7. Koodausympäristö

Kuvassa 7 on keskellä koodieditori koodaamista varten, vasemmalla on projekti-ikkuna, jossa on kaikki käytetyt moduulit ja koodin osat. Vasemmalla alareunassa on ominaisuudet-ikkuna, jossa voidaan tarkastella esimerkiksi moduuleiden aktiivisia muuttujia ja muita ominaisuuksia.



Kuva 8. VBA-moduulit

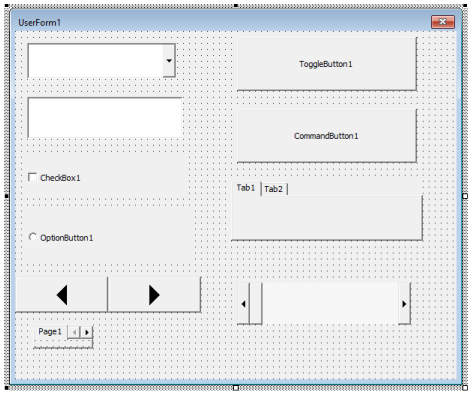
VBA:ssa on kolme päämoduulia: Module, UserForm, ja ThisDrawing. Eri moduuleissa voidaan esimerkiksi tehdä koodia moduuleihin tai lisätä käyttäjän interaktiivisuutta tai erilaisia yksilöiviä ajettavia moduuleita. Koodi voidaan jakaa eri osiin, jolloin ohjelma on hierarkkinen ja helpommin jäsenneltävä. Ohjelmaa voidaan rakentaa eri moduuleihin, ja Call-toiminnolla (ohjelman nimi) voidaan moduulista kutsua toista osuutta tai moduulia. Moduulit voidaan tehdä ohjelmoijan toiveiden mukaisesti parantamaan koodin järjestystä ja luettavuutta.

VBA jakaantuu moduuleihin, jossa voidaan suorittaa koodia kuvan 8 mukaisesti. Eri moduuleissa koodia voidaan suorittaa ja tehdä yksilöiviä asioita, joita ei voi muissa moduuleissa tehdä.

ThisDrawing-objekti on toiminto, jossa voidaan suorittaa koodia vain auki olevalle piirustukselle. Komentoa käytetään, kun auki olevaan piirustukseen tehdään merkintöjä tai koodia. Haittana on, että ThisDrawing-toiminto on tyypiltään objekti eikä moduuli. Muuttujien jakaminen eri moduuleiden ja objektien välillä on sen vuoksi haasteellista. (Ambrosius 2015: 818.)

Class Modulessa voidaan toteuttaa omia objekteja VBA:n käytettäväksi. Tätä kautta voidaan lyhentää tarvittavaa koodia ja tarvittaessa rakentaa ohjelmablokki kerrallaan. Class Moduuleiden kautta rakentaessa ohjelmaa on helpompaa rakentaa ja muokata verrattuna tavallisien moduuleihin. Koodin päivitys ei tee ongelmia muuhun ohjelmakoodiin. (Kelly 2016.)

UserForm-objekteja käytetään antamaan käyttäjälle viestejä ja syötettä ohjelmakoodin käytettäväksi. Ohjelman interaktiivisuus käyttäjän kanssa paranee. UserForm-objektissa voidaan määrittää erilaisia ehtoja.



Kuva 9. Erilaisia UserForm-ratkaisuja

Kuvassa 9 olevalle UserForm-objektille voidaan määritellä funktioita ja toimintoja esimerkiksi käyttäjän vaihtaessa arvoa tai klikatessa painiketta (Caputo & Cai. 2018c).

5.2 Ohjelmien perusfunktio ja rakenne

VBA:ssa on kahdenlaisia ohjelmia, funktioita sekä proseduureja. Proseduureja voidaan käyttää myös ali-ohjelmina. Erona näillä on se, että proseduuriohjelma ei palauta arvoa, kun funktio palauttaa arvon käyttäjälle. Ohjelmakoodi määritellään funktioksi tai proseduurityyppiseksi Sub- tai Function-termillä ja nimeämällä se halutulla tavalla. (Kelly 2015.)

```
Sub (ohjelma)
’tee jotain
End sub
m
```

Esimerkkikoodi 1. VBA-perusrakenteesta

Esimerkkikoodi 1:n mukaisesti Proseduuri-tyyppisessä rakenteessa ohjelma alkaa Sub-sanalla ja nimellä ja päättyy End Sub -ilmaisuun.

```
Function pintaala(pituus As Double, leveys As Double)
pintaala = pituus * leveys
End Function
```

Esimerkkikoodi 2. Funktio-esimerkki

Esimerkkikoodi 2 laskee pinta-alan kahdella muuttujalla ja palauttaa funktioon. Funktioiden perusrakenteessa pitää määritellä palautettava arvo ja muuttujat.

VBA:ssa ohjelman muuttujia pitää määritellä nimeämällä muuttujan nimi ja laittamalla Dim-sana sen eteen. Dim varaa muuttujan funktiolle. (Kelly 2018).

```
Dim muuttujannimi
Dim muuttujannimi as datatyyppi
```

Esimerkkikoodi 3. Muuttujan määrittely

Jos määritellään muuttuja esimerkkikoodi 3 mukaisesti ilman datatyyppiä, tällöin datatyypiksi tulee Variable. Hyvin määritellyt muuttujat helpottavat ohjelman suorittamista. Option Explicit koodin alussa varmistaa, että jokainen muuttuja on varmasti määritelty. Ohjelma pakottaa jokaiselle koodissa olevalle muuttujalle datatyypin ja ohjelma etenee ainoastaan kun muuttujat on määritelty (Caputo & Cai. 2018); (Dollard ym. 2015).

Jos muuttujia nimetään datatyypin mukaan, datatyyppi mainitaan muuttujan jälkeen. Datatyypeistä on kerrottu enemmän luvussa 5.3. Muuttuja voidaan määritellä julkiseksi, yksityiseksi tai staattiseksi Public-, Static-, Private-, tai Dim-määritelmillä.

Private-julistuksella tieto pysyy ainoastaan moduulin sisässä ilman, että käyttäjä voi sitä hyödyntää. Static-muuttujalla voidaan määritellä paikallaan pysyviä muuttujia. Esimerkiksi Static-muuttujilla voidaan ohjelmakoodi sammuttaa ja käynnistää uudestaan ja ohjelma muistaa muuttujaan syötetyt arvot. Public-julistuksella arvot ovat julkisia ja voidaan käyttää jokaisessa moduulissa. Public- ja Static-muuttujia voidaan käyttää vain moduuleissa, ThisDrawing- tai UserForm-objekteissa muuttujien käyttö ei onnistu. (Caputo & Cai 2017).

Arvojen määrittely muuttujille

Muuttujan esittelyn jälkeen muuttujille voidaan määrittää tai hakea arvo toisesta muuttujasta. Haku tehdään yhtä kuin -merkillä ja laittamalla arvo oikealle puolelle. Vasemmanpuoleinen muuttuja asettuu oikealla puolella mainituksi arvoksi. Esimerkkikoodi neljässä luku määritellään aluksi viideksi, ja toisena määrittelynä on Excelin solu A1. (Caputo & Cai 2017b).

```
dim luku as string
luku=5
luku=excel.worksheet.cells(1,1)
```

Esimerkkikoodi 4. Muuttujien arvot

Koodissa turhaa toistamista voidaan vähentää ja samalla nopeuttaa itse ohjelman suorittamista määrittelemällä funktiot lyhemmäksi esimerkkikoodi 5 mukaisesti.

```
Dim piirraviiva as thisdrawing.modelspace.addline
```

Esimerkkikoodi 5. Funktioiden lyhentäminen

Myöhemmin ohjelmassa viiva piirretään kuvaan piirraviiva() -komennolla ja merkitsemällä siihen alku- ja loppupiste.

5.3 Datatyypit

Muuttujiin ja objekteihin voidaan tallentaa erilaisia tietoja, jotka ohjelma käyttää uudestaan. VBA:ssa data on tallennettu muuttujiksi ja loput objekteihin. Molempia pystytään muuttamaan käyttäjän toiveiden mukaisesti.

Data Type	Bytes Used	Range of Values
Byte	1 byte	0 to 255
Boolean	2 bytes	True or False
Integer	2 bytes	−32,768 to 32,767
Long	4 bytes	−2,147,483,648 to 2,147,483,647
Single	4 bytes	−3.402823E38 to −1.401298E−45 (for negative values); 1.401298E-45 to 3.402823E38 (for positive values)
Double	8 bytes	−1.79769313486232E308 to −4.94065645841247E−324 (negative values); 4.940656458412 47E−324 to 1.79769313486232E308 (for positive values)
Currency	8 bytes	−922,337,203,685,477.5808 to 922,337,203,685,477.5807
Decimal	12 bytes	+/−79,228,162,514,264,337,593,543,950,335 with no decimal point; +/−7.9228162514264337593543950335 with 28 places to the right of the decimal
Date	8 bytes	January 1, 0100 to December 31, 9999
Object	4 bytes	Any object reference
String (variable length)	10 bytes + string length	0 to approximately 2 billion characters
String (fixed length)	Length of string	1 to approximately 65,400 characters
Variant (with numbers)	16 bytes	Any numeric value up to the range of a double data type. It can also hold special values, such as Empty, Error, Nothing, and Null.
Variant (with characters)	22 bytes + string length	0 to approximately 2 billion
User-defined	Varies	Varies by element

Kuva 10. VBA:n muuttujat (Mueller)

Datatyypit määrittävät tavan, jolla VBA käsittelee dataa. Mahdolliset datatyypit VBA:ssa mainitaan kuvassa 10. Tavallisesti datatyypiksi valitaan sellainen, joka mahtuu alueisiin. Alueissa pystytään käsittelemään datatyyppejä siten, että mahdollisimman vähän bittejä on käytössä. Mitä vähemmän bittejä muuttujissa on, sitä nopeampi on VBA-koodi. Muuttujien lisäksi käyttäjä voi määrittää funktioille muuttujia kontekstista riippuen. (Caputo & Cai 2018a).

```
Dim luku as long
Dim teksti as string
```

Esimerkkikoodi 6. Kahden eri muuttujan esittely

Esimerkkikoodi 6 kontekstissa teksti voi sisältää myös lukuja, mutta long-tyyppiseksi määritelty ”luku” ei voi sisältää muuta kuin lukuja.

5.4 VBA:n perusfunktioita ja ohjausrakenteet

Perusrakenteet on hallittava VBAssa. Erilaiset silmukat ovat erittäin tehokkaita ohjelmassa olioiden tai numeroiden prosessoimiseksi. VBA:ssa löytyy silmukoita eri tarkoituksiin, yleisimmät tavat ovat For- ja Do Until/While -silmukat. Tarvittaessa voidaan käyttää GoTo-rakennetta silmukan tekemiseen.

5.4.1 For-silmukka

For-silmukka on yleisin tapa lukea lukuja tai objekteja. Se lukee asioita käyttäjän määrittämästä alarajasta ylärajaan.

```
For laskuri = start To end (Step) (statements) (Exit For) (statements)
Next laskuri
```

Esimerkkikoodi 7. For-rakenne

For-rakenne muodostuu esimerkkikoodi 7:n mukaisesti For-sanasta, kahden luvun tai määritelmän välistä luettavia tiedoista, Next-sanasta ja laskurista. Laskuri, luvut ja Next-sana silmukassa on pakollisia määritelmiä. Loput määritelmät ovat vapaaehtoisia. Laskurissa olevat askeleet voidaan määrittää positiivisiksi tai negatiivisiksi. Askeleen kokoa voidaan muuttaa yhdestä eteenpäin. Jos askeleen koko jätetään määrittelemättä, VBA tulkitsee sen yhdeksi askeleeksi. Kun silmukasta halutaan poistua ennen kuin itse rajaarvo on ylitetty, voidaan käyttää Exit For –käskyä. (Caputo & Cai. 2017b)

Jos For-rakenteella luetaan objekteja, käytetään For Each -rakennetta. Ryhmä voi olla käyttäjän määrittämä tai tietty valmiiksi määritelty ryhmä, esimerkiksi kaikki objektit CAD-kuvassa. For Each-ohjausrakenteen perusrakenne on esitetty esimerkkikoodissa 8.

```
 For Each elementti in ryhmä
 'koodia
 Continue For
 'koodia
 Exit For
 'koodia
Next elementti
```

Esimerkkikoodi 8. For Each-perusrakenne

Exit For and Continue For -ilmaisuilla jatketaan ohjelmaa tai lopettaa silmukkarakenne, mikäli ohjelman määrittelyt tapahtuvat. VBA-koodi käy kerran läpi ryhmässä olevat oliot

For Each-rakenteen alussa. Jos ryhmään tulee lisää objekteja, For Each-rakenne ei tunnista muutoksia silmukassa (Wenzel ym. 2015d).

5.4.2 Do Until/While

Do Until/While -rakenteella luetaan olioita niin kauan läpi, kunnes saavutettu arvo muuttuu todeksi tai epätodeksi. Ehto voidaan määrittää epätodeksi While-rakenteella tai todeksi Until-rakenteella. Kuitenkaan While ja Until eivät voi olla samassa funktiossa samaan aikaan (Matt ym. 2015a).

```
Do while/until ehto
’koodia
Loop
```

Esimerkkikoodi 9. Do while-esimerkki

Do Until/While -silmukan perusrakenteessa pakollisia ovat esimerkkikoodi 9:ssä mainitut asiat, eli Do Until/While, ehto sekä Loop-sana lopussa. Do Until/While -rakenteessa virheellinen, päättymätön silmukka pysäytetään CTRL + Break -näppäinyhdistelmällä.

5.4.3 GoTo-rakenne

GoTo-rakennetta voidaan käyttää silmukan rakentamiseen. GoTo-rakenteella voidaan hypätä määrättyyn leimaan tai riviin (Caputo & Cai. 2017d).

```
GoTo kysymys:
'koodia
kysymys:
‘koodia lisää
```

Esimerkkikoodi 10. GoTo-perusrakenne.

Esimerkkikoodi 10:n rakenne koostuu GoTo-sanan jälkeen määriteltävästä leimasta sekä varsinaisesta leimasta myöhemmin koodissa.

5.4.4 If-lause

VBA:n tärkein ohjauslauseke on ehtolause. Ehtolauseilla määritetään ryhmä lausekkeita ja toteutetaan ohjelmoitu funktio. Ehtolauseen lausekkeita voi olla äärettömästi ilman yläarvoa. Ehtolauseen ratkaisuna on tosi tai epätosi. (Caputo & Cai. 2018b.)

```
If ehto1 Then
 'teejotain_1
ElseIf ehto_2 Then
 'teejotain_2
...
Else
 'tee jotain
End If
```

Esimerkkikoodi 11. If-rakenne

Esimerkkikoodi 11 rakentuu If-sanasta, ehdosta ja ehdon jälkeen Then-sanasta, suoritettavasta koodista ja End If -loppumääritelmästä rakenteen lopussa. Ehtoja pitää olla ainakin yksi, tarvittaessa toisia kohteita voidaan käydä läpi ElseIf- sekä Else-ilmaisuilla.

5.4.5 Select Case

Esimerkkikoodi 12:n -rakenteella voidaan lyhentää ehtolauseita. Tällöin voidaan korvata pitkät ehtolauseet siistimmin ja kompaktimmin. Select Case rakenteessa ohjelmassa tarkasteltava datatyyppi voi olla Boolean, Byte, Char, Date, Double, Decimal, Integer, Long, Object, SByte, Short, Single, String, UInteger, ULong, tai UShort. (Wenzel & Hoffman. 2015b.)

```
Select Case testikohde

Case esimerkki1
'koodi
Case esimerkki2
'koodi
Case else
'koodi
End Select

Case 5 To 10, 11 To 15, 16
```

Esimerkkikoodi 12. Select Case- esimerkki

Select Case -rakenne muodostuu Select Case -määritelmästä, haettavasta määritelmästä ja koodin lopussa End Select -loppumääritelmästä. Rakenteella voidaan laskea

tai käsitellä useita tietoja samaan aikaan. Kohteet erotellaan pilkulla ja kahden objektin välit To-sanalla.

5.4.6 With... End

With… End -rakenteella suoritetaan useita käskyjä objekteille ilman objektien tiedon uudelleenkutsumista.

```
With objekti
.objektin_määrityksiä
End
```

Esimerkkikoodi 13. Objektien määrittelyn lyhentäminen

With… End -rakenne koostuu määritetyistä objektista ja End-käskystä lopussa. Esimerkkikoodi 13:n rakenteella koodin lukeminen helpottuu ja toiminta nopeutuu, kun funktioita ei kutsuta joka kerta uudestaan. (Wenzel ym. 2015c.)

5.5 Virheiden käsittely

Yksikään ohjelmakoodi ei ole täydellinen. Virheitä syntyy ohjelmassa olevina ohjelmointivirheinä tai käyttäjän aiheuttamana virheinä. Virheiden sattuessa VBA keskeyttää ohjelmakoodin ja näyttää virhenumeron. Ohjelmaan saadaan rakennettua virheiden varalle virheiden käsittely.

GoTo rivi	Virheen tapahtuessa koodi menee riville, jossa käsitellään virheitä.
GoTo 0	Poistaa käytöstä virheiden hallinnen nykyisessä koodissa ja asettaa sen tyhjäksi.
GoTo -1	Poistaa käytöstä aktivoidun poikkeuksen ja nollaa asettaa sen tyhjäksi.
Resume Next	Jättää huomioimatta ohjelmakoodissa tapahtuvat virheet ja menee koodissa eteenpäin

Taulukko 1. Virheenhallinnan ratkaisut (Dollard ym 2015.)

Virheitä voidaan poistaa ja muokata taulukossa 1 näkyvillä ilmaisuilla. Virheiden käsittely toteutetaan On Error GoTo -rakenteella, jolloin virheen sattuessa päädytään virheiden

hallintaan. Virhetilassa voidaan poistaa ja muokata virheitä. Virheitä korjataan yksi kerrallaan. Ohjelma tarvittaessa lopetetaan Exit Sub -komennolla virheiden hallinnassa.

```
Select Case Err.Number ’valitse virhe
Case 9 'Et ole valinnut pistettä
MsgBox("et ole valinnut pistettä")

Case Else
msgbox(virhe on tapahtunut, ohjelma lopetetaan)
Exit sub
End Select
```

Esimerkkikoodi 14. Virheiden käsittely yksitellen.

5.6 Koodiesimerkki

Kun perusfunktiot ovat tiedossa, toteutetaan koodiesimerkki, jossa käytetään mainittuja funktioita. VBA rakenne on hierarkkinen. Esimerkiksi ympyrän piirtäminen tapahtuu aluksi määrittelemällä sille, missä piirustuksessa, missä malliympäristössä ohjelma toteutetaan, minkä jälkeen voidaan asettaa viiva piirustukseen.

```
Sub piirraviiva()
Dim alku(1 To 2) As Double
Dim loppu(1 To 2) As Double
Dim ympyrä As AcadCircle
Public ss As AcadSelectionSet
Public ent As AcadEntity
dim piirraympyra as ThisDrawing.ModelSpace.AddCircle

origo(0) = 0: origo(1) = 0

halkaisija=5

Set circle = piirraympyra(origo,halkaisija)
ThisDrawing.SelectionSets.Add ("ss")
Set ss = ThisDrawing.SelectionSets("ss")
ss.SelectOnScreen

For Each ent In ss
Select Case ent.ObjectName

Case Is = "AcDbCircle"
Set line = ent
MsgBox ("tämä on ympyrä")

End Select
Next ent

ThisDrawing.SendCommand "_Qsave "
End Sub
```

Esimerkkikoodi 15. Laajempi koodausesimerkki

Esimerkkikoodi 15 piirtää ympyrän kuvaan, minkä jälkeen käyttäjä tekee valinnan, jolla valitaan ympyrä kuvasta. Kun objekti on valittu, koodi käy objektin tyypit läpi. Kun ohjelma havaitsee, että objekti on ympyrä, se informoi käyttäjää laatikolla, jossa lukee ”tämä on ympyrä”, ja lopuksi tallentaa piirustuksen.

6 AutoLisp

AutoLisp on AutoCADin oma ohjelmointikieli, joka on kehitetty 1950-luvulla ja on ollut käytössä jo 1980-luvusta lähtien. Alkuperäisesti LISP-ohjelmointikieli kehitettiin tekoälyn ohjelmointikieleksi 1950-luvulla. AutoLispin suosion syy on sen saumaton yhteensopivuus AutoCADin kanssa. Lisäosia ei ladata ohjelmien kirjottamiseksi. Ohjelmalla kirjoitetaan suoraan txt-tiedostoon, mutta ohjelman kirjoittamiseen voidaan käyttää Visual Lispiä (Nicholson 2016).

AutoLispillä voidaan esimerkiksi:

- tehdä komentoja ja toteuttaa niitä AutoCADin komentolinjalta
- luoda ja muokata graafisia objekteja piirustuksissa, esimerkiksi ympyröitä ja viivoja
- muokata ei-graafisia objekteja ja niiden tyylejä
- toteuttaa laskutoimintoja
- toteuttaa käyttäjäinteraktiivisuutta joko pop-up ikkunoilla tai komentolinjalla
- tallentaa ja muokata tiedostoja käyttäjän koneella
- lukea ja kirjoittaa ulkoisiin tiedostoihin
- yhdistää ohjelmia, jotka tukevat ActiveX:ä tai COM:a
  (Ambrosius 2015: 388.)

AutoLispistä on saatavilla tietoa useista eri lähteistä internetissä. Ohjelman pääkomentoihin ei ole tullut päivityksiä tai muutoksia, joten aiemmin koodatut ohjelmat ovat pysyneet toiminnallisina useamman vuoden ajan.

AutoLisp koostuu erilaisista funktioista. Funktioilla on vastaava rakenne jokaisella koodirivillä.

```
(defun c:piirräympyrä ()
 (command "._circle" "5,5" "3,3")
 (princ)
)
```

Esimerkkikoodi 16. Ympyrän piirtäminen

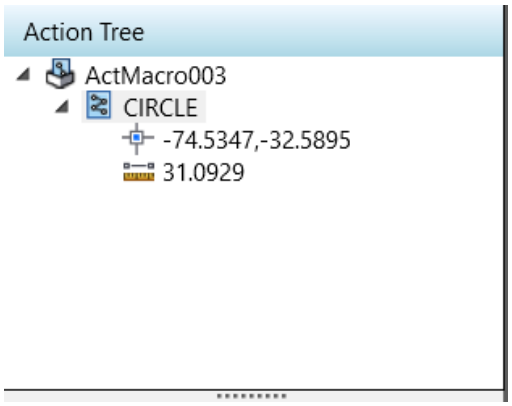
Koodi alkaa sulkeilla ja sisältää funktion nimen ja argumentteja. Funktio loppuu sulkuun. Esimerkkikoodissa defun määrittelee funktion nimen, jolla voidaan ohjelma käynnistää ja _.circle piirtää kohtaan 5,5 ympyrän, jonka halkaisija on 3. Käyttäjän nimeämiä muuttujia voidaan määritellä setq-funktiolla. Funktiot ovat voimassa niin kauan, kun CAD-kuva on auki.

Funktiot ja käyttäjän muuttamat parametrit määritellään täysin samalla tavalla. Esimerkiksi COMMAND- ja command-määritelmillä ei ole merkitystä toimivuuden kannalta, molemmat toimivat yhtä hyvin. (Ambrosius 2015: 390.)

AutoLisp on VBA:han verrattuna haasteellisempi ohjelmointikieli, mikäli kaksi tai useampi ohjelma kommunikoi keskenään samanaikaisesti. Esimerkiksi tiedon siirtäminen Excelin taulukoihin voidaan toteuttaa käyttäen ainostaan .csv tiedostoja. AutoLisp ei soveltunut ohjelmointikielen formaatin vuoksi opinnäytetyön ohjelmointikieleksi.

6.1 Makrot ja skriptit

AutoCADilla tallennetaan makroja. Excelissä makrot muuttuvat VBA-kieliseksi ja AutoCADissa makrot tallentuvat Action Macroiksi. Tiedostomuoto on .actm. Makroa muokataan ainoastaan AutoCADissa.



Kuva 11. Tallennettu makro, ympyrän piirtäminen pisteeseen

Action Macro nähdään tulosteena visuaalisena puuna, jossa on komennot Action Macrossa suoritettavat komennot. Action Macrojen lisäksi AutoCADissa toteutetaan tavallisia skriptejä. Skripti on ASCII-tekstitiedosto, jonka riveillä on komentoja, arvoja ja AutoLisp-ilmaisuja. Skripti jäljittelee komentoriviä AutoCADissa ja sitä on mahdollista luoda Notepadilla ja tallentaa .scr-muotoon (Autodesk Inc. 2015).

```
_.circle _non 0,0,0 5
```

Esimerkkikoodi 17. Esimerkkiskripti

Esimerkkikoodi 17 piirtää ympyrän kohtaan 0,0, joka on 5 halkaisijaltaan.

Skriptejä ajetaan vain yksi kerrallaan Script-komennolla tai Manage-ikkunasta. Opinnäytetyön kannalta skriptit ja Action Macrot ovat liian yksinkertainen ohjelmointitapa. Makrot ja skriptit ovat hyviä, jos tehdään yksinkertaisia asioita ja halutaan nopeasti tehdä ohjelma käytettäväksi.

7 ObjectARX®

ObjectARX on ohjelmointirajapinta, jolla voidaan tehdä ohjelmointikielillä C++, C# ja VB-Net AutoCADissa käytettäväksi. ObjectARX on ilmainen ja ominaisuuksien laajentamiseen tarkoitettu lisäosa. Rajapinnan kautta päästään käsiksi AutoCADin ytimeen, ja rajapinnassa hyödynnetään avointa arkkitehtuuria. Ohjelman avulla pystytään tekemään itse tehtyjä uusia komentoja. ObjectARX vaatii kuitenkin C++-, C#- tai VB.Net-ohjelmointikielen osaamista sekä erillisen kääntäjän, esimerkiksi Microsoft Visual Studion. (Autodesk Inc. 2018a).

```
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.Geometry


Public Sub AddCircle()
 '' Get the current document and database
 Dim acDoc As Document = Application.DocumentManager.MdiActiveDocument
 Dim acCurDb As Database = acDoc.Database
 Using acTrans As Transaction = acCurDb.TransactionManager.StartTransaction()
 Dim acBlkTbl As BlockTable
 acBlkTbl = acTrans.GetObject(acCurDb.BlockTableId, OpenMode.ForRead)

 Dim acBlkTblRec As BlockTableRecord
 acBlkTblRec = acTrans.GetObject(acBlkTbl(BlockTableRecord.ModelSpace), _
```

```
    OpenMode.ForWrite)

 Using acCirc As Circle = New Circle()
  acCirc.Center = New Point3d(2, 3, 0)
  acCirc.Radius = 4.25

  acBlkTblRec.AppendEntity(acCirc)
  acTrans.AddNewlyCreatedDBObject(acCirc, True)
 End Using

 '' Save the new object to the database
 acTrans.Commit()
 End Using
End Sub
```

Esimerkkikoodi 18. Ympyrän piirtäminen VB.Net ohjelmointikielellä (Autodesk Inc. 2017b)

VB.Netillä yhden ympyrän piirtäminen tehdään esimerkkikoodin 18 esittämällä tavalla, eli avataan kuva, avataan ja luodaan taulukon merkintä, luodaan Entity, minkä jälkeen lisätään kuvaan objekti ja tämän jälkeen tallennetaan piirretty muoto tietokantaan. VB.Net ei soveltunut opinnäytetyöhön kielen haasteellisuuden vuoksi.

8 Opinnäytetyö

8.1 Lähtötilanne ja opinnäytetyön suunnittelu

Työkalun tekeminen käynnistettiin tarvekartoituksella. Tarvekartoituksessa huomattiin, että tarvitaan ohjelma, jolla helpotetaan suunnitteluprosessia sijoituskuvien kanssa. Insta Automation Oy:ssa ensisijaisena CAD-ohjelmana on käytössä CADS, AutoCADissa on ainoastaan muutama lisenssikopio. Aikaisempia CADS:n pohjautuvia työkaluja ja ohjelmointiosaamista yrityksessä on. CADS ei kuitenkaan tue VBA:ta suoraan, ja tarvittavat komennot pitää toteuttaa makroilla, näppäimistön komennoilla tai K-kielellä. Opinnäytetyön laajuuteen työkalun tuottaminen vaatii ohjelmointitaitoa K-kielellä, joka on CADS:n oma ohjelmointikieli. CADS:n ja Excelin välinen integrointi on hankala toteuttaa, koska yhteistä ohjelmointikieltä ei ole. Opinnäytetyön ohjelma (jatkossa Työkalu) toteutettiin siksi AutoCADilla ja VBA:lla. Työn rajaus sovittiin ohjaavan henkilön kanssa.

Työn tulee sisältää:

- laitekaapeleiden pituuden laskentaa
- tiedonsiirtoa AutoCAD-kuvasta Exceliin
- suodatus AutoCAD-kuvaan eri raja-arvoin
- Z–koordinaatin lisäyksen objekteille.

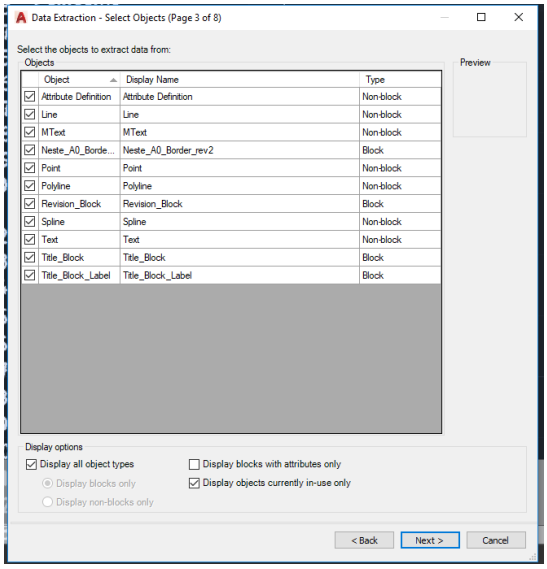
Ohjelmakoodia tuotettiin AutoCAD 2018:n ja Excel 2016:n VBA-kirjastoilla. Ohjelmakoodissa käytettävät funktiot ovat yleisiä, joten koodi toimii myös aikaisemmilla versioilla.

Työkalua aloitettiin rakentamaan perusteista alkaen vaiheittain. Kollegat antoivat palautetta ja ohjausta Työkalun toiminnollisuuteen. Työkaluun lisättiin ominaisuuksia, kun entiset ominaisuudet toimivat. Ohjelmoinnissa käytettiin siis kokeilu ja erehdys -taktiikkaa.

8.2 AutoCADin ja Excelin välinen dataintegraation toteutus

Internetissä ei ole saatavilla tietoa AutoCADin ja Excelin välisestä integraatiosta. AutoCADista siirretään tietoa ulkoiseen tiedostoon kolmella tavalla, joista kahta ei voitu hyödyntää tässä laajuudessa.

Dataa voidaan siirtää erilliseen tiedostoon DataExtraction-komennolla. Komento vie objektit, jotka on valittu CAD-kuvassa, ja kirjoittaa ne helposti muokattavaan CSV-taulukkoon. Tarvittaessa tuotavia objekteja ja tietoa voi rajata dataa vietäessä. AutoCADissa valmiiksi määritetyllä komennolla ei voida tuoda dataa sisään taulukkomuodossa, joten tämä vaihtoehto jätettiin käyttämättä. Datan määrä ja jäsentely taulukossa sekä tuonti takaisin CAD-kuvaan estää tämän tavan hyödyntämisen.



Kuva 12. Tietojen vienti CSV-taulukkoon

1	Attribute Definition	kannajuh	white		0	(0)	(0)	1038.00	15.00	0.00	0	NESTE'S D	0.0	isocpeur3	D6_DRAWING_TITLE6
1	Attribute Definition	kannajuh	white		0	(0)	(0)	1038.00	22.00	0.00	0	DRAWING I	0.0	isocpeur3	D5_DRAWING_TITLE5
1	Attribute Definition	kannajuh	white		0	(0)	(0)	1123.89	26.30	0.00	0	NESTE'S D	0.0	isocpeur3.5	E1_Drawing_number
1	Attribute Definition	kannajuh	white		0	(0)	(0)	1152.68	12.25	0.00	0	SHEET NR.	0.0	isocpeur2.5	E3_Sheet_nr
1	Attribute Definition	kannajuh	white		0	(0)	(0)	1132.35	12.25	0.00	0	LAST REVI	0.0	isocpeur2.5	E2_LAST_REVISION
1	Attribute Definition	kannajuh	white		0	(0)	(0)	1038.00	50.00	0.00	0	SITE NAME	0.0	isocpeur3	D1_Drawing_title1
1	Line	kannajuh	white			(0)	(0)								
1	Attribute Definition	kannajuh	white		0	(0)	(0)	1038.00	43.00	0.00	0	PRODUCTI	0.0	isocpeur3	D2_Drawing_title2
1	Attribute Definition	kannajuh	white		0	(0)	(0)	1038.00	29.00	0.00	0	DRAWING I	0.0	isocpeur3	D4_DRAWING_TITLE4
1	Attribute Definition	kannajuh	white		0	(0)	(0)	1038.00	36.00	0.00	0	PROCESS I	0.0	isocpeur3	D3_Drawing_title3

Kuva 13. Esimerkki CSV-taulukon tiedosta

Muita tapoja integraatioon ovat ATTIN- ja ATTOUT-komennot tai Express Tools -valikon Attribute Import ja Attribute Export -toimintojen käyttö. Tällöin tehdään VBA-ohjelma, joka tekee laitteista blokin, jossa on tarvittavat tiedot attribuutti-muodossa, ja se siirretään txt-tiedostoksi ATTOUT-komennolla. Excelissä avataan txt-tiedosto muokkausta varten. Muokatut tiedot siirretään takaisin CAD-kuvaan ATTIN-komennolla.

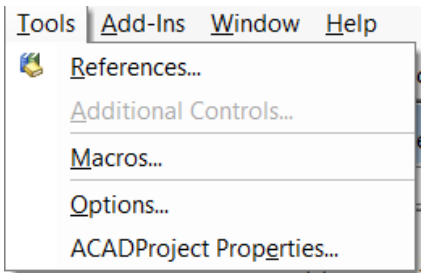
```
HANDLE   BLOCKNAME        KOJENO_68232    KOJENO_68233
'6400    KOJE15  JG       7519
'63F2    KOJE15  JG       7511
'63E4    KOJE15  JG       7510B
'637F    KOJE15  JG       7510A
```

Kuva 14. Attribuutit TXT tiedostossa

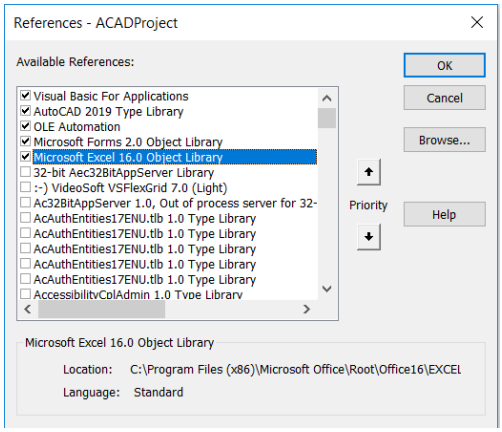
Tiedon siirtämisen jälkeen ATTOUT- ja ATTIN-komennoilla saadaan siirrettyä laitteiden nimi ja Handle-tunnus, mutta ei muuta tärkeää tietoa kuten koordinaatteja tai tasoa. Tiedot eivät ole riittäviä opinnäytetyön toteutukseen, joten tämäkin tapa hylättiin hankalan toteutustavan ja puuttuvien tietojen takia.

Ainoan jäljelle jääneen vaihtoehdon on toimittava siten, että Excelin ja AutoCADin välisen yhteyden on toimittava sulavasti VBA-funktioilla ilman erillisiä txt-tiedostoja ja ylimääräisiä datataulukkoja.

Excelin ja AutoCADin välistä dataintegraatiota tehtiin VBA-kirjastoja aktivoimalla. Oletuksena AutoCADin VBA:ssa ei ole Excel-kirjastoa, vaan käyttäjän tulee aktivoida kirjastot käyttöön References-kohdasta VBA-editorissa.



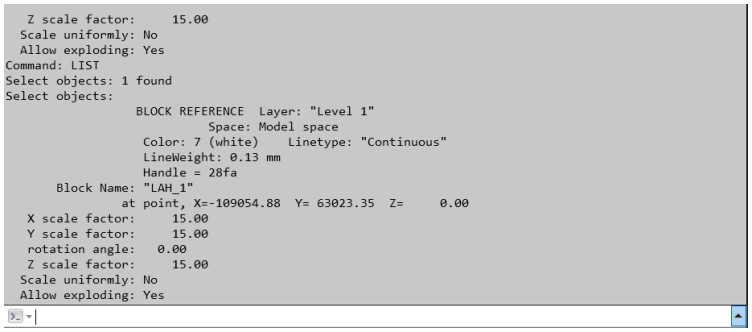
Kuva 15. Excel-kirjastojen aktivointi



Kuva 16. Excel-kirjastojen aktivointi

Aktivoinnin jälkeen Excelin VBA-funktiot toimivat AutoCADissa. OLE-automaatio varmistaa, että funktiot toimivat ohjelmien välillä.

Integraation rakentaminen muiden kuin ATTIN- ja ATTOUT-komentojen pohjalta vaatii ymmärryksen, miten CAD-kuvassa yksilöidään objektit. Jokaisella graafisella oliolla CAD-kuvassa on yksilöivä ObjectID- sekä Handle-tunnus. ObjectID-tunnus muuttuu aina tiedostoa avattaessa mutta Handle-tunnus pysyy aina samana. Handlea tarkastellaan LIST-komennolla ja objektin valinnalla.



Kuva 17. Esimerkki LIST-komennon antamista tiedoista.

Objekteja yksilöidään Handlen pohjalta. Integraatio toteutetaan tavalla, jossa objekteissa oleva Handle ja muu muokattava tieto siirretään Excel-taulukkoon. Kun objektit ovat Excel-taulukossa, objekteja valitaan yksitellen HandleToObject-funktiolla ja tehdään tarvittavat muutokset objekteille.

8.3 Työkalun koodin perustoteutus

Objektien valinta ja tuonti Excel-taulukkoon toteutetaan SelectionSetillä eli objektien valintaruudulla. Kun objektit on valittu, eritellään objektien tyypit yksitellen. Objekteja ei käsitellä yhtenä kokonaisuutena vaan objektityyppi kerrallaan. Koodi tekee kuvaan SelectionSetin nimeltä ”sset”, josta voidaan käydä objekteja läpi Entityssä. Entityllä tarkoitetaan valittujen objektien ryhmää.

```
ThisDrawing.SelectionSets.Add ("sset")
Set ss = ThisDrawing.SelectionSets("sset")
ss.SelectOnScreen
For Each ent In ss
Select Case ent.ObjectName
Case Is = "AcDbMText"
Set mtext = ent
mtext.layer = tasoMSG
ThisDrawing.Regen (True)
worksht.Cells(lastrow, 26).Value = "'" & mtext.handle
worksht.Cells(lastrow, 13).Value = "'" & mtext.TextString
```

Esimerkkikoodi 19. Ote objektien valintakoodista

Kun käyttäjä on valinnut objektin, esimerkkikoodi 19 käy jokaisen objektin läpi ja vertailee Entityssä olevien objektien nimiä. Kun koodi löytää AcDbMText-nimisen objektin Entityssä, se asettaa objektin MTextiksi. Tämän jälkeen koodi asettaa Excel-taulukon soluun tekstirivin ja Handle-tunnuksen ja siirtyy eteenpäin objektien valinnassa. Samanlainen proseduuri tehdään kaikille kuvassa oleville eri objektityypeille eri tavalla riippuen objektityypeistä.

TypeOf-rakenteella objekteja voidaan vertailla, mutta jonkin syyn takia tätä toimintoa ei saatu toimimaan. Objektien etsiminen Entityssä toteutettiin nimiä vertailemalla. Nimien vertailu toimii yhtä hyvin kuin objektien vertailu suoraan. Objektien nimien pitää olla täysin samanlaisia kuin Entityssä on määritelty. Esimerkiksi AcDbMText ja AcDbmText ovat eri asioita tekstien vertailun kannalta.

Tietojen haku Handlen avulla Excel-taulukosta on helppoa, koska nimiä ei haeta eikä vertailla vaan vertailu suoritetaan objektitasolla. Työkalun ohjelmakoodi rakennetaan TypeOf-rakenteella, ja HandleToObject-funktiolla.

```
handle = (“syötä handle”
Set object = ThisDrawing.HandleToObject(handle)

ElseIf TypeOf object Is AcadMText Then
```

```
Set mtext = object
elevation = worksht.Cells(i, 6).Value
insertionPoint(0) = mtext.insertionPoint(0)
insertionPoint(1) = mtext.insertionPoint(1)
insertionPoint(2) = elevation
mtext.insertionPoint = insertionPoint
```

Esimerkkikoodi 20. Ote koodin toisesta osasta.

Kun Handle tiedetään, objekteja valitaan yksitellen Excel-listan mukaisesti ja vertaillaan objektien tyyppejä esimerkkikoodi 20 mukaisesti. Kun oikea objektityyppi on löydetty, objekti asetetaan mtextiksi, sen paikkaa muokataan ja sille syötetään Z-koordinaatti. Z-koordinaattia ei aseteta suoraan, vaan X ja Y-koordinaatit valitaan ensin objektista ja Z-koordinaatti Excel-taulukosta.

```
handle = excel.activesheet.cells(i,10)
Set object = ThisDrawing.HandleToObject(handle)
taso = excel.ActiveSheet.Cells(i, k).Value
object.layer = taso
object.Update
```

Esimerkkikoodi 21. Esimerkkikoodi Z-koordinaatin asettamisesta

Esimerkkikoodi 21 valitsee kuvasta Handlen, asettaa sen objektiksi ja etsii tason Excel-taulukosta sekä asettaa ja päivittää objektin tason Excel-taulukon mukaisesti.

Opinnäytetyö rakennettiin näiden kahden koodiesimerkin pohjalta. Koska sijoituskuvien variaatioita on runsaasti, Työkalu kerää tietoa monista erilaisista objektityypeistä. Työkalu rakennettiin tukemaan lähes kaikki mahdolliset objektityypit. Työkalulla luetaan myös tekstejä blokin sisällä olevista attribuuteista. Tiedot attribuutista mahdollistavat sen, että kaikki mahdolliset tiedot voidaan siirtää Exceliin.

9 Työkalun kehitysajatukset, yhteenveto ja pohdinta

Mielestäni opinnäytetyö onnistui hyvin. AutoCADin ja Excelin integrointi keskenään sujui hyvin. Opinnäytetyön tekemisessä AutoCADin nettisivujen ohjeet ja neuvot nopeuttivat koodin valmistumista. Opinnäytetyön onnistumista tuki myös VBA:n käyttäjäystävällinen ohjelmointikieli.

Koodaamisen suurin haaste oli attribuutteja sisältävien blokkien käsittelyssä, koska yhden blokin sisällä on monia eri määrittelyjä erikseen käsiteltävänä. Dataintegraation rakentaminen kävi helposti, kun ymmärsi, miten Autocad käsittelee objekteja, ja kun löysi tarvittavan funktion tiedon tuomiseen takaisin kuvaan.

Lähteet

Ambrosius, Lee. 2015. AutoCAD® Platform Customization. AutoCAD® Platform Customization. Indianapolis: John Wiley & Sons Inc.

Autodesk Inc. 2015. About Command Scripts. Verkkoaineisto. <https://knowledge.autodesk.com/support/autocad-lt/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/AutoCAD-LT/files/GUID-95BB6824-0700-4019-9672-E6B502659E9E-htm.html> 14.4.2017. Luettu 24.10.2018.

Autodesk Inc. 2017a. About Customizing AutoCAD With VBA. Verkkoaineisto. <https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2018/ENU/AutoCAD-Customization/files/GUID-2CD40631-D67B-4DF0-A2C4-606E9B613252-htm.html> 14.4.2017. Luettu 23.10.2018.

Autodesk Inc. 2017b. Create a Circle Object (.NET). Verkkoaineisto. <http://help.autodesk.com/view/ACD/2017/ESP/?guid=GUID-25923EE3-1C93-4A1E-9A02-5EB4EAF5109F> Luettu 21.11.2018.

Autodesk Inc. 2018a. AutoCAD vs AutoCAD LT. Verkkoaineisto. <https://www.autodesk.fi/compare/autocad-vs-autocad-lt> Luettu 17.10.2018.

Autodesk Inc. 2018b. About ObjectARX Applications. Verkkoaineisto. <https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2019/ENU/AutoCAD-Customization/files/GUID-3FF72BD0-9863-4739-8A45-B14AF1B67B06-htm.html> 21.05.2018. Luettu 9.10.2018.

Caputo, Linda. & Cai, Saisang. 2017a. Getting Started with VBA in Office. Verkkoaineisto. <https://docs.microsoft.com/en-us/office/vba/library-reference/concepts/getting-started-with-vba-in-office> 8.6.2017. Luettu 10.10.2018.

Caputo, Linda. & Cai, Saisang., 2017b. Declaring Variables. Verkkoaineisto. <https://docs.microsoft.com/en-us/office/vba/language/concepts/getting-started/declaring-variables> 8.6.2017. Luettu 22.10.2018.

Caputo, Linda. & Cai, Saisang. 2017c. For...Next Statement. Verkkoaineisto. <https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/fornext-statement> 8.6.2017. Luettu 22.10.2018.

Caputo, Linda. & Cai, Saisang. 2017d. GoTo Statement. Verkkoaineisto. <https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/goto-statement> 8.6.2017. Luettu 11.10. 2018.

Caputo, Linda. & Cai, Saisang. 2018a. Data type summary. Verkkoaineisto. <https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/data-type-summary> 24.8.2018. Luettu 22.10.2018.

Caputo, Linda. & Cai, Saisang. 2018b. If...Then...Else statement. Verkkoaineisto. <https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/ifthenelse-statement> 24.8.2018. Luettu 15.10.2018.

Caputo, Linda. & Cai, Saisang. 2018c. UserForm object. Verkkoaineisto. <https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/userform-object> 12.11.2018. Luettu 15.11.2018.

Dollard, Kathleen. Wenzel, Maira. & Hoffman, Matt. 2015. Option Explicit Statement (Visual Basic). Verkkoaineisto. <https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/option-explicit-statement> 20.7.2015. Luettu 27.10.2018.

Insta Group Oy, Instan tarina. Verkkoaineisto. <https://www.insta.fi/insta-group/historia.html> Luettu 10.09.2018.

Kelly, Paul. 2015. The Complete Guide to the VBA Sub. Verkkoaineisto. <https://excelmacromastery.com/excel-vba-sub/> 13.4.2015. Luettu 22.10.2018.

Kelly, Paul. 2016. Excel 2016 Macro Mastery. Verkkoaineisto. <https://excelmacromastery.com/vba-class-modules/> 9.9.2016. Luettu 10.10.2018.

Kelly, Paul. 2018. VBA Dim – A Complete Guide. Verkkoaineisto. <https://excelmacromastery.com/vba-dim/> 6.3.2018. Luettu 22.10.2018.

Kennedy, Luke. 2014. A Brief History of AutoCAD. Verkkoaineisto. <https://www.scan2cad.com/tips/autocad-brief-history/> 5.1.2014. Luettu 20.10.2018.

Kymdata Oy. 2018. Instrumentointi. Verkkoaineisto. <http://www.cads.fi/index.php/ohjelmistot/cads-electric/teollisuuden-sahko-ja-automaatiosuunnittelu/instrumentointi> Luettu 1.11.2018.

Matt, Hoffman. Wenzel, Maira., B, M. & Latham, L., 2015a. Do...Loop Statement (Visual Basic). Verkkoaineisto. <https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/do-loop-statement> 20.7.2015. Luettu 11.10.2018.

John Paul Mueller. Standard VBA data types. Verkkoaineisto. <https://www.dummies.com/programming/visual-basic/standard-vba-data-types/> Luettu 9.10.2018.

Nicholson, Miles. 2016. AutoCAD 2016 – A Brief History Of AutoLISP And Its Functionality. Verkkoaineisto. <https://www.cadlinecommunity.co.uk/hc/en-us/articles/206927309-AutoCAD-2016-A-Brief-History-Of-AutoLISP-And-Its-Functionality> 20.7.2015. Luettu 24.10.2018.

Kathleen Dollard, Maira Wenzel, Alfred Myers, Matt Hoffman, Mike Blome. On Error Statement (Visual Basic). Verkkoaineisto. <https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/on-error-statement>. 20.5.2015. Luettu 10.9.2018.

Oraviita, J. 2017. Automaation ja instrumentoinnin dokumentointisuunnittelu. Insinöörityö. Metropolia Ammattikorkeakoulu. Theseus-tietokanta

Wenzel, Maira. & Hoffman, Matt. 2015b. Select...Case Statement (Visual Basic). Verkkoaineisto. <https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/select-case-statement> 20.7.2015. Luettu 11.10.2018.

Wenzel, Maira. Hoffman, Matt. & Latham, L., 2015c. With...End With Statement (Visual Basic). Verkkoaineisto. <https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/with-end-with-statement> 20.7.2015. Luettu 13.10.2018.

Wenzel, Maira. Mackiovello, E., Blome, M. & Hoffman, M., 2015d. For Each...Next Statement (Visual Basic). Verkkoaineisto. <https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/for-each-next-statement> 20.7.2015. Luettu 13.10.2018.