CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Theseus

OULUN AMMATTIKORKEAKOULU

Janne Paaso

SOLIDWORKSIN MAKROJEN KÄYTTÄMINEN PARAMETRISESSA MALLINTAMISESSA

SOLIDWORKSIN MAKROJEN KÄYTTÄMINEN PARAMETRISESSA MALLINTAMISESSA

Janne Paaso
Opinnäytetyö
Kevät 2016
Kone- ja tuotantotekniikka
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Kone ja tuotantotekniikan koulutusohjelma, tuotantotalous

Tekijä: Janne Paaso
Opinnäytetyön nimi: SolidWorksin makrojen käyttäminen parametrisessä mallinnuksessa.
Työn ohjaaja: Helena Tolonen
Työn valmistumislukukausi ja -vuosi: kevät 2016 Sivumäärä: 55 + 2 liitettä

Opinnäytetyössä tutkittiin SolidWorksin makrojen käytettävyyttä parametrisessä mallinnuksessa. Työn makrojen ohjelmointikieleksi valittiin Visual Basic for Applications. Tavoitteena oli selvittää, kuinka hyvin VBA-makrot soveltuvat parametriseen mallinnukseen ja mitkä ovat niiden rajoitteet. Toisena tavoitteena oli ideoida pilottimakro ja tehdä se VBA-kielellä. VBA-makroilla voidaan automatisoida SolidWorks-ohjelmaa. Työ tehtiin Oulun ammattikorkeakoululle.

Työn alussa selvitettiin, miten SolidWorksin VBA-makroja tehdään. Tämän jälkeen selvitettiin, miten VBA toimii ohjelmointikielenä. Seuraavana pohdittiin, miten VBA-makroja voidaan hyödyntää parametrisessä mallinnuksessa. Tämä jälkeen tehtiin työhön harjoituspilottimakro, joka on BOX 3.0 -makro. Varsinaisen pilottimakron aiheeksi ideoitiin skaalattava ihmismallimakro. Pilottimakron nimi on Human model scaler -makro. Työn lopussa pohdittiin, miten yritykset nykyään käyttävät SolidWorksin makroja ja millaista hyötyä VBA-makroilla voidaan tuottaa yrityksille.

Työn tuloksena saatiin toimiva ihmismallimakro. VBA-makroilla voidaan tuottaa ajansäästöä parametriseen mallinnukseen. Ajansäästö saavutetaan automatisoimalla toistuvia toimintoja mallintamisessa. Työssä selvisi, että VBA-makrojen käyttäminen eri kokoonpanoissa tuottaa vaikeuksia, koska kokoonpanojen yhteensopivuus on heikko. Yritykset voivat saavuttaa ajansäästön ja inhimillisten virheiden vähenemisen kautta huomattavaa taloudellista etua. Tämä etu saavutetaan käyttämällä VBA-makroja 3D-mallien luomiseen tai tiedostohallinnan avustamiseen.

Asiasanat: SolidWorks, API, VBA, makro

SISÄLLYS

SANASTO

body	kokonaisuus, joka sisältää osan kaikki piirteet.
deklaraatio	ilmaisee muuttujan tiedostotyypin makrossa.
makro	sarja tallennettuja komentoja.
mate	luo geometrisen relaation kahden eri kokoonpanon osan välille.
merkki	kaikki kirjaimet, numerot ja symbolit ovat merkkejä.
Pack and Go	SolidWorks-toiminto, joka kerää kaikki tiedostoon liittyvät tiedostot yhteen kansioon tai ZIP-tiedostoon.
properties	sisältää SolidWorks-tiedoston ominaisuustietoja.
rakentaa	tiedosto suorittaa kaikki siihen liittyvät laskutoiminnot uudelleen.
Sub-proseduuri	sarja komentoja Sub- ja End Sub -rivien välissä, tunnetaan myös nimellä aliohjelma.

1 JOHDANTO

Tässä opinnäytetyössä tutkitaan SolidWorks-3D-CAD-suunnitteluohjelmiston VBA-makrojen käytettävyyttä parametrisessä mallintamisessa. Työn tavoitteena on selvittää, kuinka hyvin VBA-makrot soveltuvat parametriseen mallinnukseen ja mitkä VBA-makrojen rajoitteet ovat. Työn tavoitteisiin kuuluu ideoida sopiva pilottimakron aihe ja tehdä ideoitu makro VBA-kielellä. Tämän työn tilaaja on Oulun ammattikorkeakoulu. Koulun yhteyshenkilö on lehtori Jari Viitala ja ohjaava opettaja on lehtori Helena Tolonen.

SolidWorks-3D-suunnitteluohjelmistoon tehdyillä makroilla voidaan automatisoida monia suunnittelutyön toimintoja. VBA on helppolukuista, ja se on helppo oppia, koska VBA-kielellä ohjelmoitaessa ei tarvita käyttää matematiikkaa. VBA-kieli on helppolukuista, koska VBA-kielen komennot ovat loogisesti nimettyjä. Tämän takia työn ohjelmointikieleksi valittiin Visual Basic for Applications eli VBA. Työssä käytetään SolidWorksin vuoden 2015 versiota ja ohjelmointikielen VBA 6.0 -versiota. Työhön liittyvät makrot kirjoitetaan SolidWorksiin asennetun SolidWorks -API SDK (Application Programming Interface, Software Development Kit) -lisäosan avulla. Työssä ohjelmoidaan myös Microsoft Excelin vuoden 2013 versiota. Työn lopussa pohditaan, miten VBA-makroja on käytetty yrityksissä ja millaista hyötyä VBA-makroilla voidaan tuottaa yritykselle.

2 SOLIDWORKS

SolidWorks on Dassault Systems SolidWorks Oy:n kehittämä tietokoneavusteinen suunnitteluohjelmisto eli CAD-ohjelmisto. SolidWorks-ohjemassa on helppokäyttöinen käyttöliittymä ja suuri työkaluvalikoima. Ensimmäinen SolidWorks julkaistiin vuonna 1995, ja nykyään se on monen suunnittelualan yrityksen valitsema työkalu suunnittelu- ja insinöörityöhön. (1.)

2.1 Makron tallennus

Kaikista yksikertaisin ja nopein tapa tehdä makro on käyttää *Record macro* -toimintoa SolidWorksissä. Makroihin liittyvät toiminnot löytyvät SolidWorksissä *Tools<Macro*-valikon alta. *Record macro* -toiminnolla on hyvä aloittaa makrojen toiminnan opiskelu. *Record macro* -toiminnolla on mahdollista tehdä yksinkertaisia makroja, jotka eivät sisällä logiikkaa. Toiminnon käyttäminen on hyvin yksinkertaista. Ensimmäiseksi painetaan *Record*-painiketta, minkä jälkeen suoritetaan halutut toiminnot, joita makrossa halutaan olevan. Makron tallennus lopetetaan painamalla *Stop*-painiketta. Tämän jälkeen voidaan valita, mille kielelle makro tallennetaan.

Taulukossa 1 näkyvät vaihtoehdot tallennusmuodoista ja tieto, millä kielellä ne tallentuvat kyseisessä tiedostotyypissä. *Record macro* -toiminnolla on helppo löytää komentojen nimiä ja niiden käyttötapoja. Esimerkiksi jos halutaan tietää, miten reunakäyrän pyöristys tehdään osatiedostoon, voidaan nauhoittaa makro, johon on tallennettuna kyseinen toiminto. Kyseisen toiminnon käytön ja komennon nimet saadaan selville avaamalla nauhoitettu makro *Edit macro* -painikkeella.

TAULUKKO 1. SolidWorksin tukemat makrojen tallennusmuodot (2)

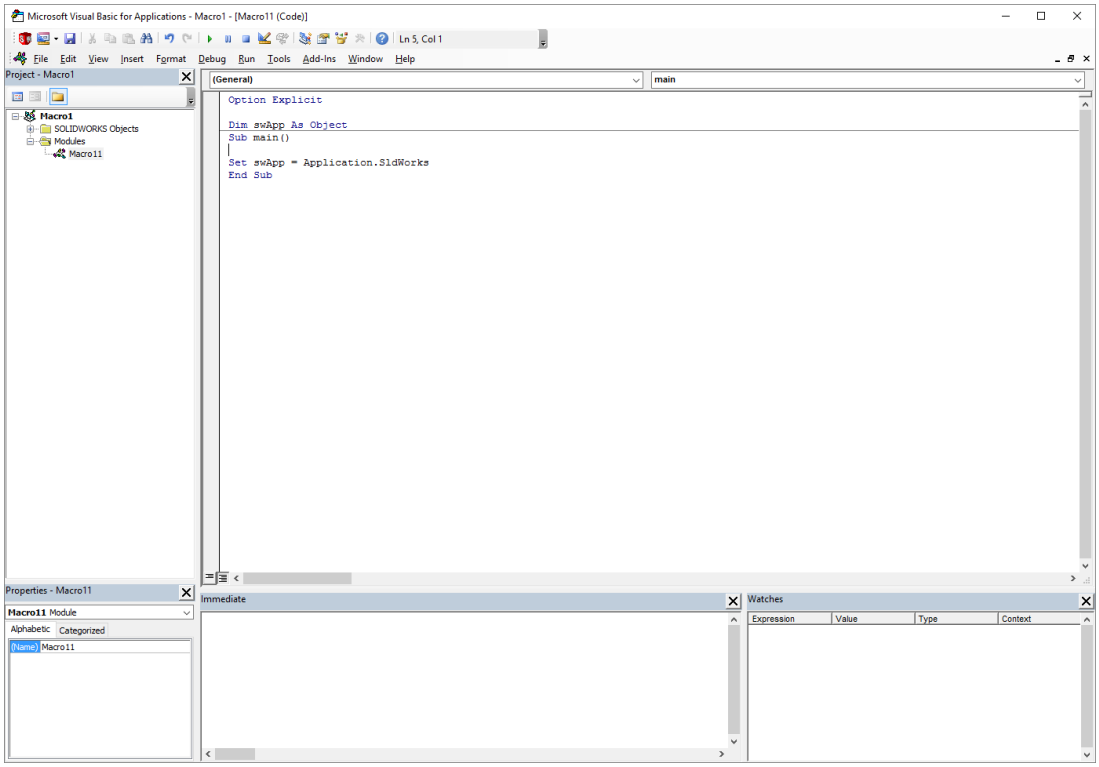
Kieli	Tiedostopääte
VBA	.swb
C#	.cdproj
VB.Net	.vbroj

Record macro -toiminnolla ei voi tehdä makroja, joissa on logiikkaa tai jos makro on monimutkainen. *Record macro* -toiminnolla logiikan tekeminen on mahdotonta, koska nauhoitettu makro toistaa samat toiminnot samassa järjestyksessä kuin makroon on ne nauhoitettu (3). Logiikkaa sisältävät makrot täytyy kirjoittaa käsin. *Record macr*o -toiminnolla makroon tulee runsaasti turhaa koodia, sillä makroon tallentuu jokainen tallennuksen aikainen toiminto. Esimerkiksi käännettäessä kuvakulmaa tallennuksen aikana koodirivejä saattaa tulla makroon satoja. *Record macro* ei aina käytä kaikista järkevintä komentoa toimintoa tallennettaessa. *Record macro* -toiminto käyttää aina SelectByid2-komentoa, joka ei ole kaikista järkevin vaihtoehto kaikissa tapauksissa. Asiasta kerrotaan enemmän luvussa 5.1.

Jos halutaan nauhoittaa VB.NET- tai C# -ohjelmointikielisiä makroja, tietokoneeseen täytyy olla asennettuna VSTA (Visual Studio for Applications) -sovellus ja Microsoft .NET Framework 3.5 (4). VSTA-sovellus on yleensä asennettuna automaattisesti Solidworksin mukana. VBA-makrot eivät tarvitse VSTA-sovellusta. (5.)

2.2 SolidWorks API

Record macro -toiminnon sijaan makroja voidaan kirjoittaa käsin. Tätä varten SolidWorks sisältää API (Application Programing Interface) -nimisen työkalun, jolla voidaan kirjoittaa ja muokata VBA-makroja. Tähän työkaluun pääsee käsiksi menemällä SolidWorksissä *Tools<Macro<New macro*- tai *Edit macro* -painikkeesta. Kuvassa 1 näkyy API-sovelluksen käyttöliittymä, jossa ovat kaikki tärkeimmät ikkunat auki.



KUVA 1. Solidworksin API-sovelluksen käyttöliittymä

Vasemmassa laidassa on Project Explorer -ikkuna. Tässä ikkunassa näkyy kaikki avoinna olevat makrot sisältöineen. Project Explorer -ikkunan alla on Properties-ikkuna, jossa on tietoa valitusta moduulista ja proseduurista (6). Isoin ikkuna kuvassa 1 on Code-ikkuna. Tässä ikkunassa on aktiivisen moduulin sisältämä koodi (7). Alalaidassa keskellä on Immediate-ikkuna. Tähän ikkunaan voidaan tulostaa tietoa käyttämällä Debug.Print-komentoa (liite 2/1). Immediate-ik-

kunaa käytetään pääasiassa muuttujan sisältämän arvon tarkastamiseen ja virheen etsintään koodista (8). API sisältää toiminnon nimeltään tapahtuma laukaisin (event trigger). Tällä toiminolla voidaan suorittaa makro jonkin tapahtuman seurauksena. Tällä toiminolla voidaan esimerkiksi lisätä tiedoston nimen perään versionumero. Oikeassa alakulmassa olevasta Watches-ikkunasta näkee kaikki makron aktiiviset laukaisimet (9).

2.3 Makron kirjoittaminen

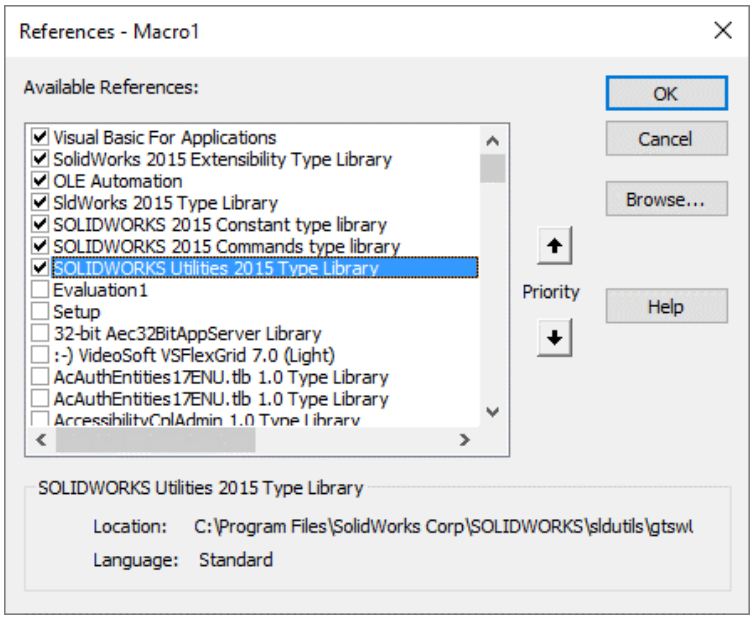
Hyvin kirjoitettu makro on modulaarinen ja hyvin optimoitu. Modulaarisesta koodista kerrotaan lisää luvussa 3.8. Makron koodi on hyvin optimoitu, kun makron tarkoitus saadaan suoritettua mahdollisimman vähillä koodiriveillä ja makro käyttää mahdollisimman vähän keskusmuistia. Hyvin optimoitu makro on myös nopea suorittaa.

Suuri ongelma makroa kirjoittaessa on tietää, mitä komentoja käyttää. Tätä varten SolidWorks sisältää API help (Application Programing Interface help) -ohjekirjan. API helpistä löytyy kaikki komennot esimerkkeineen, joita SolidWorksin makrot voivat käyttää. API helpin käyttäjän kannattaa laittaa *use SOLIDWORKS web help* pois päältä. Tällä varmistetaan sen, että API help -ohjekirja on saman SolidWorks-version ohjekirja, mikä käyttäjällä on. SolidWorks API help sisältää monta versiota samasta komennosta, koska vanhalla VBA-kielen versiolla kirjoitettujen makrojen täytyy toimia uudemmissa versioissa.

SolidWorksin API-sovelluksessa on Microsoft Visual Basic for Application help. Täältä löytyvät kaikki VBA-kielen toiminnot, jotka eivät ole sidoksissa SolidWorks-ohjelmaan. Nämä kaksi helpiä auttavat makron tekijää löytämään kaikki hänen tarvitsemansa toiminnot. SolidWorksin foorumit ovat hyvä informaation lähde, jos käyttäjä haluaa tarkempia selityksiä toimintojen käytöstä.

2.4 Referenssit

API-sovellusreferenssit täytyy asettaa oikein, jotta SolidWorksin API-sovellus tietäisi, mitä mikäkin komento tarkoittaa. Referenssivalikko löytyy API-sovelluksessa *Tools<References* -painikkeella. Kuvassa 2 näkyy kyseinen valikko. Valikossa on automaattisesti valittuna kaikki referenssit, mitä SolidWorksin yleisimmät komennot tarvitsevat. Jos koodissa halutaan käyttää SolidWorksin lisäosia, kuten Motion Study -lisäosaa, listasta täytyy käydä etsimässä *SOLIDWORKS 2015 MotionStudy Type Library* ja valita tämä referenssi. Kyseiset tiedostot ovat DLL (dynamic link library) -tiedostoja. Nämä tiedostot sisältävät dataa siitä, mitä eri API-komentojen tulisi tehdä (10). Luvussa 3.8 käsitellään esimerkki siitä, miten referenssivirhe korjataan.



KUVA 2. SolidWorksin API-sovelluksen referenssivalikko

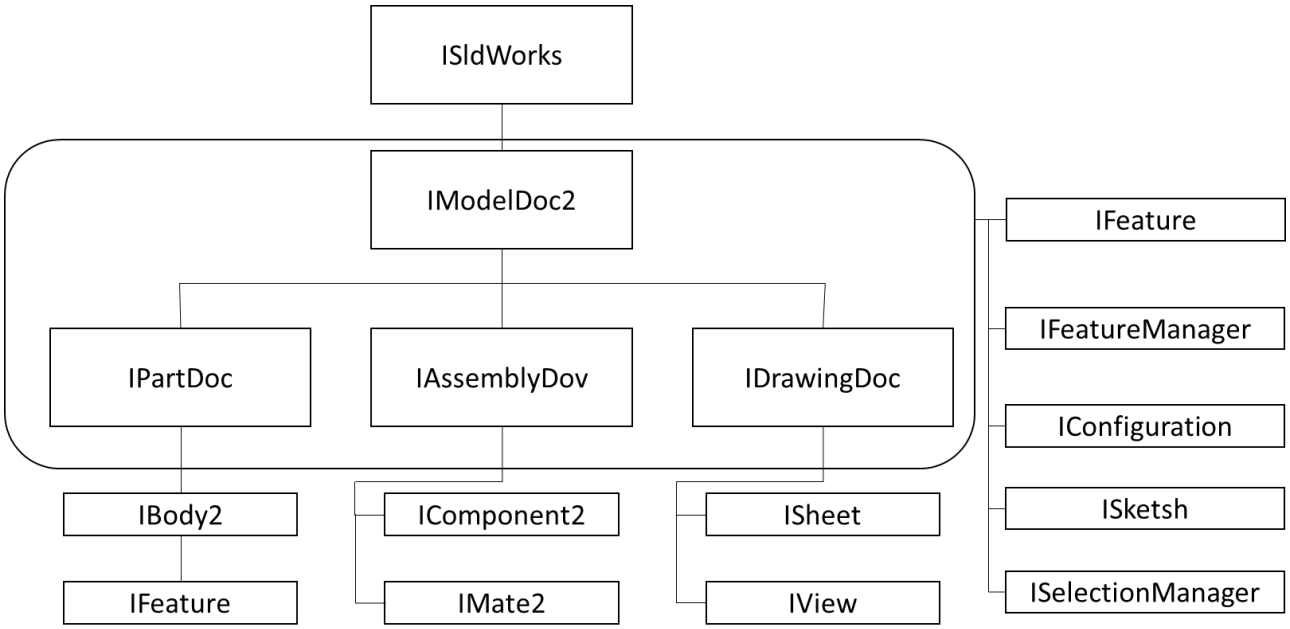
3 VBA

VBA eli Visual Basic for Applications on Microsoftin kehittämä ohjelmointikieli. VB 1.0 julkaistiin vuonna 1991 ja se rakennettiin vanhempaan BASIC-kieleen pohjautuen (11). Nyt VBA-kielestä on tehty VBA 7.0 -versio, joka julkaistiin Microsoft Office 2010:n mukana. Tässä työssä kuitenkin käytetään VBA 6.0 -versiota, koska SolidWorks 2015 tukee vain VBA 6.0 -versiota (12).

3.1 SolidWorksin objektimalli ja rajapinnat

Jotta pystyttäisiin käyttämään SolidWorksin APIta tehokkaasti, makron kirjoittajan täytyy ymmärtää, mitä tarkoittaa SolidWorksin objektimalli ja millainen on sen hierarkia. SolidWorksin API käsittelee objektina kaiken sen, mitä SolidWorksissä voidaan käsitellä. API-sovellus käsittelee SolidWorks-ohjelmaa yhtenä objektina. Tämä objekti on ISldWorks-rajapinta, ja se on ylin objekti SolidWorksin objektihierarkiassa. (13.)

Jotta makron kirjoittaja voisi käsitellä oikeaa objektia, kirjoittajan täytyy navigoida oikeaan rajapintaan (interface), joka sisältää halutun objektin käsittelyyn liittyvät komennot. Esimerkiksi jos kirjoittaja haluaa makron muokkaavan reunakäyrään asetettavaa pyöristystä, kirjoittajan täytyy navigoida IEdge-rajapintaan ja käyttää siellä olevia oikeita komentoja (14). Kuvassa 3 näkyy, millainen hierarkia logiikka SolidWorksissä on. Kuvan 3 hierarkia ei ole lähellekään täydellinen, sillä näitä rajapintoja on SolidWorksin API-sovelluksessa 330. Rajapinta sisältää kaikki sille asetetut metodit ja ominaisuudet (15). Esimerkiksi jos halutaan käsitellä luonnostelman (sketch) spline-viivaa (käyräviiva), makron täytyy navigoida I*SldWorks<IModelDoc2<ISketchSegment<ISketchSpline* -rajapintaan.



KUVA 3. Solidworksin käyttöliittymä hierarkia (16)

Jotta Solidworks API voi käyttää rajapinnan metodeja ja ominaisuuksia, makron täytyy valita ensin oikea objekti. Esimerkiksi jos halutaan muuttaa osan reunakäyrän pyöristystä, reunakäyrä pitää olla valittuna. Yleinen sääntö makron teossa on tehdä kaikki toiminnot samassa järjestyksessä kuin normaalisti SolidWorksillä mallinnettaessa. Tämä ei kuitenkaan tarkoita sitä, että kaikki valinnat pitäisi tehdä makron komennoilla. Makron suorittamisen aikana käyttäjä voi tehdä useita valintoja SolidWorksissä. (17.)

3.2 Rajapinnan perintä

SolidWorksin APIssa on käytännöllinen tärkeä toiminto nimeltään rajapinnan perintä (interface inheritance). Tämä toiminto sallii joidenkin rajapintojen käyttävän muiden rajapintojen toimintoja ja ominaisuuksia. (public properties). Seuraavassa on lista rajapintoja, jotka voivat periä tietoa muista rajapinnoista:

- IModelDoc2
  - IAssemblyDoc, IDrawingDoc ja IPartDoc
- IEntity
  - IEdge, IFace2, IFeature, ILoop ja IVertex

- ITableAnnotation
  - IBomTableAnnotation, IHoleTableAnnotation, IRevisionTableAnnotation, ja IWeldmentCutListAnnotation
- IFeature
  - IAttribute
- ISketchSegment
  - ISketchArc, ISketchEllipse, ISketchLine, ISketchParabola, ISketchPoint, ISketchSpline ja ISketchText
- IPropertyManagerPageControl
  - IPropertyManagerPageActiveX, IPropertyManagerPageBitmap, IPropertyManagerPageBitmapButton, IPropertyManagerPageButton. (18.)

IModelDoc2 on yleisin rajapinta, koska sitä joudutaan käyttämään aina, kun tehdään SolidWorksin osiin liittyen. IModelDoc2 on niin laaja rajapinta, että siihen on täytynyt tehdä lisäosa. Tämän lisäosan rajapinnan nimi on IModelDocExtension. IModelDoc-rajapinta sisältää komentoja, jotka ovat kaikki vanhentuneita. (19.)

3.3 Muuttujien deklaraatio

Kaikissa makroissa on tehty muuttujien deklaraatiota, jolloin määritellään SolidWorksin API-sovellukselle muuttujan datatyppi. Tämä toiminto ilmaisee tietokoneelle, kuinka paljon sen pitää varata muistitilaa muuttujan tyypille. Taulukossa 2 näkyy tärkeimmät datatyypit, mitä SolidWorksin makroissa käytetään. (20.)

TAULUKKO 2. Yleisimmät käytetyt datatyypit SolidWorksin makroissa ja niiden muistin käyttö. Taulukossa x tarkoittaa sitä, että muistin käyttö on riippuvainen käyttöjärjestelmästä (21)

Muuttujan tyypin nimitys	Sisältö	Muistinkäyttö (bittiä)	Hungarian notiation
String	Merkkejä.	x	str
Boolean	True (1) tai False (0).	x	bl
Date	Päivämäärä- ja aika-arvot 0:00:00 01.01.0001 ja 11:59:59 31.12.9999 väliltä.	8	dat
Single	Numeroarvo 7 desimaalin tarkkuudella.	4	sng
Double	Numeroarvo 16 desimaalin tarkkuudella	8	db
Integer	Numeroarvo -32768 ja 32767 väliltä. Ei sisällä desimaalia.	4	int
Long	Numero -2147483648 ja 2147483648 väliltä.	8	lng
Object	Sisältää 32 bittisen polun käytettyyn objektiin. (esim. sldprt tiedosto.).	4 bittiä 32-bittisessä käyttöjärjestelmässä. 8 bittiä 64-bittisessä käyttöjärjestelmässä.	obj
Variant	Sisältää minkä tahansa datatyypin	x	v tai var
Desimal	Numeroarvo 29 desimaalin tarkkuudella.	16	des
Byte	Sisältää numeroarvon 0 ja 255	1	b

Muuttujan datatyypin määrittely eli deklaraatio voidaan tehdä joko Sub-proseduurin sisällä tai sen ulkopuolella. Kun muuttujan tyyppi ilmaistaan koodin sisällä, muuttuja ja sen tyyppi vaikuttavat vain Sub-proseduurissa, jossa määrittely on. Kun muuttuja nimitetään Sub-proseduurin ulkopuolella, se vaikuttaa kaikkiin Sub-proseduureihin kyseisessä moduulissa (20). Seuraavana on esimerkki deklaraatiosta, joka löytyy jokaisesta SolidWorksin liittyvästä makrosta.

Dim swApp As SldWorks.SldWorks

Sanan Dim tilalla voi olla myös Public tai Private. Taulukossa 3 näkyy, mitä ominaisuuksia näillä sanoilla on. Kaikki taulukon 3 sanat varaavat RAM-muistista muistipaikan kyseiselle deklaraatiolle.

TAULUKKO 3. Työssä käytetyt eri määritteet ja niiden käyttöoikeudet

Määrite	Käyttöoikeudet
Dim	Muuttujaa voidaan käyttää kyseisessä moduulissa, kun deklaraatio on tehty proseduurin ulkopuolella (22).
Private	Muuttujaa voidaan käyttää vain moduulin sisällä (23).
Public	Muuttujaa voidaan käyttää kaikissa makron moduuleissa (24).

Jotta makron kirjoittaja välttyisi väärinkäsityksiltä, makrossa kannattaa käyttää Option Explicit -toimintoa. Kun tämä toiminto on käytössä, jokaisen muuttujan deklaraatio täytyy tehdä erikseen. Jos Option Explicit -riviä ei ole, koodissa olevat kirjoitusvirheet aiheuttavat ongelmia. Esimerkiksi voi kirjoittaa MacroPth, kun tarkoitus olisi kirjoittaa MacroPath. Tässä tapauksessa SolidWorksin API varaa kummallekin versiolle MacroPath-termistä oman muistipaikkansa. Tällöin tieto, mitä MacroPath-muuttujaan oli tallennettu, ei löydy MacroPth-nimellä.

3.4 Koodin luettavuus

Makron tekijän kannattaa miettiä makron koodin ulkoasua, jos hän haluaa tehdä koodista helposti luettavan. Yksi tapa tähän on sisentää joitain rivejä koodista. Esimerkiksi if…then…else-lohkossa on hyvä sisentää lohkossa toistettavat rivit. Tarkempi esimerkki koodin sisentämisestä if…then…else-lohkossa on luvussa 4.1.

Helppo tapa estää virheitä koodin kirjoittamisessa on käyttää Hungarian Notiation -menetelmää. Menetelmässä jokaisen muuttujan alkuun liitetään lyhenne siitä, miksi datatyypiksi muuttuja on määritelty. Tässä työssä käytetyt datatyyppien lyhenteet näkyvät taulukossa 2. Menetelmässä käytetyt lyhenteet vaihtelevat paljon koodin kirjoittajasta mukaisesti. Käytännön tarkoitus on ilmaista, mikä on muuttujan datatyyppi, ettei koodin lukijan tarvitse joka kerta mennä katsomaan muuttujan tyyppiä deklaraatio-kohdasta. Monessa deklaraatiossa näkyvä sw viittaa SolidWorks-ohjelmaan. (25.)

3.5 Early Binding ja IntelliSense

Uusi makro tehdään valitsemalla SolidWorksissä *Tools<Macro<New*. Uudessa makrossa on jonkin verran koodia automaattisesti. Niistä tärkein rivi on ”*Dim swApp As Object”.* Tämä on Late binding -menetelmä. Muuttujaan on tehty Late binding, kun se on nimetty objekti-datatyypiksi. Kirjoittaessa makroja käsin on hyvä käyttää Early binding -menetelmää. Muuttujaan on tehty Early binding, kun siihen on määritelty rajapinnan nimi. Seuraavana on esimerkki siitä, miten Early binding tehdään. (26.)

Dim swApp As SldWorks.SldWorks

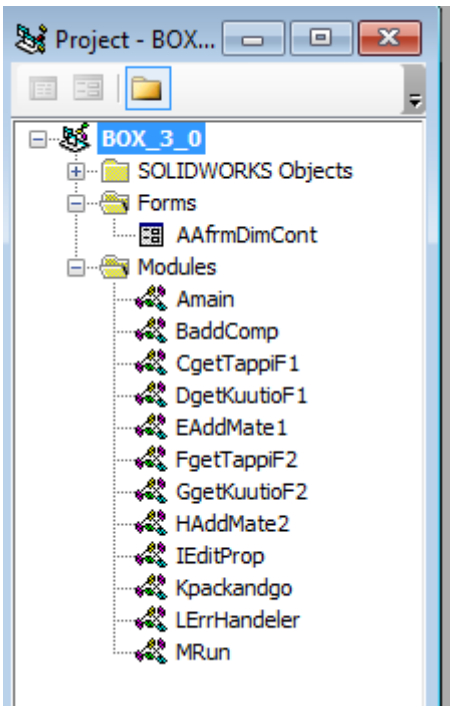
Early binding tehdään makron deklaraatio-osiossa. Early binding:n loppuosassa on aina *SldWorks*, jonka jälkeen tulee haluttu rajapinta, johon Early binding tehdään. Esimerkiksi IModelDoc2-rajapinnan Early binding on *Dim swModel As SldWorks.ModelDoc2.* Kun ”*Dim swApp As SldWorks.SldWorks”* on kirjoitettuna

makron proseduurin ulkopuolelle, se antaa mahdollisuuden käyttää IntelliSense -toimintoa. Edellä mainittu rivi yhdistää makron SldWorks type library -tiedostoon (dll-tiedosto) (27). IntelliSense-toiminto arvaa, mitä tekijä on kirjoittamassa ja ehdottaa automaattisesti täyttövaihtoehtoja. Menetelmä auttaa oikeinkirjoituksessa, komentojen etsinnässä ja virheiden korjaamisessa. IntelliSense -toiminnon antamat vaihtoehdot riippuvat siitä, mihin rajapintoihin Early binding on tehty (28).

3.6 Modulaarinen ohjelmointi

Makroa suunniteltaessa olisi hyvä pohtia makron askelrakennetta. Kun jokaiseen askeleeseen sijoittaa vain yhden toiminnon, esimerkiksi Pack and Go -toiminto, koodin moduulien rajojen asettaminen helpottuu. Toimintojen kirjoittaminen omiin moduuleihin tekee koodista uudelleen käytettävän, ja se helpottaa virheiden etsimissä. Yhteen moduuliin on järkevää kirjoittaa yksi itsenäisesti toimiva toiminto. Koodi kannattaa kirjoittaa moduulin sisälle niin, että moduulissa on tiedonsyöttökohdat ja omat tulostuskohdat. Tämä mahdollistaa koodin uudelleen käyttämisen ja saman moduulin nopean asettamisen uuteen makroon. (29.)

Kuvassa 4 näkyvät kaikki moduulit ja käyttäjälomake, joita on tähän työhön liittyvässä BOX 3.0 -makrossa.



KUVA 4. BOX 3.0 -makron moduulit

Kuvassa 4 näkyvissä moduulien nimissä on aakkonen edessä, jotta moduulien ajojärjestys olisi selkeä. Taulukossa 4 on BOX 3.0 -makron uudelleen käytettävät moduulit ja niiden tehtävät. Nämä moduulit tallennettiin erillisinä bas-tiedostoina, jotta niitä voitaisiin käyttää myöhemmin muissa makroissa.

TAULUKKO 4. BOX 3.0 -makroon tehdyt uudelleen käytettävät moduulit. Moduulien nimiin on pistetty X niihin kohtiin, jotka ovat riippuvaisia makrosta.

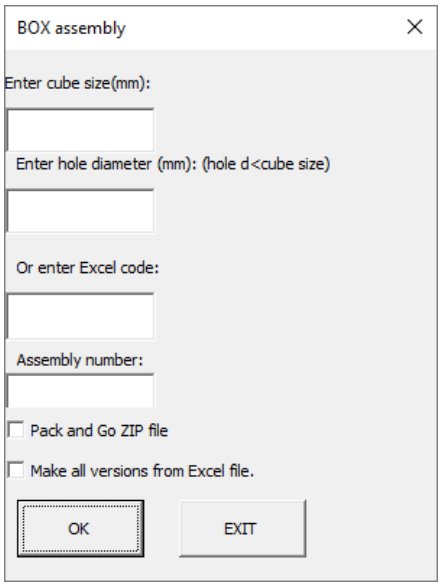
Moduuli	Moduulin tehtävä
XaddCompX	Lisää aktiiviseen kokoonpanoon halutut osa tiedostot.
XgetXXXXFX	Hakee tiedostosta nimetyn pinnan.
XAddMateX	Lisää mate-liitännän kokoonpano tiedostoon.
XEditPropX	Muokkaa tiedoston ominaisuus (properties) tietoja.
Xpackandgo	Moduuli tekee aktiivisesta tiedostosta kokoonpanon.

Moduuleja voidaan lisätä makroon Solidworksin API-sovelluksessa työkalurivin *Insert*-painikkeen alta. Tästä aukeaa valintaikkuna, josta voidaan navigoida haluttuun bas-tiedostoon. Moduuliin tulisi lisätä sellainen nimi, joka ilmaisee minkä, toiminnon moduuli ratkaisee (29). Kun moduuli on lisätty, kannattaa vaihtaa nimi sellaiseksi, josta selviää moduulin sisältö. Kun moduulin koodi on kirjoitettu, kyseinen moduuli voidaan tallentaa erillisenä tiedostona käyttämällä APIssa olevaa *Export File*. -toimintoa. Toiminto tallentaa moduulin erillisenä bas-tiedostona, joka voidaan tuoda toiseen makroon *Import File* -toiminolla. Nämä toiminnot löytyvät Project Explorer -ikkunassa painamalla oikeaa hiiren näppäintä.

3.7 UserForm

UserForm eli käyttäjälomake on yksi moduulityyppi, jota voidaan käyttää VBA-makroissa. UserForm toimii yleisesti makron käyttöliittymänä. Käyttäjälomakkeita hyödyntämällä on helppo tehdä makro, jossa on monta ajopolkua, joiden suorittaminen riippuu käyttäjän syöttämistä arvoista. UserFrom-moduulin voi tehdä makron kirjoittajan haluaman näköiseksi sille tarkoitetuilla työkaluilla.

UserForm-moduulien toiminta eroaa muiden moduulien toiminnasta siten, että lisättäessä moduuliin uusi osa API lisää kyseiselle osalle uuden proseduurin UserForm-moduuliin koodiin. Kuvassa 5 näkyy työhön liittyvän BOX 3.0 -makron käyttäjälomake. (30.)



KUVA 5. BOX 3.0 -makron UserForm-moduuli

3.8 Makrojen toiminta eri kokoonpanoissa

Makron toimivuus eri tietokonekokoonpanoissa riippuu monesta tekijästä. Kaikki SolidWorksin 2013 ja sitä uudemmat API versiot ovat 64-bittisiä versioita. 64-bittisellä API-sovelluksella ei voi suorittaa makroja, jotka on tehty 32-bittisellä APIn versiolla. Jotta 64-bittisellä API-sovelluksella voidaan ajaa 32-bittisellä API-sovelluksella tehtyä koodia, kaikkien muuttujien deklaraatiot täytyy nimetä PtrSafe-attribuutilla. Tämä attribuutti sallii muuttujan käytön 32- ja 64-bittisessä järjestelmässä. (31.)

Makrojen yleisin suoritusvirhe tulee referenssivirheestä. Virhe ilmenee usein ajettaessa makroa eri koneella kuin millä se tehtiin. Seuraavana on esimerkki, millaisilla järjestelmän kokoonpanoilla makron suoritusvirheet yleensä ilmenevät. Taulukossa 5 on kerrottu esimerkin tietokoneiden järjestelmien kokoonpanot.

TAULUKKO 5. Makron suoritusvirhe esimerkin tietokone järjestelmien kokoonpanot

Tietokone 1 järjestelmän kokoonpano	Tietokone 2 järjestelmän kokoonpano

• SolidWorks 2015
• Microsoft Office 2013
  ○ *Microsoft Excel 15.0 Object library.*
	• SolidWorks 2012
	• Microsoft Office 2010
	  ○ *Microsoft Excel 14.0 Object library.*

Taulukon 5 esimerkissä makrot eivät voi toimia mitenkään toistensa kanssa, jos jokaisessa muuttujassa ei ole PtrSafe-attribuuttia. Jos tietokoneella 2 tehtyä makroa, joka käyttää Exceliä, koetetaan suorittaa tietokoneessa 1, makro antaa virheilmoituksen *Run-time error '438'*. Tämä virhe tulee siitä, että makro yrittää käyttää Excel 2010:n tukemaa Microsoft Excel 14.0 Object library -referenssiä. Tämän virhe korjataan menemällä makron referensseihin, josta pitää poistaa Microsoft Excel 14.0 Object library -referenssi, ja sen tilalle tulee asettaa Microsoft Excel 15.0 Object library -referenssi. API ilmoittaa kaikki virheelliset referenssit MISSING-etuliitteellä.

4 AVAINSANAT, KONDITIONAALIT JA JATKUMOT

SolidWorks APIssa on monia avainsanoja, jotka eivät liity SolidWorksiin millään lailla. Nämä sanat näkyvät koodissa sinisellä. Näitä avainsanoja ei voi käyttää koodiin nimissä, ellei sanaa pistä hakasulkeisiin (32). Taulukossa 6 näkyy työn makroihin liittyviä avainsanoja ja operaattoreita, jotka eivät ole yksiselitteisiä tai suoria käännöksiä englannin kielestä.

TAULUKKO 6. Esimerkki avainsanoista, jotka esiintyvät koodissa

Avainsana	Määritelmä
<>	Operaattori, joka testaa onko X erisuuruinen kuin Y (33).
Select	Valitsee yhden määritellyn Case -komentokokoelman (34).
Case	Määrittelee komentorivi kokoelman (34).
&	Operaattori yhdistää kaksi eri ilmaisua yhdeksi ilmaisuksi (35).

Jotta voidaan käsitellä VBA-ohjelmointikieliä, täytyy aluksi käsitellä konditionaaleja. Tässä luvussa käydään läpi vain ne jatkumolohkot ja konditionaalit, joita käytetiin tähän työhön liittyvissä makroissa. Jatkumotyyppejä on monia, joista ohjelmoijat käyttävät niitä, joita ovat tottuneet käyttämään.

4.1 If…Then…Else-konditionaali

If-konditionaalilla voidaan tehdä makroon haarautuvia ajopolkuja. If…Then-konditionaali ehdoista riippuen suorittaa eroavan koodipolun. Koodissa if-sanan jälkeen löytyy jokin ehtolause. Jos tämä ehtolause on tosi, suoritetaan komennot, jotka tulevat Then-sanan jälkeen. Suorittamisen jälkeen poistutaan If…Then…Else-lohkosta. Jos if-sanan jälkeinen ehtolause ei ole tosi suoritetaan komennot, jotka löytyvät Else-sanan jälkeen, ja poistutaan If…Then…Else-lohkosta. (36.)

If…Then…Else-lohkoon voidaan lisätä ehtoja käyttämällä And- tai Or-avainsanoja. And-sanalla voidaan asettaa ehtoja, joiden on oltava tosia, jotta ne voitaisiin suorittaa Then-sanan jälkeiset komennot. Or-sanalla voidaan tehdä ehtosarja, josta tarvitsee vain yhden olla tosi, jotta voitaisiin suorittaa Then-sanan jälkeiset komennot. Seuraavana on esimerkkimakro, jossa esitellään, miten If…Then…Else-konditionaali toimii. (36.)

```
1   Dim intNumero1 As Integer
2   Dim intNumero2 As Integer

3   Sub main()
4       intNumero1 = 5
5       intNumero2 = 30
6       If intNumero1 = intNumero2 Or intNumero2 < 30 Then
7           MsgBox "Väite 1 on tosi."
8       Else
9           MsgBox "Väite 2 on tosi."
10      End If
11  End Sub
```

Yllä olevan esimerkin koodi testaa, onko väite 1 vai väite 2 tosi. Esimerkin koodiriveillä 1 ja 2 nimitetään muuttujat 1 ja 2 integer-datatyypiksi eli kokonaislu-

vuiksi. Proseduurin riveillä 4 ja 5 asetetaan integer-muuttujille numeroarvot. Rivillä 6 testataan ovatko numeromuuttujat samankokoisia. Tämä on ehto 1. Rivillä 6 or-sanan jälkeen testataan, onko numeron 2 arvo pienempi kuin 30. Tämä on ehto 2. Riippuen siitä kumpi väite on tosi, makro tulostaa käyttäjälle viestin tekstiruutu-ikkunassa. Ehto 1 on tosi, kun numeromuuttujat 1 ja 2 ovat samankokoisia. Ehto 2 on tosi, kun *intNumero2*-muuttuja on pienempi kuin 30. Rivillä 7 oleva väite tulostetaan, jos ehto 1 tai ehto 2 on tosi. Jos kumpikaan ehdoista ei ole tosi, tulostetaan rivillä 9 oleva väite.

4.2 While…Wend-jatkumo

While…Wend-jatkumo on kaikista yleisin jatkumolohko, kun halutaan toistaa koodi useaan kertaan. Tähän työhön liittyvässä Human model scaler -makrossa käytetiin tätä lohkoa (luku 7.3). While…Wend-lohko suorittaa komentosarjaa niin kauan, kuin määritelty ehto on tosi. Lohkoon voidaan lisätä useampia ehtoja käyttämällä avainsanoja. Lohko alkaa While-sanasta ja loppuu Wend-sanaan. Seuraavana on yksinkertainen laskuri esimerkki While…Wend-lohkosta. (37.)

```
1    Dim i As Integer

2    Sub main()
3        i = 0
4        While i <> 10
5            i = i + 1
6        Wend
7    End Sub
```

Esimerkissä alussa nimetään i-muuttuja integer-datatyypiksi ja sille asetetaan arvoksi 0. Yksittäisiä kirjaimia käytetään usein muuttujina, jos niillä ei ole muuta tarkoitusta kuin toimia laskurina. Lohkossa testataan, onko i-muuttujan arvo 10.

Jos i-muuttujan arvo ei ole 10, suoritetaan komennot While- ja Wend-avainsanojeni välissä. Rivillä 5 i-muuttujan arvoa kasvatetaan yhdellä. Esimerkin lohko suoritetaan 11 kertaa, jonka jälkeen makron suoritus loppuu.

4.3 For…Next-lohko

For…Next-lohkolla voidaan toistaa komentorivit niin monta kertaa kuin lohkoon määritetään. For…Next-lohko on harvinaisempi kuin While…Wend-lohko, sillä While…Wend-lohkoa on joustavampi käyttää. While…Wend-lohkossa ei tarvitse tietää kuinka monta kertaa koodi pitää suorittaa. For…Next-lohkosta voidaan poistua ennen sen loppuun ajamista käyttämällä Exit For -avainsanoja. (38.)

```
1       Dim i As Integer

2       Sub main
3       For i = 0 To 5
4           If i = 3 Then Exit For
5       Next i
6       End Sub
```

Esimerkki koodin alussa nimitetään i-muuttuja integer-datatyypiksi. Rivillä 3 ilmaistaan kuinka monta kertaa halutaan For…Next-lohko ajettavan ja samalla määritetään i-muuttujan lähtöarvo. Tässä tapauksessa lohkoa suoritettaisiin 6 kertaa, mutta se kuitenkin ajetaan vain neljästi, sillä For…Next-lohkon sisälle on If…Then -konditionaali, jonka lopputuloksena koodin ajo loppuu, kun i-muuttujan arvo on 3. Jos i-muuttujan arvo on eri kuin 3 tai 5, i-muuttujan arvoa kasvatetaan aina yhdellä lähtien nollasta.

5 MAKROT PARAMETRISESSÄ MALLINNUKSESSA

Parametrisellä mallinnuksella tarkoitetaan mallintamista, joka on riippuvainen muuttujista. Toisin sanoen parametrisessä mallinnuksessa mallin ominaisuuksia käsitellään muuttujina, kuten kappaleen paksuus. SolidWorksin VBA-makroilla pystytään käytännössä tekemään kaikki mahdolliset toimenpiteet, mitä voidaan tehdä normaalisti SolidWorksillä. Tässä työssä ei voida kokeilla kaikkia SolidWorksin-toimintojen ohjaamista VBA-makroilla, koska kokeilun työmäärä muodostuisi liian suureksi.

Ennen makron kirjoittamisen aloittamista pitää miettiä, mitä askeleita joudutaan tekemään silloin, kun makron tehtävä tehdään normaalisti SolidWorksissä. Näihin askeleisiin kuuluu jokainen hiiren ja näppäimistön painallus, joita joudutaan tekemään SolidWorksillä mallintaessa.

Esimerkkinä voidaan käyttää 3D-malligeneraattoria, joka tulostaa erikokoisia muttereita. Tässä esimerkissä tulee miettiä kaikista lyhyin reitti erikokoisien mallien luomiselle. Lyhyin reitti tässä tapauksessa on tehdä yksi juurimalli, josta koodilla muokataan erikokoisia malleja. Tähän generaattoriin voidaan tehdä käyttöliittymä, josta makron käyttäjä valitsee kulloinkin haluamansa mutterityypin. Kaikki käyttäjän haluamat mutteriin tarvittavat mitat voidaan viedä Excel-tiedostoon, josta ne voidaan hakea makron käyttäjän antamien tietojen perusteella. Esimerkin generaattori voisi toimia seuraavasti:

1. Käyttöliittymä avautuu.
2. Käyttäjä syöttää haluamansa mutterin tiedot (esim. M10).
3. Makro hakee Excelistä halutun mutterin mitat.
4. Avataan juurimalli
5. Muokataan juurimallin mittoja.
6. Tallennetaan haluttu malli eri nimellä.

Makrolla ei kannata tehdä mitään muuta kuin on pakko tehdä, jotta makron tarkoitus saavutettaisiin. Makron kirjoittajan tulee miettiä, mitkä arvot muuttuvat

mallia tehtäessä. Makroon ei kannatta lisätä toimintoja, jotka tarvitsee tehdä vain kerran, sillä se tekee makron tekemisestä työlään ja mahdollisesti epävakaan.

5.1 Objektien valinta ja mallinnus

Jotta SolidWorksin API-sovelluksella voidaan käsitellä objektia, objekti täytyy ensin valita. Tähän on olemassa monta menetelmää, mutta tärkeimmät näistä menetelmistä ovat SelectById2- ja Select4-komennot (liite 1/3 ja 1/8). SelectById2-menetelmä on näistä huonompi, koska se on paljon epävarmempi toiminnaltaan. SelectById2-komentoa käyttäessä pitää käyttää sen objektin nimeä, joka on tarkoitus valita. Kokoonpanossa saa olla vai yksi sen niminen objekti, mitä ollaan valitsemassa, jos aiotaan käyttää SelectById2-komentoa. SelectById2-menetelmä on nopein menetelmä valita objekteja SolidWorksin historiapuusta. Select4-menetelmällä voidaan valita objekteja, jotka on tallennettu valmiiksi toiseen muuttujaan. Komento, jolla data tallennetaan muuttujaan, riippuu objekti-tyypistä. Tästä löytyy esimerkki luvusta 6.4. (39.)

SolidWorksissä jokaisella piirteellä (featurella) ja luonnostelmalla (sketch) on automaattisesti oma nimi. Kaikki automaattisesti asetetut nimet ovat muokattavia. Objektien valinta nimen perusteella on kaikista varmin menetelmä VBA-makroissa. Tätä menetelmää käytetiin BOX 3.0 -makron malleissa. Nimen perusteella objektin valitseminen voi olla työläs menetelmä, koska kappaleen 3D-mallissa pitää olla nimettynä ne pinnat, joita makrossa aiotaan käyttää. Jos tätä nimeämistä ei ole tehty valmiiksi, koodin kirjoittamisen vaiheessa kannattaa miettiä muita objektin valitsemisen menetelmiä. (39.)

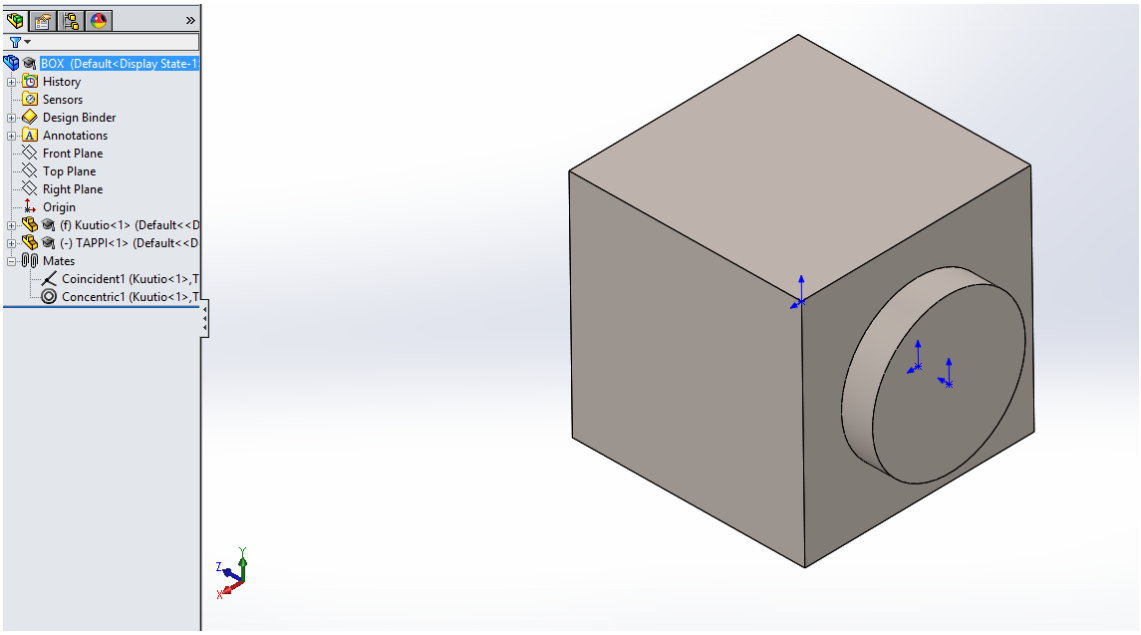
5.2 Asetukset

Jotta makroilla voidaan objekteja käsitellä ilman suoritusvirheitä, joitain asetuksia täytyy muuttaa. Jos makrolla halutaan valita tiedostoja joistain kansiosta, tiedostojen tiedostopäätteet tulisi laittaa näkyväksi. Microsoftin Windows 10 -käyttöjärjestelmässä tiedostopäätteet saa näkyväksi menemällä kansiossa työkalurivin näytä-valikkoon, ja valitsemalla tiedostotunnisteet päälle.

Kun SolidWorksissä makrolla muutetaan sketsin mittaa, koodilla tulisi sulkea *Input Dimension Value* pois päältä. Tämä estää mitan syöttöikkunan esille ponnahtamisen, mikä pysäyttää makron suorittamisen (40). Muitakin asetuksia voidaan koodilla joutua muuttamaan virheettömän suorittamisen takia. Makron lopussa tulisi muistaa palauttaa muutetut asetukset samanlaisiksi kuin ne olivat. Tämä tekee makrosta käyttäjäystävällisemmän.

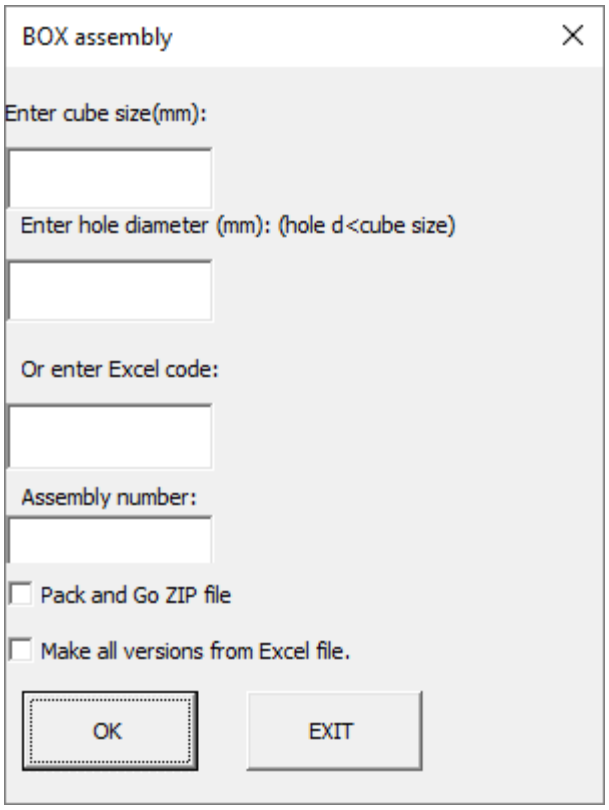
6 BOX 3.0 -MAKRO

Työssä tehtiin aluksi harjoitusmakro, jolla tutkittiin, miten VBA-makroja tehdään. Tämä harjoittelumakro on nimeltään BOX 3.0. Makron pääasiallinen tarkoitus oli selvittää, miten makroja tehdään, ja mitä menetelmiä tulisi käyttää parametrista mallinnusta ajatellen. Makrosta tehtiin mahdollisimman monessa kokoonpanossa toimiva ja helppokäyttöinen. Makron pituus on 823 koodiriviä. Kuvassa 6 on esitetty makron BOX 3.0 suorittamisen lopputulos.



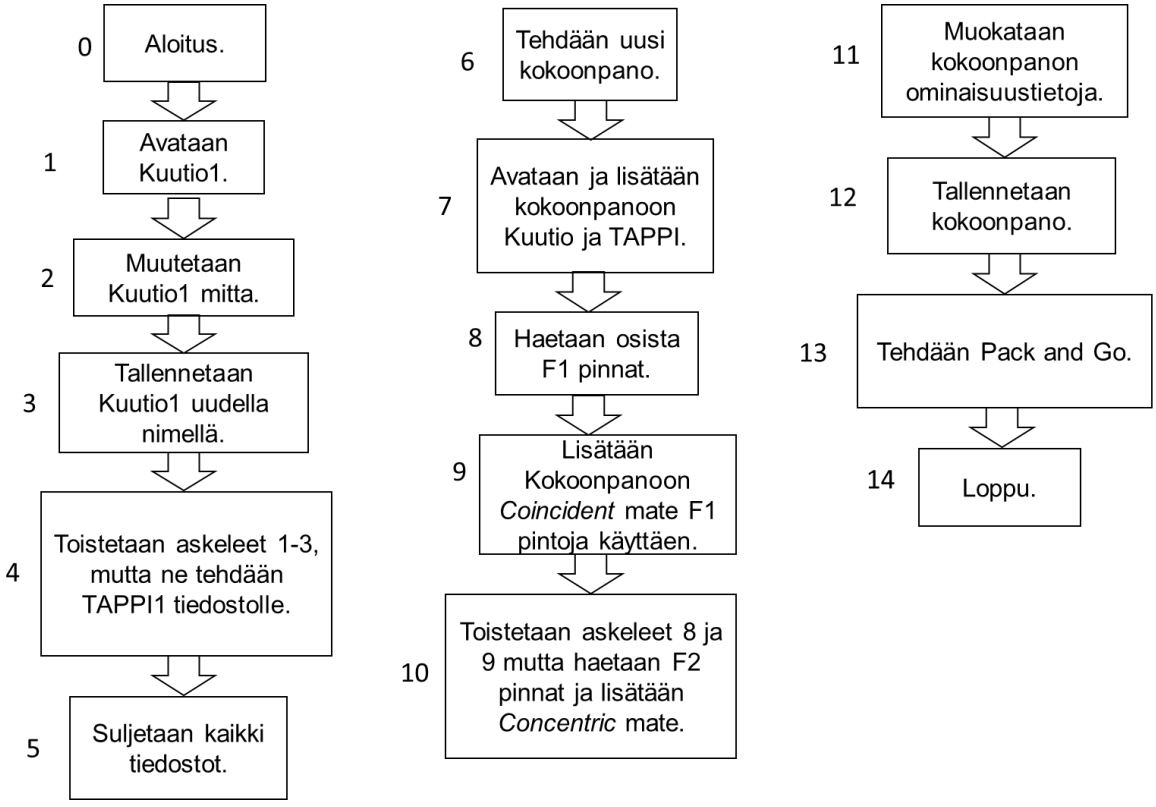
KUVA 6. BOX 3.0 -makron lopputulos kappale

BOX 3.0 -makro muuttaa juuriosien Kuutio- ja TAPPI-mitat halutuksi, ja tekee niistä kokoonpanon. Makrolla voidaan tulostaa malleja suoraan Excel-tiedostoon asetettujen mittojen perusteella. Kuvassa 7 on BOX 3.0 -makron käyttöliittymä.



KUVA 7. BOX 3.0 -makron käyttöliittymä

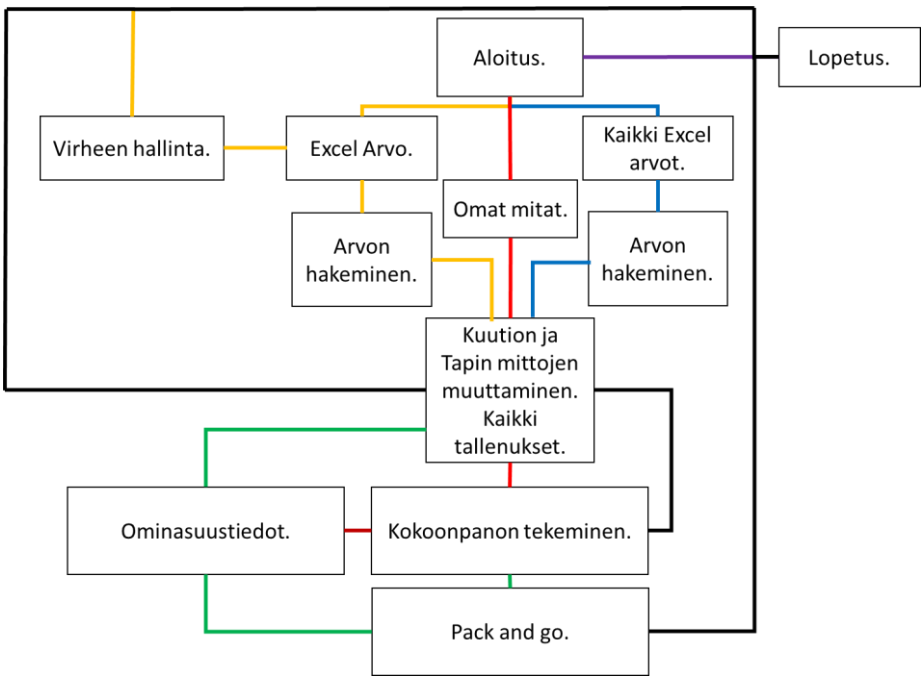
Kuvassa 8 on kaavio, mitä makro BOX 3.0 tekee SolidWorksissä ja missä järjestyksessä. Askel 0 tapahtuu M-moduulissa. Askeleet 1-5,12 ja 14 tapahtuvat BOX 3.0 -makron AA UserForm -moduulissa. Askel 6 tapahtuu A-moduulissa. Askel 7 tapahtuu B-moduulissa. Askel 8 on C-ja D-moduuleissa. Askel 9 on E-moduulissa. Askel 10 on moduuleissa-F-H. Askel 11 on I-moduulissa. Askel 13 on K-moduulissa. L-moduuli suoritetaan vain silloin, kun makron käyttöliittymän mukautettu mitta -kohtiin syötetään jotain muuta kuin numeroita. M-moduulissa ei ole muuta kuin *AAfrmDimCont.Show* rivi. Tämä rivi tuo makron käyttöliittymän esille. M-moduuli on pakko olla makrossa, sillä *Play macro* -toimintoa käytettäessä SolidWorksissä, API suorittaa viimeisimmän lisätyn moduulin makrosta. Kuvassa 4 näkyvät kaikki BOX 3.0 -makrossa olevat moduulit.



KUVA 8. BOX 3.0 -makron operaatiot

6.1 Makron suorittamisen polut

Makrossa BOX 3.0 on monta mahdollista eri suorituspolkua. Kuvassa 9 on kaavio, siitä millaisia polkuja makro voi suorittaa. Kuvassa 9 on eri väreillä merkitty mahdolliset suorituspolut.



KUVA 9.10 BOX 3.0 -makron mahdolliset suorittamisen polut

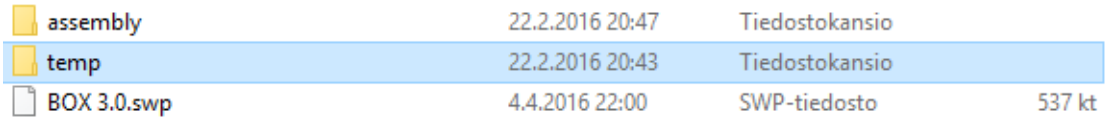
Taulukossa 7 on selitetty, minkä polun makro suorittaa, kun BOX 3.0 -makron käyttöliittymän (kuva 7) valikoita painetaan.

TAULUKKO 7. BOX 3.0 –makron suorituspolkujen värien selvitys

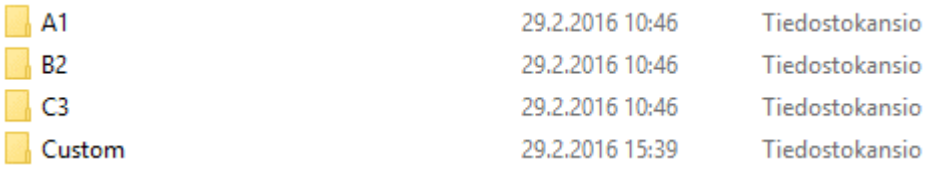
Väri	Polku
Violetti	Kun valitaan *EXIT* käyttöliittymästä, tällöin suoritetaan violettia viivaa myöten. *EXIT*-vaihtoehto sulkee makron.
Punainen	Punaista viivaa suoritetaan silloin, kun käyttäjä käyttää mukautettuja mittoja makrossa.
Keltainen	Keltaista viivaa suoritetaan, kun käyttäjä kirjoittaa jotain *Excel code* -ikkunaan.
Sininen	Sinistä viivaa suoritetaan käyttäjän valitessa käyttöliittymästä *Make all versions from Excel file* -toiminnon. Toiminto tekee kaikki versiot Excel -tiedostosta.
Vihreä	Vihreää viivaa suoritetaan silloin, kun käyttäjä valitsee Pack and Go -vaihtoehdon käyttöliittymästä. Toiminto voidaan suorittaa kaikissa poluissa.
Tumman punainen	Tumman punaista viivaa suoritetaan käyttäjän kirjoittaessa käyttöliittymään *Assembly number* -kohtaan jotain. Toiminto lisää mallin ominaisuustietoihin kirjatun numeron. Tämä voidaan tehdä kaikissa poluissa.
Musta	Mustalla merkityt polut riippuvat siitä, mitä vaihtoehtoja on valittu. Kaikki mustat viivat menevät lopetukseen.

6.2 Kansiorakenne ja toiminta BOX 3.0 -makrossa

Kaikki tiedostot laitettiin yhden kansiopolkurakenteen alle, jotta makro toimisi mistä vain kansiosta. Kuvassa 10 näkyy millaisessa kansiorakenteessa tiedostot tulee olla BOX 3.0 -makrossa. Kuvassa 11 näkyy *assembly*-kansion sisältö.



KUVA 110. BOX 3.0 -makro kansion sisältö



KUVA 121. BOX 3.0 -makron assembly-kansion sisältö

Temp-kansiossa on kaikki pohjatiedostot, joita makro käyttää oikeiden mallien tekemiseen. Tässä kansiossa on myös BOX 3.0 -makron Excel-taulukko. Assembly-kansioon tallentuu makron tekemät mallit oikeaan kansioon. Custom -kansioon tallentuu mallit, jos käyttäjä on käyttänyt mukautettuja mittoja. Makro tehtiin sellaiseksi, että se toimii mistä tahansa kansiosta. Tämä ominaisuus makroon saadaan aikaan seuraavilla koodiriveillä.

```
Set swApp = swApp.ActiveDoc
strMacroPath = swApp.GetCurrentMacroPathFolder
```

Näillä riveillä tallennetaan *strMacroPath* -muuttujaan sen kansion polku, missä makro on. Seuraavana on esimerkki siitä, miten tätä muuttujaa käytetiin makrossa. Kyseinen esimerkki tallentaa *strTAPPIPath*-muuttujaan TAPPI-osan polun.

```
strTAPPIPath = (strMacroPath + "\assembly\Custom\TAPPI.SLDPRT")
```

6.3 Excel-toiminta

BOX 3.0 -makrossa on mahdollisuus tehdä BOX-kokoonpano Excel-taulukoon lisättyjen mittojen perusteella. Kyseisen taulukon nimi on BOX3.0 ja se löytyy temp-kansiosta. Alla olevassa esimerkki koodissa haetaan kyseisestä Excel -taulukosta tiedetyistä soluista haluttu arvo. Taulukossa 8 on tarkat selitykset siitä, mitä koodirivit tekevät.

```
1   Dim xlApp As Excel.Application
2   Dim xlWb As Excel.Workbook

3   Set xlApp = CreateObject("Excel.Application")
4   xlApp.Visible = False
5   strExcelPath = (strMacroPath + "\temp\BOX3.0.xlsx")
6   Set xlWb = xlApp.Workbooks.Open(strExcelPath)
7   intRow = 2
8       While Cells(intRow, 1).Value <> ""
9       vExcelCubeSize = Cells(intRow, 2).Value
10      vExcelHoleSize = Cells(intRow, 3).Value
11      vExcelName = Cells(intRow, 1).Value
12  intRow = intRow + 1
```

TAULUKKO 8. Excel-koodin selitykset

Numero	Selitys
1	Deklaraatiolla tehdään early binding xlApp-muuttujan ja Microsoft Excel 15.0 Object Library –referenssin välille.
2	Deklaraatiolla xlWb tehdään early binding Excel-taulukoihin liittyviin toimintoihin.
3	xlApp-muuttujaan asetetaan tietokoneessa oleva Excelin oletusarvo. Komennon tilalla voisi olla *Set xlApp= Application.Excel*. Jos tätä riviä käytetäisiin, Excel pitäisi olla avattuna.
4	Rivi asettaa Excel-tiedoston avattavaksi vain tietokoneen muistissa eikä näytöllä.
5	strExcelPath-muuttujaan tallennetaan Excel-tiedoston polku.
6	Rivillä asetetaan xlWb-muuttujaan strExcelPath-muuttujaan tallennettu tiedosto.
7	intRow-muuttujan arvoksi asetus 2. Arvoksi asetetaan 2, koska Excelissä rivillä 1 on otsikoita. Muuttujaa käytetään Excelistä oikean rivin valintaan.
8	Rivillä tarkastetaan onko aktiivisessa Excelin solussa mitään arvoa. Kahdella ”-merkillä voidaan ilmaista onko merkkien määrä nolla. Esimerkissä aloitetaan *While*-lohko, koska se on kohdasta koodia, jolla tulostetaan koko Excel-taulukon sisältö. Lohkosta poistutaan, kun Excelistä haetun solun arvo on 0 merkkiä pitkä.
9	vExcelCubeSize-muuttujaan tallennetaan Excelin solun arvo. Ensimmäisellä kierroksella valittavan solun koordinaatit on rivi 2 ja sarake 2. Muuttujaa käytetään Kuutio-mallin mitan muuttamiseen.
10	vExcelHoleSize-muuttujaan tallennetaan Excelin solun arvo. Ensimmäisellä kierroksella valittavan solun koordinaatit on rivi 2 ja sarake 3. Muuttujaa käytetään TAPPI-mallin mitan muuttamiseen.
11	vExcelName-muuttujaan tallennetaan Excelin solun arvo. Ensimmäisellä kierroksella valittavan solun koordinaatit on rivi 2 ja sarake 1. Muuttujaa käytetään kansion tekemiseen osien tallennusta varten.

BOX 3.0 -makro rakentuu siten, että makron Excel-taulukkoon voidaan lisätä mittoja, joiden perusteella makro tekee uusia malleja. Jotta tämä uusi rivi tekisi oikeanlaisen mallin, mallikoodi täytyy koostua yhdestä kirjaimesta ja numeroista. Esimerkiksi A1 on toimiva mallikoodi. Näitä koodeja voidaan tehdä 1-99

väliltä. Makroa ajaessa tulee suoritusvirhe, jos *assembly*-kansiossa on saman niminen koodilla nimetty kansio. Kun makrolla tehdään kokoonpano tiedetyllä mallikoodilla, mallikoodin numeroa käytetään oikean Excel-solun valintaan.

6.4 Oikean pinnan valinta

Esimerkkinä makron BOX 3.0 -koodista käsitellään moduulia, joka valitsee kummastakin osasta nimetyt pinnat. Kappaleet liitetään yhteen mate-komennolla, jota varten koodilla täytyy valita halutut pinnat. Alla olevassa esimerkkikoodissa valitaan Kuution KF1-pinta, ja tallennetaan se *swKuutioF1*-muuttujaan. Taulukossa 9 on tarkat selitykset siitä, mitä jokainen rivi tekee. Jos makro ei löydä kaikkien pintojen testaamisen jälkeen pintaa nimeltä KF1, makro menee komentoon *End Sub*, jolloin makron ajo loppuu.

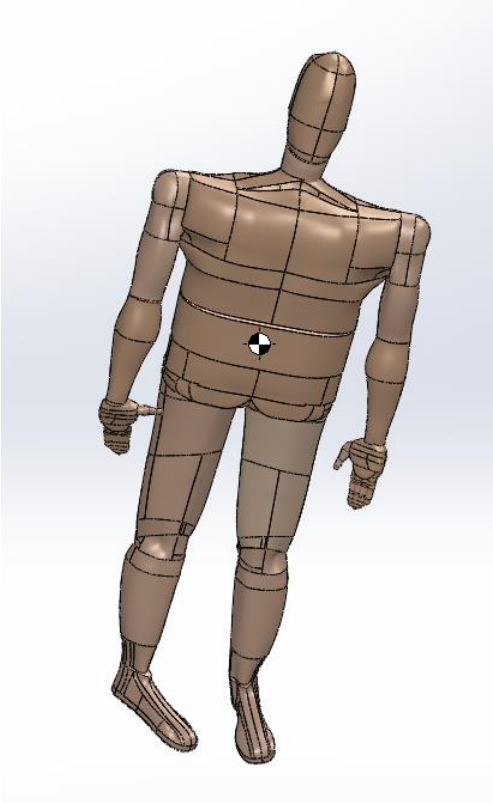
```
1   Sub GetKuutioFace1()
2   Set swApp = Application.SldWorks
3   Set swAssy = swApp.ActiveDoc
4       Set swModel = swAssy
5       vComps = swAssy.GetComponents(False)
6       For i = 0 To UBound(vComps)
7           Set swComp = vComps(i)
8           vBodies = swComp.GetBodies3(-1, Empty)
9           Set swBody = vBodies(0)
10          vFaceF1 = swBody.GetFaces
11          For j = 0 To UBound(vFaceF1)
12              Set swFace = vFaceF1(j)
13              Set swKuutioF1 = swFace
14              If swModel.GetEntityName(swKuutioF1) = "KF1" Then Exit Sub
15              Next j
16      Next i
17  End Sub
```

TAULUKKO 9. DgetKuutioF1-moduulin sisältämien koodirivien selvitykset

Numero	Selitys
1	Proseduurin nimeäminen, joka on GetKuutioFace1.
2	swApp-muuttujaan asetetaan tällä hetkellä käynnissä oleva SolidWorks.
3	swAssy-muuttujaan asetetaan tällä hetkellä oleva aktiivinen dokumentti.
4	swModel-muuttuja asetetaan samaksi swAssy-muuttujan kanssa.
5	vComps-muuttujaan tallennetaan kaikki kokoonpanon komponentit. Jos komennon lopussa olisi true, tällöin vComps-muuttujaan tallennettaisiin kaikki komponentit ja kaikki alikokoonpanon komponentit.
6	Rivi valitsee yhden komponentin niistä, joita on tallennettu vComps-muuttujaan.
7	Rivi asettaa valitun komponentin swComp-muuttujaan.
8	vBodies-muuttujaan tallennetaan 1 body, joita swComp-muuttujaan on tallennettu.
9	Asetetaan swBody-muuttujaan 1 pursotus vBodies-muuttujasta. Samalla tyhjennetään vBodies-muuttuja.
10	vFaceF1-muuttujaan tallennetaan kaikki pinnat swBody-muuttujasta.
11	Rivi valitsee yhden pinnan vFaceF1-muuttujasta.
12	Rivi asettaa swFace-muuttujaan valitun pinnan.
13	Asettaa swKuutioF1-muuttujaan valitun swFace-muuttujaan tallennetun pinnan.
14	Jos swKuutioF1-muuttujan nimi on KF1, tällöin makron ajo poistuu For…To-lohkosta.
15	Jos rivillä 14 ei lohkosta poistuttu, makron ajo siirtyy riville 11. Rivillä kasvatetaan j-muuttujan arvoa yhdellä.
16	Jos For…Next-j-lohkosta ei löytynyt pintaa nimeltä KF1 kaikkien pintojen testaamisen jälkeen, valitaan seuraava body. Rivi kasvattaa i-muuttujan arvoa yhdellä.
17	Proseduurin lopetus.

7 HUMAN MODEL SCALER -MAKRO

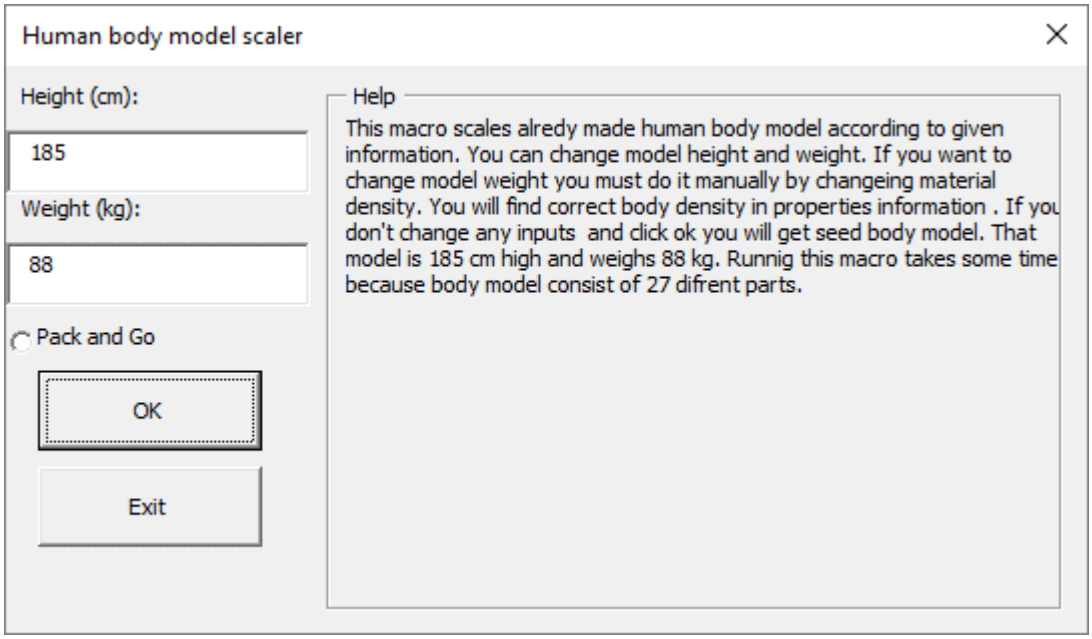
Työhön tehtävän pilottimakron aiheeksi ideoitiin makro, jolla voidaan tehdä ihmismalli annettujen parametrien perusteella. Ihmismallista pyrittiin tekemään parempi malli, mitä on tarjolla ilmaiseksi internetissä. Tehty ihmismalli näkyy kuvassa 12, ja makroon tehty käyttöliittymä näkyy kuvassa 13. Human model scaler -makron pituus on 411 koodiriviä.



KUVA 132. Human model scaler -makron ihmismalli

Pilottiin ihmismalli tehtiin itse. Malli koostuu 27 osasta. Malliin tehtiin lähes kaikki ihmiskehosta löytyvät nivelet luonnollisilla liikeradoilla. Malliin ei tehty jokaista sormea erikseen, koska sormien mallintaminen olisi ollut liian työlästä, ja tehnyt mallista raskaan. Käteen tehtiin peukalo kahdella erillisellä nivelellä. Malliin ei tehty varpaita, koska pilotin tarkoitus oli tehdä skaalattava ihmismalli -makro eikä täydellistä ihmismallia. Mallin nivelet tehtiin pallo- ja sarananivelillä.

Sarananiveliä käytetiin niissä paikoissa, joissa taittumiskulma on vain yhden akselin suuntaista. Pohjamallin korkeus on 185 cm ja paino on 88 kg.

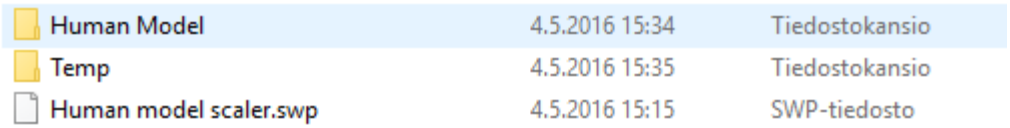


KUVA 143. Human model scaler -makron käyttöliittymä

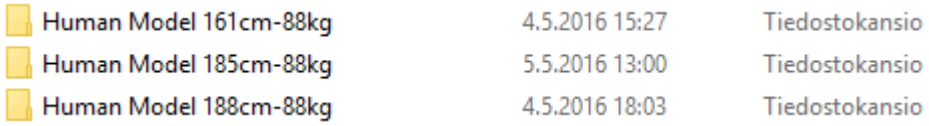
Makroon tehtiin oma materiaali. Materiaalin nimeksi annettiin Body Material ja sille asetetiin oma väri ja tiheys. Body Materialin tiheydeksi asetetiin 700 kg/m^3. Tämä tiheys saadaan laskemalla pohjamallille sellainen tiheys, jolla saataisiin pohjamallin painoksi 88 kg.

7.1 Human model scaler -makron kansiorakenne

Human model scaler -makron voi ajaa mistä vain kansiosta silloin, kun Human model– ja Temp-kansio ovat samassa kansiossa, jossa makrotiedosto on. Tämä on sama toiminto, mikä on BOX 3.0 –makrossa. Kuvassa 14 näkyy oikea kansiorakenne. Kuvassa 15 vuorostaan näkyy Human model -kansion sisältö. Kuvassa 15 näkyvät kansiot luodaan makrolla käyttöliittymään annettujen tietojen perusteella.



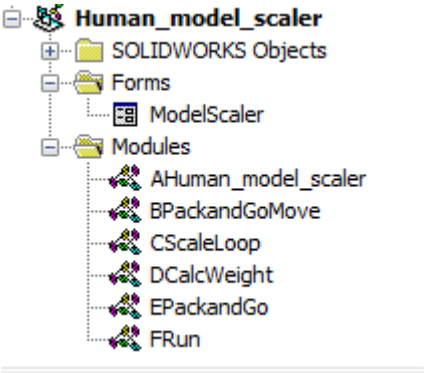
KUVA 154. Human model scaler -makron kansio rakenne



KUVA 165. Human model -kansio. Kuvassa olevat kansiot on luotu Human model scaler-makroa suorittamalla.

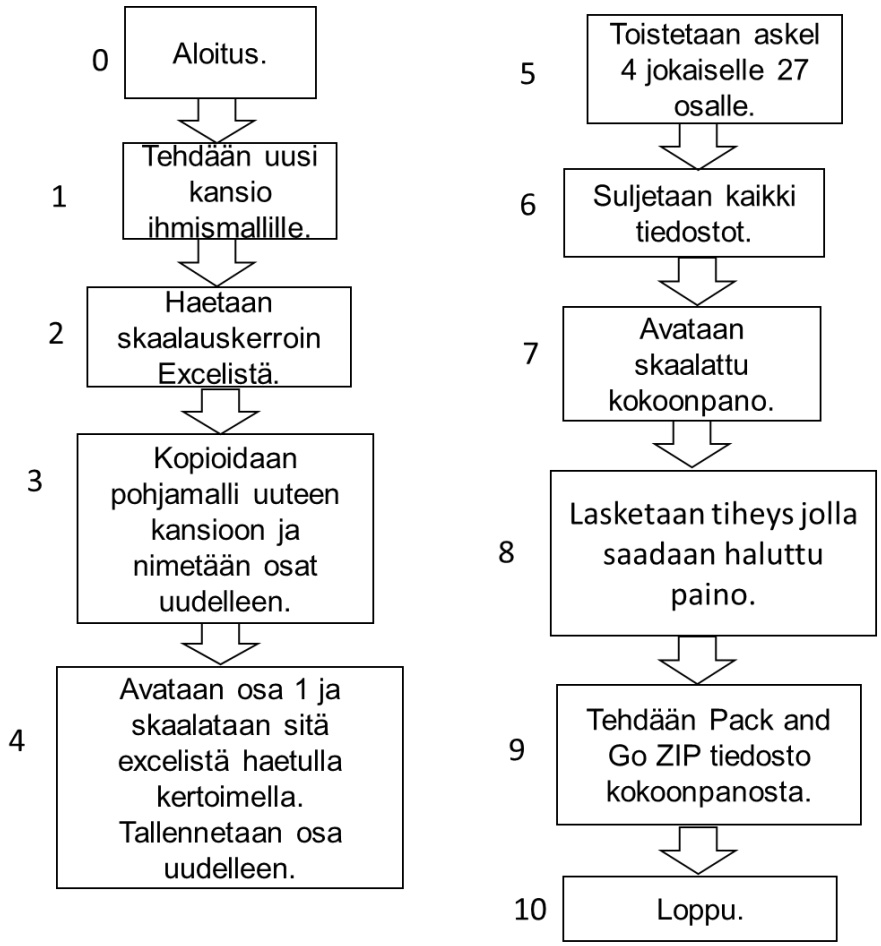
7.2 Makron suorittamien

Kuvassa 16 näkyy mistä moduuleista Human model scaler -makro koostuu. Kuvassa näkyvä *EPackandGo*-moduuli on sama moduuli, joka on Box 3.0 -makrossa. Tässä makrossa moduulit B- ja E-moduulit ovat sellaisia moduuleja, joita voidaan käyttää muissa makroissa. B-moduuli siirtää kaikki kokoonpanon tiedostot kansiosta toiseen ja E-moduuli tekee ZIP-tiedoston kokoonpanosta.



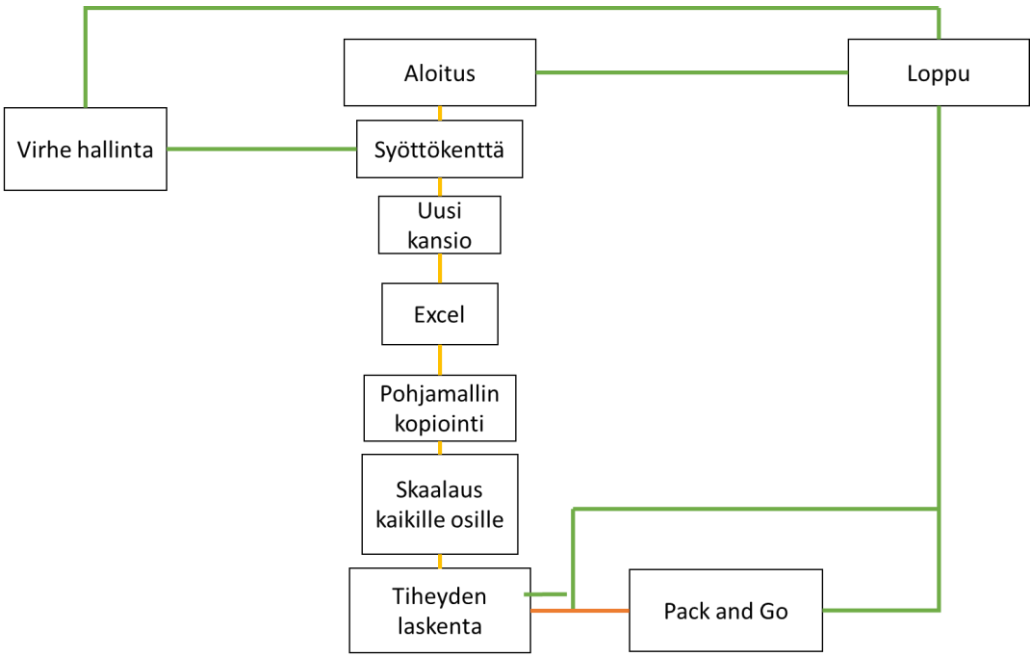
KUVA 176. Human model scaler -makron moduulit

Kuvassa 18 näkyy, mitä Human model scaler -makro tekee SolidWorksissä. Askel 0 tapahtuu moduulissa F. Askeleet 1 ja 2 tapahtuvat ModelScaler UserForm -moduulissa. Askel 3 tapahtuu moduulissa B. Askeleet 4 ja 5 tapahtuvat C-moduulissa. Askeleet 7 ja 8 tapahtuvat moduulissa D. Askel 9 tapahtuu moduulissa E.



KUVA 18. Human model scaler -makron operaatiot

Human model scaler -makron mahdolliset polut näkyvät kuvassa 19. Taulukossa 10 on selvitys siitä, miten millekin poluille päädytään.



KUVA 19. Human model scaler -makron suorittamisen polut

TAULUKKO 10. Human model scaler -makron polkujen värien selvitys

Väri	Polku
Vihreä	Vihreää viivaa suoritetaan silloin, kun koodin ajo on loppumassa tai painetaan EXIT-painiketta käyttöliittymästä.
Keltainen	Keltaista viivaa suoritetaan aina, kun makrossa ei tule suorittamisen virhettä.
Oranssi	Oranssia viivaa suoritetaan silloin, kun käyttöliittymästä valitaan Pack and Go vaihtoehto päälle.

7.3 Skaalaus jatkumolohko

Seuraavassa esimerkissä käydään läpi mitä mikäkin rivi *CScaleLoop*-moduulissa tekee. Taulukossa 11 on tarkat selitykset kullekin koodiriville. Esimerkistä on jätetty Early binding- ja deklaraatio-osio pois, koska nämä eivät ole esimerkin kannalta oleellisia.

1. *i = 1*
2. *While i < 28*
3. *strNewBodyFilePath = strMacroPath & "\Human Model" & ModelScaler.strNewFolderPath & "\Human model Part" & i & ".sldprt"*
4. *Debug.Print strNewBodyFilePath*
5. *swApp.OpenDoc6 strNewBodyFilePath, swDocPART, 128, "", lngErrors, lngWarnings*
6. *Set swModel = swApp.ActiveDoc*
7. *Set swFeatMgr = swModel.FeatureManager*
8. *Set swScaleFeat = swFeatMgr.InsertScale(1, True, ModelScaler.dbScaleFactorY, 1, 1)*
9. *swModel.ForceRebuild3 False*
10. *swModel.Save3 1, lngErrors, lngWarnings*
11. *swApp.CloseAllDocuments True*
12. *i = i + 1*
13. *Wend*

TAULUKKO 11. CScaleLoop-moduuliin rivien selitykset

Rivi	Selitys
1	Määritellään i-muuttujan arvoksi 1
2	Aloitetaan *While*-lohko, joka toistetaan 27 kertaa. Lohko toistetaan 27 kertaa, koska ihmismallin osia on 27 kappaletta.

(jatkuu)

TAULOKKO 11. (jatkuu)

3	Asetetaan strNewBodyFilePath-muuttujaan avattavan tiedoston polku. Polussa i-muuttuja määrittelee, mikä tiedosto avataan milläkin kierroksella.
4	Rivi tulostaa immediate-ikkunaan, mikä polku on tallennettuna *strNewBodyFilePath*-muuttujaan.
5	Komento avaa tiedoston, joka on strNewBodyFilePath-muuttujassa.
6	Asettaa swModel-muuttujaan aktiivisen SolidWorks-dokumentin.
7	Tekee early binding -toiminnon swFeatMgr-muuttujalle. Eli swFeatMgr-muuttujalla voidaan käyttää IFeatureManager-rajapinnan sisältämiä komentoja.
8	Rivi tekee skaalauksen yhtenäisellä kertoimella jokaisen akselin suhteen. *ModelScaler.dbScaleFactorY* kohta hakee Excel-tiedostosta kertoimen, jolla skaalaus tehdään. *dbScaleFactorY* edessä on *ModelScaler,* koska kyseinen muuttujan arvo on *ModelScaler* UserForm -moduulissa.
9	Rivi rakentaa uudelleen swModel-muuttujassa olevan tiedoston.
10	Rivi sulkee kaikki aktiiviset dokumentit.
12	Rivi kasvattaa i-muuttujan arvoa yhdellä.
13	While...Wend-lohkon lopetus. Tämän jälkeen komentojen suorittaminen menee takaisin riville 1. Tätä toistetaan niin kauan, kun i-muuttujan arvo on vähemmän kuin 28.

7.4 Ihmismallin massa

Human model scaler -makrossa oli tarkoitus pystyä muokkaamaan ihmismallin massaa kirjoittamalla haluttu paino makron käyttöliittymään. Tämä ei kuitenkaan onnistu, sillä VBA-makrolla ei pysty muokkaamaan sldmat-tiedostoa. Sldmat -tiedostotyyppi on SolidWorksin käyttämä materiaalitiedostotyyppi. Makron tarkoituksena oli muokata mallille tehdyn Body material -materiaalin tiheyttä sellaiseksi, että sillä saataisiin haluttu paino. Tätä ongelmaa ei pystytty ratkaisemaan, joten makro laskee tarvittavan materiaalin tiheyden ja lisää sen ihmismallin properties-tietoihin. Tämän jälkeen makron käyttäjä joutuu itse muokkaamaan materiaalin tiheyden samaksi, mikä on properties-tiedoissa.

8 VBA-MAKROJEN TALOUDELLISET HYÖDYT

Solidworks API-sovelluksella tehdyt VBA-makrot voivat tuottaa huomattavan ajansäästön yritykselle, koska makroilla voidaan automatisoida toistuvia toimintoja SolidWorksin käytössä. VBA-makrot estävät myös inhimillisen virheen sattumisen hyvin tehokkaasti. Ajansäästön taloudellista arvoa on hyvin vaikea arvioida. Makrot täytyy tehdä yritys- ja tapauskohtaiseksi, minkä takia säästetty aika on täysin riippuvainen siitä, minkä ongelma ratkaisemiseksi makro tehdään.

VBA-makroilla voidaan automatisoida kaikki toiminnot, jotka ovat matemaattisesti kuvattavissa. Esimerkiksi makroilla voidaan automatisoida ruuvin ja mutterin laittaminen jokaiseen samankokoiseen reikään. VBA-makroilla on tehty muun muassa

- nettisivulla olevia osageneraattoreita
- piirustuksien generointi makroja
- tiedostonhallintaan ja -nimeämiseen liittyviä makroja
- osan properties-tietojen pohjalta automatisoitu dokumentaation tekemiseen liittyviä makroja

Kaikista yleisin yritysten tapa käyttää makroja internetpohjainen 3D-malligeneraattori. Näillä generaattoreilla pystytään tulostamaan 3D-malleja monille eri suunnitteluohjelmistoille. Koska 3D-malli tulostetaan juurimallien pohjalta moneen eri ohjelmistoon, tulostetut 3D-mallit eivät säilytä metatietoja mallin mukana. Metatieto on dataa, joka kulkee tiedoston mukana, esimerkiksi tallennuspäivämäärä. Koska metatieto ei säily tiedoston mukana, saatavissa 3D-malleissa ei ole toimivia mate-liitoksia. Jos yritykset tekisivät jokaiselle ohjelmistolle oman 3D-malligeneraattorin, metatiedot säilyisivät malleissa.

Ennen makron kirjoittamista kirjoittajan tulee miettiä millaiseen ja kuinka laajaan ongelmaan makroa ollaan tekemässä. Pitkän makron kirjoittaminen kokematto-malle API-ohjelmoijalle on työläs prosessi. Tämän takia makron kirjoittajalla olisi hyvä olla kokemusta API-ohjelmoinnista, jotta kirjoittaja osaisi arvioida, kuinka

kauan hänellä menee makron tekemiseen. Tämä auttaa makron kirjoittajaa tekemään laskelmia kannattaako makroa tehdä itse vai kannattaako se tehdä alihankintana. Yhteen makroon ei kannata tehdä suuria määriä toimintoja. Mitä monimutkaisempi makro on, sitä epävakaampi se on ja sitä vaikeampi se on kirjoittaa.

Yrityksen tulee miettiä kannattaako yrityksen kouluttaa omaa API-koodaajaa yrityksen tarpeisiin. Koodin opiskelu tehokkaaseen käyttöön asti on aikaa vievä prosessi. Usein yrityksen on järkevintä tehdä API-ohjelmointi alihankintana, sillä VBA-kielen opiskelu voi viedä satoja tunteja. Tämän takia yrityksen olisi syytä tehdä tarkat investointilaskelmat investoinnin kannattavuudesta. Jos tehtävien makrojen tarve on suuri, tällöin investointi on kannattava. Yrityksessä olisi kuitenkin hyvä olla työntekijä, jolla on jonkinlaista ohjelmointikokemusta ratkaisemaan ongelmat makrojen kanssa. On lähes varmaa, että joidenkin päivitysten myötä vanhat makrot eivät toimi.

SolidWorksin VBA-makrojen käyttö on hyvä keino aloittaa API-ohjelmoinnin opiskelu, sillä VBA on ohjelmointikielenä helppo oppia. VBA-kieli ei ole yhtä tehokas verrattuna NET-kieliin. NET-kieliä, jotka toimivat SolidWorksissä, ovat VB-NET ja C#. Koodin tehokuudella tarkoitetaan sitä, että sama asia voidaan tehdä vähemmillä koodiriveillä verrattuna VBA-kieleen. NET-kielillä voidaan kirjoittaa omia add in -sovelluksia SolidWorksiin. Add in -lisäosilla voidaan tehdä omia työkaluja SolidWorksiin. Näitä lisäosia ei tarvitse suorittaa samalla lailla kuin VBA-makroja, vaan ne ovat aina käytettävissä SolidWorksin käyttöliittymässä.

9 YHTEENVETO

Työssä selvitettiin, kuinka hyvin SolidWorksin VBA-makroja voidaan käyttää parametrisessä mallinnuksessa. Työn tavoitteena oli selvittää, kuinka hyvin VBA-ohjelmointikielen makrot soveltuvat suunnittelun työkaluksi ja kuinka hyvin ne toimivat päivittäisessä käytössä. Toisena tavoitteena oli tehdä toimiva pilottimakro valitusta aiheesta. Työn lopussa pohdittiin VBA-makrojen käytettävyyttä yrityksen päivittäisessä toiminnassa.

Työssä selvisi, että VBA-kielisillä makroilla on hyvin suuri potentiaali tuoda hyötyä yrityksen toimintaan. VBA-makroilla yritys voi automatisoida lähes jokaisen toistuvan tehtävän SolidWorksiä käytettäessä. Tämän takia VBA-makrot tuovat ajansäästöä SolidWorksin käyttäjän toimintaan. VBA-makrot voivat estää täysin kokonaan inhimillisen virheen tapahtumisen esimerkiksi tiedostojen versiohallinnassa.

VBA-kielen makroilla saavutettava hyöty parametrisessä mallinnuksessa ei ole niin suuri kuin tiedostonhallintaan perustuvilla makroilla. Ongelma näissä parametriseen mallinnukseen liittyvissä makroissa tulee siitä, ettei niitä ei voida käyttää niin monessa paikassa kuin tiedostonhallintamakroja. Tiedostohallintamakroilla tarkoitetaan tässä esimerkiksi oikeanlaisen osanumeron lisäämistä mallin properties-tietoihin.

Parametriseen mallinnukseen liittyvillä makroilla pystytään tekemään esimerkiksi 3D-malligeneraattori, joka tulostaa mallin annettujen mittojen perusteella. Tällaisella 3D-malligeneraattorilla voidaan tehdä 3D-malli jokaisesta tuotteen eri koosta. Toisin sanoen voidaan tehdä yksi malli yhdelle sylinterityypille ja 3D-malligeneraattorilla voidaan tämän jälkeen tulostaa kaikki sylinterin eri kokoluokat. Ainoa rajoite näillä 3D-malleilla on, että niiden täytyy olla geometrisesti samanmuotoisia. Tällainen generaattori voi tuottaa hyötyä yritykselle, mutta tällaisten generaattorien tarve on kuitenkin harvinaista.

VBA-makrojen tehokkaaseen käyttämiseen tarvittava VBA-kielen ja SolidWorksin API-sovelluksen opiskelu oppiminen on hyvin aikaa vievä prosessi. Työntekijällä, jolla ei ole aikaisempaa kokemusta ohjelmoinnista, voi mennä satoja tunteja oppia tekemään makroja tehokkaasti. VBA-kieli on kaikista nopein oppia mahdollisista SolidWorks-makroihin sopivista kielistä. Suuri oppimisen tarve tulee hyvin suuresta komentojen määrästä. Komentojen nimiä tulee opetella ulkoa, jotta makroja voidaan kirjoittaa tehokkaasti. Yksinkertaisia makroja voidaan tehdä hyvin lyhyessä ajassa, sillä VBA-kielen makroista löytyy paljon esimerkkejä ja ohjemateriaalia internetistä.

Työn aikana huomattiin, että VBA-ohjelmointikieli ei ole välttämättä kaikista paras vaihtoehto makrojen ohjelmointikieleksi. VBA-kieli ei ole yhtä tehokas verrattuna VB.NET- ja C#-kieliin. Ohjelmointikieli on tehokas, kun monta asiaa voidaan suorittaa yhdellä komennolla. VBA-ohjelmointikieltä suositellaan aloittelijalle, koska VBA on helppolukuista ja se on helppo oppia. VBA-ohjelmointikieli on vakaampi kuin NET-kielet.

Työssä tehtiin kaksi pilottimakroa, koska työtä tehtäessä nähtiin parhaaksi tehdä aluksi yksi harjoittelumakro. Tämä harjoittelumakro on BOX 3.0 -makro. Varsinainen työn pilotti on Human model scaler -makro. BOX 3.0 -makron pituus on 823 koodiriviä ja sen tekemiseen meni aikaa noin 80 tuntia. Human model scaler -makron pituus on 411 koodiriviä, ja sen tekemiseen meni noin 20 tuntia. Tähän 20 tuntiin ei lasketa varsinaisen ihmismallin tekemistä. Näistä ajoista huomataan makrojen tekemisen nopeutuminen. Tämä nopeus saavutettiin käyttämällä BOX 3.0 -makroon tehtyä Pack and Go -moduulia. Human model scaler -makron tekeminen nopeutui myös komentojen muistamisen ja virheiden vähenemisen takia.

Tämä opinnäytetyö oli hyvin laaja ja antoi hyvän pohjan makrojen tekemiseen tulevaisuudessa. Tässä työssä ei käyty läpi kaikkia SolidWorks APIn toimintoja, jotka voivat auttaa makrojen tekemisessä ja suunnittelutyössä. Tärkein näistä toiminoista on event trigger -toiminto. Työn suurin haaste oli siinä, ettei ohjelmoinnista ei ollut aikaisempaa kokemusta. Toinen haaste oli hyvän lähdemateriaalin löytäminen. SolidWorksin ohjelmoinnista on hyvin vähän suomenkielistä

materiaalia olemassa, joten kaikki käytetty materiaali oli englanninkielistä. Työn aikana opittiin hyvin, miten monimutkaisia makroja tehdään. Työ antoi myös hyvän pohjan SolidWorksiin add in -lisäosien tekemiselle.

LÄHTEET

1. Company History. Dassault Systems. Saatavissa: https://www.solidworks.com/sw/656_ENU_HTML.htm. Hakupäivä 10.3.2016.

2. Record SolidWorks Macro. SOLIDWORKS API Help. Saatavissa: http://help.solidworks.com/2013/English/api/sldworksapiprogguide/GettingStarted/Record_Solidworks_Macro.htm. Hakupäivä 10.3.2016.

3. RECORD/PAUSE Macro. SOLIDWORKS API Help. Saatavissa: http://help.solidworks.com/2013/English/SolidWorks/sldworks/t_record_pause_macro.htm Hakupäivä 10.3.2016.

4. SOLIDWORKS Macros. SOLIDWORKS API Help. Saatavissa: http://help.solidworks.com/2015/English/api/sldworksapiprogguide/GettingStarted/SolidWorks_Macros.htm Hakupäivä 11.3.2016.

5. Microsoft Visual Studio Tools for Applications Requires Microsoft .NET Framework 3.5. SOLIDWORKS API Help. Saatavissa: http://help.solidworks.com/2015/english/api/sldworksapiprogguide/gettingstarted/vsta_dependency_on_dotnet_3dot5.htm. Hakupäivä 11.3.2016.

6. Use the Properties Widow. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/ee440574(v=office.12).aspx. Hakupäivä20.3.2016.

7. Code Window. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/ee440738(v=office.12).aspx. Hakupäivä 20.3.2016.

8. Immediate Widow. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/f177hahy.aspx. Hakupäivä 20.3.2016.

9. Editing a Value in the Watch Window. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/aa6fy2x5(v=vs.71).aspx. Hakupäivä 20.3.2016.

10. What is a DLL?. Microsoft. Saatavissa: https://support.microsoft.com/en-us/kb/815065. Hakupäivä 18.3.2016.

11. History of Visual Basic. Max Visual Basic. Saatavissa: http://www.max-visual-basic.com/history-of-visual-basic.html. Hakupäivä 25.3.2016.

12. WELCOME. SOLIDWORKS API Help. Saatavissa: http://help.solidworks.com/2015/English/api/sldworksapiprogguide/Welcome.htm. Hakupäivä 25.3.2016.

13. SOLIDWORKS API Object Model Overview. SOLIDWORKS API Help. Saatavissa: http://help.solidworks.com/2016/English/api/sldworksapiprogguide/GettingStarted/SolidWorks_API_Object_Model_Overview.htm?verRedirect=1. Hakupäivä 28.3.2016.

14. Rice, Keith. 7 Mistakes New SolidWorks API Programmers Make. CadSharp. Saatavissa: http://www.cadsharp.com/blog/7-mistakes-new-solidworks-api-programmers-make/. Hakupäivä 28.3.2016.

15. Interfaces. SOLIDWORKS API Help. Saatavissa: http://help.solidworks.com/2013/English/api/SWHelp_List.html?id=c64d3add5f4443bba1a7070ab6358873#Pg0. Hakupäivä 28.3.2016.

16. Rice, Keith. The SolidWorks API Object Model. CadSharp. Video. Saatavissa: http://www.cadsharp.com/videos/lesson-2-3-vba/ (vaatii käyttäjälisenssin). Hakupäivä 15.3.2016.

17. Accessing SOLIDWORKS Objects. SOLIDWORKS API Help. Saatavissa: http://help.solidworks.com/2016/english/api/sldworksapiprogguide/overview/accessing_objects.htm. Hakupäivä 28.3.2016.

18. Understanding the SOLIDWORKS API Class Hierarchy. SOLIDWORKS API Help. Saatavissa: http://help.solidworks.com/2016/english/api/sldworksapiprogguide/gettingstarted/understanding_the_solidworks_api_class_hierarchy.htm?verRedirect=1. Hakupäivä 28.3 2016.

19. Rice, Keith. Working With Configurations. CadSharp. Video. Saatavissa: http://www.cadsharp.com/videos/lesson-3-2-vba/. (vaatii käyttäjälisenssin) Hakupäivä 16.3.2016.

20. Rice, Keith. Using Variables. CadSahrp. Saatavissa: http://www.cadsharp.com/videos/lesson-1-2-vba/. Hakupäivä 30.3.2016.

21. Data Type Summary (Visual Basic). Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/47zceaw7.aspx. Hakupäivä 30.3.2016.

22. Dim Statement. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/7ee5a7s1.aspx. Hakupäivä 30.3.2016.

23. Private Statement. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/wx059ey1.aspx. Hakupäivä 30.3.2016.

24. Public Statement. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/9dc6we3z.aspx. Hakupäivä 30.3.2016.

25. Hungarian Notation. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/aa260976(v=vs.60). Hakupäivä 31.3.2016.

26. Early and Late Binding. SOLIDWORKS API Help. Saatavissa: http://help.solidworks.com/2012/English/api/sldworksapiprogguide/GettingStarted/Early_and_Late_Binding.htm. Hakupäivä 29.3.2016.

27. Type Libraries. SOLIDWORKS API Help. Saatavissa: http://help.solidworks.com/2012/English/api/sldworksapiprogguide/Overview/Type_Libraries.htm. Hakupäivä 29.3.2016.

28. Visual Basic-Specific IntelliSense. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/cxy240ac.aspx. Hakupäivä 29.3.2016.

29. Rice, Keith. Modular Programming. CadSahrp. Saatavissa: http://www.cadsharp.com/videos/lesson-1-7-vba/. Hakupäivä 15.3.2016.

30. Rice, Keith. UserForms and Controls. CadSahrp. Saatavissa: http://www.cadsharp.com/videos/lesson-1-8-vba/. Hakupäivä 16.3.2016.

31. Compatibility Between the 32-bit and 64-bit Versions of Office 2010. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/ee691831.aspx. Hakupäivä 30.3.2016.

32. Visual Basic Language Keywords. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/ksh7h19t(v=vs.90).aspx. Hakupäivä 30.3.2016.

33. Comparison Operators in Visual Basic. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/215yacb6.aspx. Hakupäivä 1.4.2016.

34. Select...Case Statement. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/cy37t14y.aspx. Hakupäivä 1.4.2016.

35. & Operator (Visual Basic). Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/wfx50zyk.aspx. Hakupäivä 1.4.2016.

36. If...Then...Else Statement (Visual Basic). Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/752y8abs(v=vs.90).aspx. Hakupäivä 1.4.2016.

37. While...Wend Statement. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/office/gg264682.aspx. Hakupäivä 1.4.2016.

38. For...Next Statement (Visual Basic). Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/5z06z1kb(v=vs.90).aspx. Hakupäivä 1.4.2016.

39. Rice, Keith. Selection. CadSharp. Video. Saatavissa: http://www.cadsharp.com/videos/lesson-3-4-vba/. (vaatii käyttäjälisenssin) Hakupäivä 17.3.2016.

40. Rice, Keith. System and Document Options. CadSharp. Video. Saatavissa: http://www.cadsharp.com/videos/lesson-3-5-vba/. (vaatii käyttäjälisenssin) Hakupäivä 17.3.2016.

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
AddComponent4	IAssemblyDoc	Lisää määritellyn komponentein kokoonpanoon.	swAssy.AddComponent4(strKuutioPath, "", 0, 0, 0)	http://help.solidworks.com/2014/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IAssemblyDoc~AddComponent5.html
AddMate5	IAssemblyDoc	Lisää määritellyn mate-liitoksen kokoonpanoon.	swAssy.AddMate5(1, 1, False, 0.0035, 0, 0, 0.001, 0.001, 0, 0.5235987755983, 0.5235987755983, False, False, 0, lngErrors)	http://help.solidworks.com/2014/english/api/sldworksapi/SolidWorks.Interop.sldworks~Soli
GetComponents	IAssemblyDoc	Hakee kaikki komponentit aktiivisesta konfiguraatiosta.	swAssy.GetComponents	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.iassemblydoc~getcomponents.html
GetFaces	IBody2	Hakee aktiivisen tiedoston kaikki pinnat.	swBody.GetFaces	http://help.solidworks.com/2012/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.ibody2~getfaces.html

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
FeatureByName	IPartDoc	Valitsee nimetyn featuren historiapuusta.	swModel.FeatureByName("Boss-Extrude1")	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.ipartdoc~featurebyname.html
Add3	ICustomPropertyManager	Lisää tiedoston ominaisuuksiin määritellyn tiedon.	swCustPropMgr.Add3 "Osa numero"	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solid
Get5	ICustomPropertyManager	Hakee tiedoston määritetyn ominaisuuden arvon.	swCustPropMgr.Get4 strCustProp, True, Empty, strValOut	http://help.solidworks.com/2014/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.ICustomPropertyManager~Get5.html
SystemValue	IDimenssion	Valitsee määritetyn parametrin järjestelmän arvon.	swModel.Parameter("D1@Sketch2").SystemValue	help.solidworks.com/2014/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IDimension~ISetSystemValue3.html

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
Select4	IEntity	Valitsee määritellyn objektin.	swTAPPIF2.Select4 True, Nothing	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solid
SetDebth	IExtrudeFeatureData2	Asettaa featuren syvyyden eteen tai taaksepäin.	swExtrudeFeatData.SetDepth	http://help.solidworks.com/2014/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IExtrudeFeatureData2~SetDepth.html
GetDefinition	IFeature	Hakee featuren määritelmän, jotta sitä voitaisiin muuttaa.	swFeat.GetDefinition	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.ifeature~getdefinition.html
ModifyDefination	IFeature	Päivittää kaikki järjestelmän arvot, jotka on hankittu GetDefination menetelmällä.	swFeat.ModifyDefinition swExtrudeFeatData, swModel, Nothing	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.ifeature~modifydefinition.html

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
Parameter	IFeature	Valitsee nimetyn parametrin.	swModel.Parameter("D1@Sketch2")	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.ifeature~parameter.html
ClearSelection2	IMdelDoc2	Tyhjentää kaikki valinnat.	swAssy.ClearSelection2 False	http://help.solidworks.com/2014/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IModelDoc2~ClearSelection2.html
ShowNamedView2	IMdelDoc2	Näyttää nimetyn näkymän. Esimerkiksi *Isometric.*	swAssy.ShowNamedView2 "Isometric", -1	http://help.solidworks.com/2014/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IModelDoc2~ShowNamedView2.html
ViewZoomtoFit2	IMdelDoc2	Zoomaa aktiivisen näkymään mahtumaan näytölle.	swAssy.ViewZoomtofit2	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.imodeldoc2~viewzoomtofit2.html

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
EditRebuild3	IModelDoc2	Rakentaa uudelleen kaikki featuret, jotka tarvitsevat uudelleen rakennusta.	swAssy.EditRebuild3	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.imodeldoc2~editrebuild3.html
EditSketch	IModelDoc2	Sallii valitun sketsin muokkaamisen.	swModel.EditSketch	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.imodeldoc2~editsketch.html
ForceRebuild3	IModelDoc2	Pakottaa uudelleen rakentamaan kaikki featuret, jos niitä tarvitsee uudelleen rakentaa.	swModel.ForceRebuild3 False	http://help.solidworks.com/2014/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IModelDoc2~ForceRebuild3.html
GetEntityName	IModelDoc2	Hakee pinnan, reunana tai huipun nimen perusteella.	swModel.GetEntityName(swTAPPIF1	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solid

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
CustomPropertyManager	IModelDocExtension	Hakee kaikki dokumentin ominaisuudet.	swModel.Extension.CustomPropertyManager("")	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solid
GetPackAndGo	IModelDocExtension	Hakee Pack and Go objektin.	swModel.Extension.GetPackAndGo	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.imodeldocextension~getpackandgo.html
SaveAs	IModelDocExtension	Tallentaa tiedoston uudella nimellä.	swModel.Extension.SaveAs strTAPPICSavepath, 0, 1, Nothing, lngErrors, lngWarnings	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.imodeldocextension~saveas.html
SavePackAndGo	IModelDocExtension	Tallentaa kaikki tiedostot, jotka on kohdistettu Pack and Go:hon yhteen ZIP-tiedostoon tai yhteen kansioon.	swModel.Extension.SavePackAndGo(swPackAndGo)	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.imodeldocextension~savepackandgo.html

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
SelectByID2	IModelDocExtension	Valitsee määritetyn objektin.	swModel.Extension.SelectByID2 "D1@Sketch2@TAPPI1.SLDPRT", "DIMENSON", 0, 0, 0, False, 0, Nothing, 0	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.imodeldocextension~selectbyid2.html
GetDocumentNames	IPackAndGo	Hakee kaikki alkuperäiset polut ja tiedostojen nimet Pack and Go -toimintoa varten.	swPackAndGo.GetDocumentNames	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.ipackandgo~getdocumentnames.html
GetDocumentNamesCount	IPackAndGo	Hakee tiedostojen lukumäärän Pack and Go -toimintoa varten.	swPackAndGo.GetDocumentNamesCount	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.ipackandgo~getdocumentnamescount.html

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
IncludeDrawings	IPackAndGo	Hakee piirustukset ja niiden nimet Pack and Go -toimintoa varten.	swPackAndGo.IncludeDrawings = True	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.ipackandgo~includedrawings.html
SetDocumentSaveToNames	IPackAndGo	Asettaa tiedostojen polut ja nimet Pack and Go –toimintoa varten.	swPackAndGo.SetDocumentSaveToNames strSetPathNames	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.ipackandgo~setdocumentsavetonames.html
SetSaveToName	IPackAndGo	Asettaa kaikkien Pack and Go tiedostojen nimet ja polut saman muuttujan alle.	swPackAndGo.SetSaveToName True, vPaGPath	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.ipackandgo~setsavetoname.html

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
GetBodies	IPartDoc	Hakee aktiivisen osatiedoston rungon.	swComp.GetBodies3(-1, Empty)	http://help.solidworks.com/2012/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IPartDoc~GetBodies2.html
ActiveDoc	ISldWorks	Hakee tällä hetkellä aktiivisen dokumentin.	swApp.ActiveDoc	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.isldworks~activedoc.html
CloseAllDocuments	ISldWorks	Sulkee kaikki avoinna olevat SolidWorks tiedostot.	swApp.CloseAllDocuments True	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.isldworks~closealldocuments.html
DocumentVisible	ISldWorks	Komento tekee sille määritellystä tiedosto tyypistä näkyvän tai näkymättömän.	swApp.DocumentVisible False, swDocPART	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.isldworks~documentvisible.html

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
GetCurrentMacroPathFolder	ISldWorks	Hakee tämän makron kansion polun.	swApp.GetCurrentMacroPathFolder	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.isldworks~getcurrentmacropathfolder.html
GetOpenDocumentByName	ISldWorks	Hakee avoinna olevan tiedoston määritellyllä nimellä.	swApp.GetOpenDocumentByName	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.isldworks~getopendocumentbyname.html
GetUserPreferenceStringValue	ISldWorks	Hakee aktiivisen SolidWorksiin käyttäjän asettaman arvon. Voidaan käyttää asetusten muokkaamiseen.	swApp.GetUserPreferenceStringValue(swDefaultTemplateAssembly)	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.isldworks~getuserpreferencestringvalue.html

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
NewDocument	ISldWorks	Luo uuden dokumentin määritellystä templatesta.	swApp.NewDocument(strDefAssTemp, 0, 0, 0)	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.isldworks~newdocument.html
OpenDoc6	ISldWorks	Aukaisee tiedoston ja valitsee kappaleen.	swApp.OpenDoc6(strKuutioPath, swDocPART, 1, "", lngErrors, lngWarnings)	http://help.solidworks.com/2014/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.isldworks~opendoc6.html
Option Explicit		Kun tämä komento on päällä, tällöin jokainen muuttuja täytyy nimetä erikseen.	Option Explicit	http://help.solidworks.com/2012/English/api/sldworksapiprogguide/GettingStarted/Option_Explicit_Statement.htm
swDocASSEMBLY		Ilmaisee komennolle, että kyseessä on kokoonpano tiedosto.	swDocASSEMBLY	http://help.solidworks.com/2014/english/api/swconst/SolidWorks.Interop.swconst~SolidWorks.Interop.swconst.swDocumentTypes_e.htm

Komento/Termi/Rivi	Interface	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
swDocDRAWING		Ilmaisee komennolle, että kyseessä on piirustus-tiedosto.	SwDocDRAWING	http://help.solidworks.com/2014/english/api/swconst/SolidWorks.Interop.swconst~SolidWorks.Interop.swconst.swDocumentTypes_e.htm
swDocPART		Ilmaisee komennolle, että kyseessä on osa-tiedosto.	swDocPART	http://help.solidworks.com/2014/english/api/swconst/SolidWorks.Interop.swconst~SolidWorks.Interop.swconst.swDocumentTypes_e.htm

Komento/Termi/Rivi	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
Call	Komento siirtää makron ajon toiseen moduuliin tai proseduuriin.	Call ExcelCode	
Case	Suorittaa yhden valitun rivi kokoelman riippuen Case arvosta.	Select Case vStatus(i)	https://msdn.microsoft.com/en-us/library/office/gg278665.aspx
CDbl	Nämä toiminnot määrittelevät, missä datamuodossa käyttäjän antama tieto pitää käsitellä. Kyseinen muoto käsitellee tietoa double-tiedostotyyppinä	dbCubeSize = CDbl(txtCubeSize)	https://support.office.com/en-US/article/Type-Conversion-Functions-8EBB0E94-2D43-4975-BB13-87AC8D1A2202
Cells.Value	Hakee Excel taulukosta määritellyn solun arvon.	vExcelHoleSize = Cells(intRow, 3).Value	https://msdn.microsoft.com/en-us/library/office/ff195193.aspx
Close	Avainsana close sulkee toiminnon tai ohjelman.	xlWb.Close True	https://support.office.com/en-US/article/Close-Macro-Action-59887C57-FCBE-4991-AFC8-1790018137B1
CloseDoc	Sulkee määritellyn tiedoston.	swApp.CloseDoc ("TAPPI.SLDPRT")	elp.solidworks.com/2012/English/api/sldworksapi/Close_Document_Example_VB.htm
Cmd	Cmd alkupääte viittaa SolidWorksin APIn käyttäjälomakkeiden command button -painikkeiden toimintaan.	CmdExit_Click	Hungarina notiation
Cvar	Nämä toiminnot määrittelevät, missä datamuodossa käyttäjän antama tieto pitää käsitellä. Kyseinen muoto käsitellee tietoa Variant-datatyyppinä	vExcelValue = CVar(txtExcel)	https://support.office.com/en-US/article/Type-Conversion-Functions-8EBB0E94-2D43-4975-BB13-87AC8D1A2202
Debug.print	Tulostaa muuttujan tai muuttujan arvon immediate-ikkunaan.	Debug.Print lngErrors	https://msdn.microsoft.com/en-us/library/aa716276(v=vs.60).aspx

Komento/Termi/Rivi	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
Empty	Jos komennon muuttuvissa arvoissa on tilalla sana Empty, tällöin kyseinen muuttuja ohitetaan. Sanalla voidaan myös ilmaista tyhjää arvoa.	swCustPropMgr.Get4 strCustProp, True, Empty, strValOut. If strValOut <> Empty Then	https://msdn.microsoft.com/en-us/library/office/gg264232.aspx
End	Pysäyttää proseduurin.	End	https://msdn.microsoft.com/en-us/library/office/gg251671.aspx
Exit Sub	Poistuu heti proseduurista, jossa Exit Sub -komento on.	Exit Sub	https://msdn.microsoft.com/en-us/library/office/gg251570.aspx
InStrRev	Etsii tekstistä määrätyn tekstin.	InStrRev(strSetPathNames(i), "\"))	https://support.office.com/en-US/article/InStrRev-Function-73193c1f-8db7-4558-ae91-fae75fe5fd56
Left	Hakee tekstistä määritellyn määrän merkkejä vasemmalta päin laskettuna.	Left(strSetPathNames(i)	https://support.office.com/en-US/article/LEFT-function-913824B4-FC42-4C8C-9229-0EAA57DCDF49
Len	Antaa tekstistä käytettyjen merkkien määrän.	Len(strSetPathNames(i)	https://support.office.com/en-us/article/LEN-function-038F2E0D-57D3-4F35-AE74-9A79D66E8FC2
MsgBox	Näyttää käyttäjälle teksti-ikkunan ja odottaa käyttäjän kuittausta. Palauttaa Integer-arvon siitä, mitä painiketta painettiin.	MsgBox "Unkown error. Dimension input must be numbers. Use posibbel excel codes"	https://msdn.microsoft.com/en-us/library/office/gg251821.aspx
On Error GoTo	Virheen sattuessa koodissa ohjelma suorittaa GoTo-avainsanan jälkeiset komennot tai proseduurin.	On Error GoTo Errhandeler	https://msdn.microsoft.com/en-us/library/office/gg251688.aspx

Komento/Termi/Rivi	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
Open	Komennolla avataan määritelty tiedosto.	Set xlWb = xlApp.Workbooks.Open(strExcelPath)	https://msdn.microsoft.com/en-us/library/office/gg264163.aspx
opt	Opt alku viittaa Solidworks APIn UserForm-moduuleissa käytettävin CheckBox-painikkeisiin.	optExcelLoop.Value	
Quit	Avainsana quit lopettaa toiminnon tai ohjelman.	xlWb.Quit	https://support.office.com/en-US/article/Quit-Macro-Action-3C67235D-A429-4D04-A455-113B611F18E2
ReDim	Funktion aikana tällä toiminolla voidaan muuttujalle antaa uusi arvo.	ReDim strSetPathNames(intDocNum - 1) As String	https://msdn.microsoft.com/en-us/library/office/gg251578.aspx
Replace	Komento kirjoittaa määritellyn tekstin päälle määrätyn tekstin.	Replace(strKuutioName, "Kuutio1", "Kuutio")	https://support.office.com/en-us/article/Replace-Function-6ACF209B-01B7-4078-B4B8-E0A4EF67D181
Right	Hakee tekstistä määritellyn määrän merkkejä oikealta päin laskettuna.	Right(strSetPathNames(i)	ttps://support.office.com/en-US/article/Right-Function-C02A18A8-B224-437E-AABA-1B785C6C61BF
Set	Kun rivin alussa Set-termi, tällöin muuttujalle, jonka edessä Set-termi on, asetetaan uusi arvo. Ominaisuuksissa Set asettaa halutulle tiedolle uuden arvon.	Set swApp = Application.SldWorks	https://msdn.microsoft.com/en-us/library/office/gg251642.aspx
Set xlApp = CreateObject("Excel.Application")	Luo ja antaa referenssin Excel ActiveX -objektiin.	Set xlApp = CreateObject("Excel.Application")	https://msdn.microsoft.com/en-us/library/office/gg264813.aspx
UBound	Antaa määritellyn arvon kokoelmasta.	For i = 0 To UBound(vStatus)	https://support.office.com/en-us/article/UBound-Function-212B1998-327B-4F1E-BA3F-DB4F9B5BA251?ui=en-

Komento/Termi/Rivi	Määritelmä	Esimerkki siitä, miten komento/termi voi ilmetä koodissa	Lähde
			US&rs=en-US&ad=US&fromAR=1
UCase	Hakee tekstistä isoja kirjaimia.	If UCase(Right(strPaGPath, 3)) = "ZIP" Then	https://support.office.com/en-us/article/UCase-Function-0C9B8B77-DD3B-46EC-AC46-B034FEF54224
Value	Value avainsana antaa muuttujan arvon. Esimerkiksi Boolean -arvo 1 tai 0.	If optExcelLoop.Value = True Then Call ExcelLoop	https://msdn.microsoft.com/en-us/library/office/ff195193.aspx
xlApp.Visible	Tekee Excel-ohjelmasta näkyvän, mikäli xlApp.Visible=True, jos xlApp.Visible=False, Excel-ohjelma suoritettaan vain tietokoneen muistissa.	xlApp.Visible = False	https://msdn.microsoft.com/en-us/library/office/ff866799.aspx