Juho Anttila

SUUNNITTELUAUTOMAATIO CAD-OHJELMISSA

Kone- ja tuotantotekniikan koulutusohjelma

2010

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

SUUNNITTELUAUTOMAATIO CAD-OHJELMISSA

Anttila, Juho
Satakunnan ammattikorkeakoulu
Kone- ja tuotantotekniikan koulutusohjelma
maaliskuu 2010
Ohjaaja: Teinilä, Teuvo
Sivumäärä: 42
Liitteitä: 7

Asiasanat: automatisointi, konfiguraattori, solidworks, vertex, autocad, vba, autolisp, makro

---

Tässä työssä tutkittiin suunnitteluautomaation mahdollisuuksia cad-ohjelmissa. Työtä varten perehdyttiin useampaan cad-ohjelmaan ja tehtiin suurehko automaatti Vertex G4 alustalle, jonka kokemuksiin työn pohdinta pitkälti perustuu. Työ etenee yleisestä työkalujen esittelystä automatisoinnin teoriaan. Teoriassa pohditaan erilaisia tapoja ja tasoja miten automatisointia voidaan toteuttaa. Työn aihe oli kiinnostava, koska aiheesta on tehty melko vähän teoksia.

Teoriapohdiskelun ohella esitettiin ja pohdittiin käytännön läheisempiä vaatimuksia automatisointiin liittyen. Näihin paneuduttiin tuotteen ja suunnittelijan näkökulmista.

Työn käytännön läheisyyttä esitellään hyötyjen ja vaarojen merkeissä, sekä usealla erityylisellä case-esimerkillä. Käytännön esimerkit tehtiin Visual Basic for Applications -kielellä SolidWorksiin sekä AutoLisp-kielellä Autocadiin.

Lopullisena tavoitteena oli saada aikaan teos, joka kuvaa alan mahdollisuudet havainnollistavin esimerkein.

DESIGN AUTOMATION IN CAD PROGRAMS

Anttila, Juho
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Program in Mechanical and Production Engineering
March 2010
Teinilä, Teuvo
Number of pages: 42
Appendices: 7

Key words: macro, configuration, solidworks, vertex, autocad, vba, autolisp

---

The main focus of this thesis was to investigate the possibilities of design automation. Several CAD programs were researched for this work, and a major CAD automaton was created with Vertex G4. This thesis was mainly built on experiences of the Vertex program. The work starts from tool introduction and continues to the theory of automation. In theory, different styles and levels of realizing automation were considered. The topic of the work was very interesting because only a few books have been written on this topic.

Along with theoretical consideration practical requirements concerning design automation were represented. Practical details were imagined from the point of view of the product and the designer. Practical case examples were made with Visual Basic for Applications to SolidWorks and AutoLisp to Autocad.

The final objective of this thesis was to create a book that describes the possibilities in the field with clear examples.

SISÄLLYS

1 JOHDANTO

1.1 Työn taustat

Tämän insinöörityön raportin lopullinen aihe muodostui rinnalla tehdyn käytännön osion perusteella projektin ollessa jo melko hyvässä vaiheessa. Raportin rinnalla on rakennettu suurehkoa suunnitteluautomaattia teräsrakenteiden suunnitteluun Raumaster Oy:lle. Aiheen valinta minulle henkilökohtaisesti oli luonnollinen, tämän kaltaisella aiheella pystyin yhdistämään sellaista osaamista, jota ei yleensä koneinsinöörillä ole. Tämä osaaminen on siis ohjelmointitaito yhdistettynä muuhun tekniseen insinööritietoon. Tässä tapauksessa ohjelmointi ja tekninen piirustus yhdistetään.

Otsikon taustaa ja tarkoitusta lienee myös hyvä hieman avata tarkemmin lukijalle. Suunnittelutyöllä tarkoitetaan cad-piirtämistä ja siihen liittyviä toimintoja, sitä mekaanista työtä, jota suunnittelija tekee tietokoneensa ja jonkun ohjelman avulla. Kirjaa ei ole varsinaisesti rajattu vain konetekniseen piirtämiseen, pääasiallisesti asiat on siirrettävissä niin rakennus- kuin sähkö-teollisuuteenkin. Automatisoinnilla taas tarkoitetaan edellä esitetyn suunnittelutyön automatisointia. Esimerkkinä käytän teollistumisen aikakautta, kun koneet tulivat helpottamaan työntekijän työtä, nopeuttamaan ja tehostamaan tuotantoa. Seuraava askel oli koneen automatisointi, josta seurasi tuottavuuden nousua tai jopa työntekijän korvaamista.

Vastaava malli toimii cad-työskentelyssä, ensin piirustuslauta korvattiin cad-ohjelmalla. Tässä kohtaa tuottavuus nousi huimasti mutta työntekijän rooli oli ja pysyi tärkeänä silti. Piirustustyötä kun automatisoidaan edelleen niin automaation roolia toimittaan ohjelma. Ohjelmistojen vaikutukset voivat olla yhtä suuria kuin tehtaan suorittavissa töissä. Kirjan sisällön ymmärtäminen vaatii jossain määriin teknisen piirtämisen tuntemusta ja mielellään jonkun 3d cad-ohjelman tuntemista edes välttävästi etukäteen.

Päätin hyvissä ajoin ennen käytännön työn valmistumista, että en kirjoita raporttia suoranaisesti Raumasterin projektista tai sen vaiheista, vaan paneudun siitä kertynei-

siin ajatuksiin ja kokemuksiin. Tässä kirjassa ei esitellä tekemäni suunnitteluautomaatin rakennetta, eikä suoranaisesti projektin kulkua. Sen sijaan tässä kirjassa esitetään yleisemmin tietoa ja kokemuspohjaista ajattelua suunnitteluautomaatiosta cad-ohjelmissa ja siihen liittyvien projektin taustatiedoista.

Korostan kuitenkin käytännön työn merkitystä, ilman sitä ei tämän kirjan ajatuksiakaan olisi pystytty jäsentämään. Aiheeni etsiminen oli lähtökohtaisesti omatoimista, mutta käytännön työn aiheen löytymisen mahdollisti Raumaster Oy.

Raumasterilla oli hieman aiempaa kokemusta jo tämän kaltaisista automaateista, ja sitä myötä heidän kiinnostuksensa projektiin oli melko suurta. Tällä hetkellä vallitseva huonohko taloustilanne tosin saattoi auttaa tämän kirjan syntyä, uuden ajatuksen läpi lyöminen hyvinä aikoina ei olisi ollut näin helppoa kuin nyt oli.

1.2 Tavoitteet

Käytännön työssä tavoiteltiin suunnittelun tehostumista, niin ajallisesti kuin siitä suoraan johtuvassa kustannusmielessä. Toisaalta tavoitteena käytännön työssä oli selvittää Vertex G4 mahdollisuuksia automatisoida suuria teräsrakenteita, tästä yrityksellä oli hieman epävarma olo alun perin.

Automatisoinnin asettamina vaatimuksina myös tuotteen standardointi-astetta pyrittiin nostamaan entisestään. Tämä vaati kompromissien tekoa tuotteen yksityiskohtiin liittyen, mutta tällekin annettiin painoarvoa suunnittelutyön tehostumisen näkökulmasta tulevaisuusaspektilla. Käytännön työn tavoitteet kulkivat pitkälti työn laajuuden rajauksen kanssa käsi kädessä. Tuote, jota automatisointiin, on niin laaja, että siihen olisi voitu tehdä yksityiskohtia todella paljon. Tämän työn puitteissa kuitenkin rajaukset tehtiin layout-suunnitteluun ja perusmallien aikaansaamiseksi automaatin kanssa.

Tälle kirjalle asetettiin ihan erilaiset tavoitteet kuin tavallisesti tämänkaltaisista töistä kirjoitetuissa raporteissa olisi. Tarpeettomaksi koettiin kirjoittaa projektikohtaisista yksityiskohdista, tai siitä miksi johonkin ratkaisuun pyrittiin, jos sillä valinnalla ei

ole suurta painoarvoa tulevissa projekteissa. Tässä kirjassa pyritäänkin esittämään objektiivisemmasta katsontakannasta asioita, joita tulee huomioida ja tiedostaa, kun tietokoneavusteista suunnittelua automatisoidaan tai sen tuottavuutta pyritään nostamaan ohjelmallisesti.

Lähdeaineistoa tähän kirjaan etsiessäni havahduin myös siihen, ettei varsinaisesti tästä aiheesta ole juurikaan suomenkielellä kirjoitettu. Pääasiallisesti tässä kirjassa lähteinä toimiikin yksittäisistä ohjelmista kirjoitetut käyttöohjeet ja oppikirjat, erilaiset vanhemmat kirjat liittyen tuotantoautomaatioon ja standardointeihin, sekä muutama vanhempi insinöörityö, jotka noudattavat mallia, jonka halusin hylätä.

Tähän kirjaan on jäsennelty ajatuksia sillä tavoitteella, että lukija, joka ei ole tietoinen suunnittelutyön automatisoinnin mahdollisuuksista, ymmärtää mahdollisuudet sekä osaa sovittaa näitä mahdollisuuksia omiin tuotteisiinsa ja prosesseihinsa.

1.3 Tutkimusote

Lukija saa ratkaista pitäisikö tämä kirja määritellä tutkimustyön tulosraportiksi vai ei. Aihetta ja työn tuloksia voidaan kuitenkin tulkita konstruktiivisen tutkimusotteen kautta. Konstruktiivisen tutkimusotteen oleellisia kohtia on, että aihe liittyy tosielämän ongelmiin ja että se koetaan tarpeelliseksi ratkaista. Lopputuloksena pyritään tuottamaan innovatiivinen konstruktio, joka ratkaisee alkuperäisen ongelman tosielämän metodein.

Konstruktiivinen tutkimusote vaatii tutkijan ja käytännön edustajien olevan joko yhtä, tai vaihtoehtoisesti riittävän lähellä toisiaan, molempien kuuluu oppia toisiltaan. Tämän opinnäytetyön tapauksessa tutkimusongelmani oli kutakuinkin ”Minkälaisia mahdollisuuksia ja työkaluja on suunnittelutyön automatisoinnilla?”. /1/

2 OHJELMAT

2.1 Yleiset työkalut

Työkalut, joita automaation toteutuksessa käytetään, voidaan jakaa kahteen ryhmään. Ne ohjelmat, jotka vaativat ohjelmointitaitoa ja ne, jotka eivät vaadi. Molemmissa on omat mahdollisuutensa ja toisaalta myös rajoitteensa. Kun automatisointiprojektiin ryhdytään, on käytettävien työkalujen valinta melko oleellinen asia. Valintaan vaikuttaa mm. käytössä oleva suunnitteluohjelma, henkilöstön osaaminen, ohjelmointitaidot ja projektin laajuus. Eri suunnitteluohjelmilla on omia kytköksiään muihin työkaluihin ja tätä kautta mahdollisuuksia tai rajoituksia. Ohjelmointitaito tai sen puute rajoittaa myös työkalujen käyttöä. Ohjelmointitaidon puute tulee oleelliseksi silloin, kun valmiiden työkalujen valmiit toiminnot loppuvat kesken tai niiden rajoitukset tulevat määrääviksi.

Työkalujen valinta ja tarve tulisi hahmottaa projektin tarpeiden mukaan. Molemmilla on toisiaan määrittelevä vaikutus. Ohjelmointitaidon tarpeen määrittely voi olla vaikeaa, joskin sen olemassaolosta on automaattisesti hyötyä.

Työkalujen valintaa voisi selvittää seuraavilla ajatuksilla:

1) Määritä tuotteen tai prosessin haluttu automatisoinnin taso.
2) Valitse käytettävä CAD-ohjelma, selvitä sen rajoitukset.
3) Tutki valmiiden kolmannen osapuolen sovelluksien mahdollisuudet ja vaihtoehtona tälle henkilöstön ohjelmointitaidot.
4) Pohdi käyttöliittymälle asetetut vaatimukset.
5) Punnitse projektin ulkoistamisen hyötyjä ja haittoja.

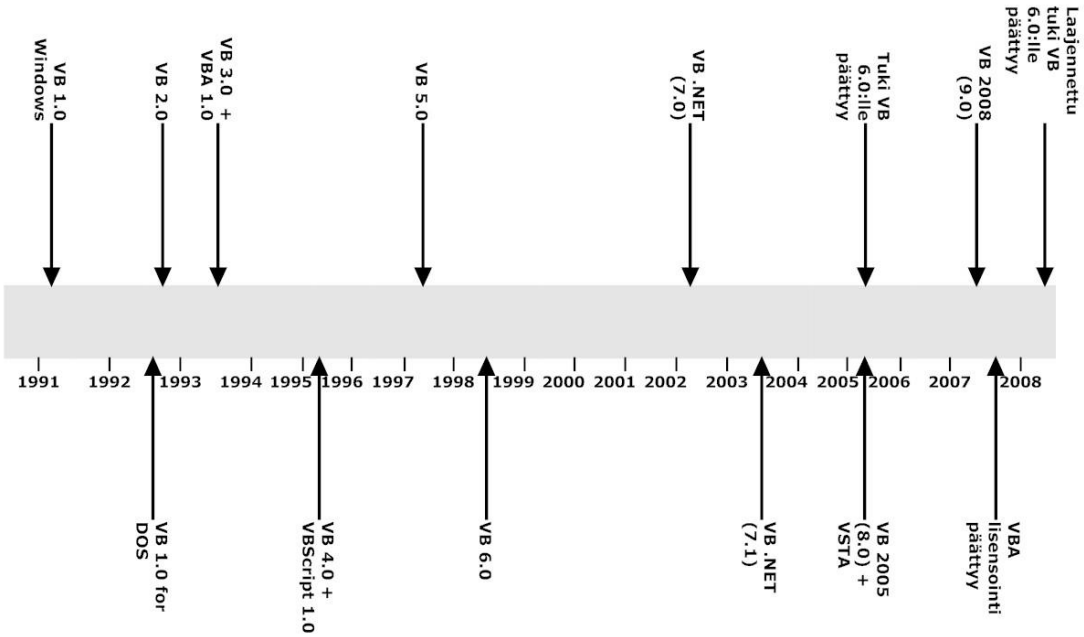
Microsoft Visual Basic for Applications (VBA)

Visual Basic for Applications, myöhemmin tässä dokumentissa terminä VBA, on Visual Basic ohjelmointikielestä tehty lisensoitava tekniikka, jota Microsoft on myynyt vuodesta 1993 alkaen. VBA tarjoaa Visual Basic 6 (VB) ohjelmointikielen, integroitavan käyttöliittymän ja ohjelmointiympäristön. Käytännön erona VB:n ja VBA:n välillä on itsenäisesti toimivan sovelluksen tekomahdollisuus, VBA-ohjelma

on aina sidottu johonkin toiseen ohjelmaan, kun taas pelkällä Visual Basicilla tehty sovellus voi toimia itsenäisesti. Kuvassa yksi esitetään koko Visual Basic perheen kehitystä aikajanalla.

VBA-rajapinta on oleellinen suunnitteluautomaatiossa, sen yleisyyden takia cad-ohjelmissa, sekä Microsoftin omissa tuotteissa kuten Excelissä. Suunnitteluohjelman ja VBA:n välinen rajapinta on toteutettu COM-objektilla, joka on ulkopuolisen toimittajan tekemä paketti, jota hyödynnetään ohjelmointikielessä. Suunnitteluohjelmistojen tapauksissa COM-objektin on tehnyt suunnitteluohjelmiston toimittaja. Tällä tekniikalla on yhdistetty kolmannen osapuolen ohjelman toiminnot helposti ohjelmointiympäristön kautta käytettäväksi. Eri ohjelmissa on tarjottu eriasteisesti ohjelman omia toimintoja, Microsoftin omissa ohjelmissa lähes kaikki toiminnot on käytettävissä myös ohjelmointiympäristön kautta, mutta suunnitteluohjelmissa on enemmän rajoitteita. Käytännössä siis aivan kaikkea, jota voidaan ohjelmassa tehdä manuaalisesti, ei voida tehdä ohjelmallisesti.

/2/



Kuva 1, Aikajana Microsoft Visual Basicin kehityksestä.

Microsoft Visual Studio for Applications (VSTA)

VSTA-paketin tulisi edustaa nykypäivää jo tätä kirjoittaessa eikä tulevaisuutta, mutta kuitenkin Microsoftin suunnitelmat siirtyä VBA:sta eteenpäin nykyaikaisempiin tekniikoihin ei ole edennyt niin nopeasti kuin olisi voinut kuvitella. VSTA edustaa tämän päivän ohjelmointikielistä tehtyä vastaavaa pakettia, jollainen VBA on. Visual Basicin viimeinen varsinainen versio on julkaistu vuonna 1998, VBA-paketin lisensointi on loppunut myös jo vuonna 2007. Vanhat ohjelmistot saavat käyttää edelleen olemassa olevaa lisenssiään ja jakaa ohjelmistojensa kanssa VBA-pakettia.

On selvää kuitenkin, että jollain aikavälillä VSTA tulee korvaamaan lopullisesti edeltäjänsä. Loppukäyttäjän kannalta tällä ei ole merkitystä, sovelluskehittäjälle tällä on merkitystä, koska vanhojen ohjelmistojen lähdekoodit eivät ole suoraan yhteensopivia uusien tekniikoiden kanssa. Siksi on suositeltavaa uusissa projekteissa heti hylätä vanhentunut tekniikka, vaikka sen käyttö voisi olla houkuttelevaa osaamisen tai olemassa olevan koodin kannalta.

VSTA-paketti sisältää tuen .NET rajapinnan ohjelmointikielille. Uudemmilla ohjelmointitekniikoilla ei ole niin suurta hyötyä suunnitteluautomaatiolle kuin on yleiselle ohjelmistotuotannolle. Pääsääntöisesti näissä sovelluksissa ei tarvita niin monitahoisia tekniikoita, vaan hyvin tavallisilla toiminto- ja komentorakenteilla saadaan aikaan suuriakin kokonaisuuksia.

Microsoft Excel

Excel on tehokas työkalu, joka on jokaiselle suunnittelijalle varmasti tuttu työkalu monistakin erilaisista ympäristöistä. Sen tarjoamia mahdollisuuksia suunnitteluautomaatiolla on monenlaisia. Suunnitteluautomaatiossa tulee helposti vastaan tilanteita, joissa suunnitteluohjelman omat työkalut tuntuvat rajoittuneilta tai niiden hallittavuus ja käytettävyys alkaa heikentyä sovelluksen laajentuessa. Excel tarjoaa hyvän vaihtoehdon tehdä asioita irrallaan suunnitteluohjelmasta.

Käytännön esimerkkeinä voisi olla asioiden riippuvuuksien hoito irrallaan suunnitteluohjelmasta. Käyttäjän valinta voi johtaa erilaisiin vaihtoehtoihin, vaihtoehdot taas voivat riippua muista vaihtoehdoista, ja tämä ketju voi jatkua pitkälle. Excel on myös tehokas työkalu matemaattisissa toiminnoissa, esimerkkinä vaikka klassisen lujuuslaskennan yhdistäminen automaatioon. Lopputuotteeseen asti vaikuttava mitoitus voi määräytyä käyttäjän antamista parametreista. Ohjelmiston tehtävänä voisi tässä välissä olla laskelmien teko tarvittavista asioista.

Taulukkolaskentaominaisuudet ja Excelin omat käskyt ovat tehokkaita apuvälineitä automaatin rakentamisessa. Ohjelmointityöstä jää myös jokusia vaiheita pois, jos hyödynnetään Exceliä, esimerkiksi ohjelman käyttöliittymä voidaan suoraan rakentaa laskentataulukkoon.

Käyttöliittymä

Kaikilla sovelluksilla on jonkunlainen käyttöliittymä. Yksinkertaisimmillaan automaatin käyttöliittymä on pikakuvake tai yksittäisessä valikossa oleva kohta, joka käynnistää halutun toiminnon tai toimintoketjun. Yksinkertaisimmillaan käyttöliittymä on vain tapa käynnistää ohjelma, kaikissa sovelluksissa ei ole tarvetta nähdä suoritusta tai hallita käyttäjän toimesta sitä. Toisessa ääripäässä käyttöliittymää edustaa itsenäistä sovellusta, joka on räätälöity vain ja ainoastaan yhteen käyttötarkoitukseen.

Käyttöliittymän tarpeen määrittely kulkee suoraviivaisesti automatisoinnin tason kanssa käsi kädessä. Ohjelman ollessa yksinkertainen tai hyvin suoraviivainen ei käyttöliittymänkään tarvitse olla monipuolinen. Kokonaisuuksia ohjatessa on käyttöliittymän merkitys ohjelman käyttöä ajatellen jo melko oleellinen asia, huono käyttöliittymä voi pilata hyvänkin ohjelman.

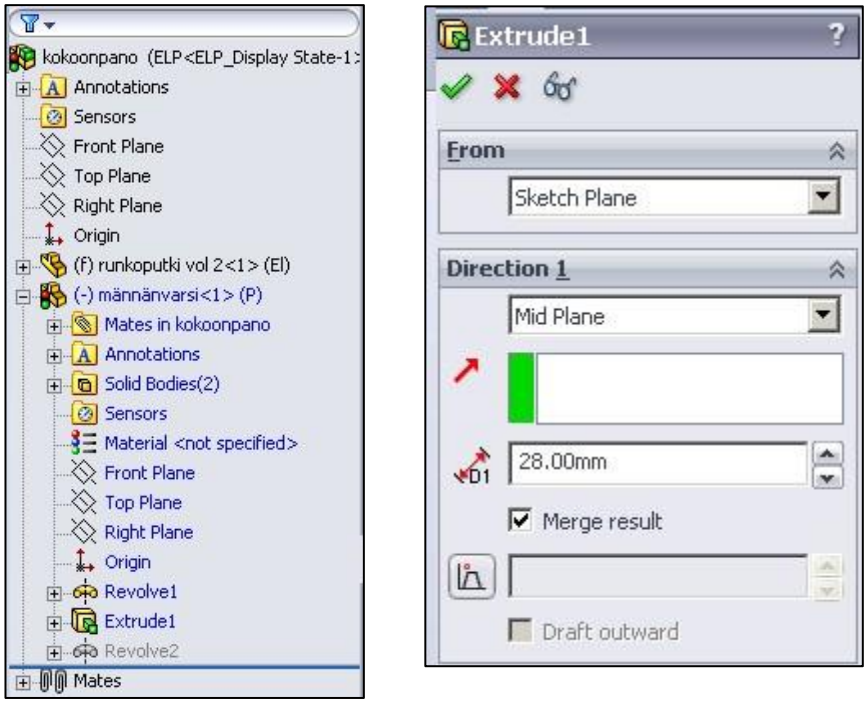
Käyttöliittymä on myös mahdollista integroida suunnitteluohjelman sisään joissain ohjelmissa. Tällöin ohjelmaa voidaan ajaa aivan samoilla tavoilla kuin suunnitteluohjelmassa mitä tahansa muutakin toimintoa. Excelin käyttöä käyttöliittymänä kannattaa harkita useissa tapauksissa, sen mukana saadut muut hyödyt ja ennen kaikkea

helppo ylläpidettävyys ja päivitettävyys ovat aivan omaa luokkaansa. Liitteessä 1 on esimerkkikuva Excelillä tehdystä suunnitteluautomaation käyttöliittymästä.

2.2 SolidWorks

SolidWorks on Dassaultin kehittämä 3d-pohjainen cad-ohjelma. Ensimmäinen julkaisu oli vuonna 1995, ja tästä eteenpäin versioita on tullut vuosittain. Ohjelman mallinnustekniikka on nykypäivän cad-ohjelmille tyypillinen parametrinen piirremallinnus. Vastaavia kilpailijoita samoilla markkinoilla ovat Pro/Engineer, Catia, Solid Edge ja Autodesk Inventor. Näihin ohjelmiin ei paneuduta tässä työssä, mutta todettakoon niistä, että ovat mahdollisuuksiltaan ja ominaisuuksiltaan hyvin lähellä SolidWorksia.

Piirrepohjaisuus tarkoittaa mallin rakentamista piirre kerrallaan, ensin luodaan peruspiirre, jota aletaan muokata uusilla piirteillä. Parametrisuus taas kertoo, että piirteeseen tai malliin liittyvien mittojen tai muiden attribuuttien hallinta on mahdollista luomisen jälkeenkin. Molemmat ominaisuudet yhdessä tekevät 3d-suunnittelusta niin tehokasta kuin se parhaimmillaan on. Kuvassa kaksi esitetään käytännön esimerkit piirrepuusta ja parametrisuudesta.



Kuva 2, SolidWorksin piirrepuu vasemmalla ja oikealla pursotuksen tietoikkuna.

SW tarjoaa erittäin kattavan ja tehokkaan API-rajapinnan VBA:n kautta käytettäväksi. Rajapinnan läpi on mahdollista tehdä sisäisiä integroituja käyttöliittymiä, sekä ulkoisia liityntöjä muihin ohjelmiin ja tätä kautta käyttöliittymiä muilla ohjelmilla. Esimerkkinä Excelin kautta voidaan hallita suoraan SW:tä.

Makro apuohjelmia voidaan nauhoittaa erityisellä toiminnolla tai kirjoittaa käsin ohjelmakoodia. Makrojen nauhoitus on melko helppo tapa tehdä pohja apuohjelmalle, jota sitten muokataan käsin haluttuun suuntaan. Muutama uusin versio ohjelmasta tarjoaa tuen uusille .NET-pohjaisille ohjelmointikielille, mutta myös VBA-tekniikka on tuettuna edelleen uusimmissakin versioissa.

Luvussa kuusi esitetään kaksi case-esimerkkiä, jotka on tehty SolidWorksille ja kirjoitettu VBA-kielellä. Ensimmäisessä esimerkissä esitetään ulkoista käyttöliittymää ja kokoonpanon käsittelyä. Toinen esimerkki on levysuunnittelijalle tehty makro, joilla pyritään nopeuttamaan suunnittelijan rutiineja.

Ohjelman yleisyyden takia internetistä löytää melko paljon valmiita makroja ja laajojakin valmiita apuohjelmia työn helpottamiseen, myös malliesimerkkejä on esitelty useilla internet-sivuilla. Ohjelman omat tutoriaalit ja API-dokumentaatio ovat melko laajoja sekä sisältävät myös paljon malliesimerkkejä.

/3/

2.3 Vertex G4

Vertex on suomalainen suunnitteluohjelmisto, jolla on pitkä historia alkaen jo vuodesta 1977. Tuoteperheeseen kuuluu erilaisia variaatioita ohjelmasta, näitä on mm. mekaniikkasuunnitteluversio, sähkö-, laitos- ja rakennusversiot. G4-versio on mekaniikkasuunnitteluun tarkoitettu versio.

Mekaniikkaversio Vertexistä on hyvin samankaltainen toiminnoiltaan ja käyttöominaisuuksiltaan kuin vastaavat SolidWorksin versiot. Syy miksi Vertex on esitelty täs-

sä dokumentissa erikseen, on sen kotimaisuus ja hyvin erilainen lähestymistapa suunnitteluautomaatioon.

Vertexissä ei ole niin pitkälle kehitettyä API-rajapintaa ohjelmoijia varten kuin SW:ssä, mutta siinä on yksisuuntainen käskykieli, jolla voidaan ohjata ohjelman toimintoja. Yksisuuntaisella tarkoitetaan sitä, että ohjelma ottaa käskyjä vastaan mutta ei anna tietoa takaisin esimerkiksi käyttöliittymän suuntaan, tämä asettaa tietynlaisia rajoituksia. Automaattien toiminta on myös rajattu hyvin kaavamaiseksi, esimerkkinä käyttöliittymän ja automaatin suoritusjärjestys. Ohjelma aloittaa käskytiedoston ajamisen vasta, kun käyttöliittymä on suljettu, ellei käytetä vaihtoehtoja, joissa ohjelmaa ja automaattia ei ole enää sidottu ollenkaan toisiinsa.

Vertex on itse tuotteistanut automaattituotteet hyvin pitkälle, se tarjoaakin avaimet käteen -ratkaisuja. Tämä on suuri ero siihen, että SW:n tapauksessa vastaava palvelu on lähinnä jälleenmyyjillä. Vertex toimittaa automaattitoimintoja kolmella eri tuotenimikkeellä, alla lyhyet kuvaukset niistä.

Presto

Presto on perustason automaattituote, joka tukee vain sisäisiä käyttöliittymiä. Tarkoittaa siis, että automaatti on ajettava auki olevasta mallista ohjelman sisältä. Presto sisältää pääasialliset käskyt mallien, kokoonpanojen ja niiden tuotetietojen hallintaan. Presto on tarkoitettu lähinnä yksittäisen suunnittelijan työkaluksi yksittäisten automaattien tekoon. Tuotteen mukana tulee valmis Excel-taulukko, jota voi hyödyntää käyttöliittymänä.

Tempo

Tempo on automaatti, joka tukee ulkoisia käyttöliittymiä, ja sitä on mahdollista ajaa verkkoympäristöstä ilman, että työasemalla on edes perusohjelmistoa asennettuna. Paketin myötä tulee muutamia lisäkäskyjä tuotetiedon hallintaan. Kohderyhmänä tälle paketille on tarjoussuunnittelu ja myyntihenkilöstö.

Forte

Forte on automaattituotesarjan raskain ilmentymä. Siinä tulee vielä laajemmat verkkokäyttömahdollisuudet sekä jopa www-selain pohjaiset käyttöliittymä vaihtoehdot. Fortella pystytään automatisoimaan ja hallitsemaan suuria kokonaisuuksia kaikkien kohderyhmien käyttöön.

Käskytiedoston kieli

Käskytiedosto, jonka perusteella Vertex suorittaa automaatiota, on puhdas tekstimuotoinen tiedosto, joka voidaan kirjoittaa periaatteessa millä tahansa käyttöliittymällä, esimerkiksi puhtaasti käsin kirjoittamalla muistiolla. Liitteessä 3 esitetään malliesimerkki käskytiedoston rakenteesta, esimerkki on kommentoituna esitetty. Toiminnot kielessä rajoittuvat lähinnä konfiguraattorin ja kokonaisuuden hallintaan sekä jälkikäsittelyvaiheeseen eli mallin visualisointiin ja piirustuksien laadintaan ja koko projektin tiedonhallintaan. Vertexillä ei voi hallita automaattisesti alkeistoimintoja, esimerkiksi sillä ei voi piirtää viivoja tai kustomoida ohjelmistoa.

/4/

2.4 Autocad

Autocad on Autodeskin kehittämä ohjelmisto, jonka ensimmäinen versio julkaistiin jo vuonna 1982. Autocadin parasta aikakautta edusti 90-luvun alku, jolloin mm. sen tiedostoformaateista dxf ja dwg muodostui alan de facto -standardit. Autocad esitellään tässä kirjassa kolmantena erilaisena esimerkkinä ja mahdollisuuksien tarjoajana suunnitteluautomaatiolle. Tässä esitellään vain toiminnot kaksiulotteisen tekniseen piirtämiseen liittyen. Vaikka kaksiulotteiset ohjelmat ovat menettäneet tällä vuosikymmenellä markkinoita kolmiulotteisille ohjelmille, ei pidä väheksyä Autocadin asemaa markkinoilla. Missään tapauksessa ei myöskään saa ajatella, että Autocadilla ei pystyisi tekemään samanlaisia automaatioita kuin sen kolmiulotteisilla kilpailijoilla.

Lisp

Autocadin versiossa 2.18, joka julkaistiin jo vuonna 1986, esiteltiin AutoLISP ohjelmointiympäristö. Tämän ympäristön kehitys jatkui aina vuoteen 1995 asti, jolloin kehityssuunta kääntyi graafisiin ympäristöihin kuten VBA ja edelleen .NET arkkitehtuureihin. Vuonna 2000 AutoLISP korvattiin uudemmalla VisualLISP paketilla, joka lisäsi LISPin käyttömahdollisuuksia Autocadin sisällä. Kehitystrendi jatkuu kuitenkin edelleen uudemmilla graafisilla ympäristöillä.

Lisp itsessään on ohjelmointikieli, joka esiteltiin jo vuonna 1958. Se edustaa toiseksi vanhinta sukupolvea korkeatasoisista ohjelmointikielistä. Kielen rakenne ja operaatiot perustuvat funktionaaliseen lähestymistapaan ja täten eroavat paljon yleisimmistä tämän päivän ohjelmointikielistä kuten Basic, C ja Java, jotka ovat proseduraalisia. Lisp on tulkattava kieli, eli sillä tehtyjä ohjelmia ei tarvitse erikseen kääntää, tulkki on sisäänrakennettu Autocadiin.

Lispin integroimiselle Autocadiin jo sen hyvin varhaisessa versiossa on johtanut suureen määrään apuohjelmia ja kaupallisia kolmannen sektorin sovelluksia, jotka tehostavat suunnittelutyötä. Lisp on tehokas ja looginen työkalu, mutta sen tehokas hyödyntäminen vaatii ohjelmointitaitoa. Lispiin voidaan yhdistää DCL-kielellä tehtäviä graafisia elementtejä. Siinä ei ole kaikkia nykyaikaisen käyttöjärjestelmän mahdollistamia graafisia elementtejä, vaan ainoastaan yleiset elementit käyttöliittymien tekemistä varten. 2007 versiosta alkaen Lisp ja VBA/.NET rajapinnat ovat pystyneet keskustelemaan tehokkaammin keskenään, tekniikoita voidaan yhdistää esimerkiksi kutsumalla graafisia osia, jotka on tehty VBA:lla, Lisp-koodista.

Makrot

Autocadissa on useita erilaisia lähestymistapoja makrojen tekoon. Ohjelman omia komentoriviltä suoritettavia käskyjä voidaan ketjuttaa skripteiksi, joita on mahdollista ajaa sellaisenaan. Näillä voidaan tehdä laajojakin rutiinitöitä, yksinkertaisin tapa muodosta skripti on kirjoittaa käskyt määrätyssä järjestyksessä tiedostoon ja tallentaa se scr-päätteellä. Skriptien ajaminen tapahtuu käskyllä *script*. Skriptejä ja AutoLISPiä voidaan yhdistellä myös, näin laajennetaan käytössä olevia mahdollisuuksia.

Makroja voidaan nauhoittaa myös *Action Recorder* –toiminnolla, sillä seurataan tarkasti käyttäjän tekemiä toimia ja nauhoitetaan nämä siinä järjestyksessä. Tämä tekniikka ei kirjoita varsinaista skriptiä eikä myöskään VBA/.NET-koodia, vaan on täysin oma tekniikkansa. Action recorder -toiminto esiteltiin Autocadin versiossa 2009.

ObjectARX

ObjectARX on nimitys, jolla kutsutaan Autocadin API-rajapintaa. Paketti ei tule automaattisesti Autocadin asennuksen mukana, mutta sitä jaetaan ilmaiseksi internetin välityksellä. Rajapinta mahdollistaa koodin kirjoittamisen C++, C# ja VB.NET kielillä. Valmiista ohjelmista tehdään DLL-tiedostoja, joita hyödynnetään Autocadissa. Ohjelmointiympäristönä toimii Microsoftin Visual Studio .NET, ObjectARX paketin versioilla sekä Visual Studion versioilla on riippuvuuksia. Myös käytetyn rajapinnan versiolla on riippuvuuksia toimivuuteen Autocadissa.

VBA

Kuten Office-paketissa ja SolidWorksissa on Autocadissa VBA-paketti integroituna aivan samalla tavalla. Sitä voidaan käyttää suoraan ohjelman sisältä, ja sillä voidaan hallita ohjelmaa todella tehokkaasti ja lähestulkoon rajattomasti.

Blokit

Autocadin symbolit, tai blokit, on syytä myös mainita erikseen automatisoinnissa. Versiossa 2006 esiteltiin dynaamiset blokit, jotka tarkoittavat, että blokkiin voidaan luoda mittasidonnaista järkeä. Voit luoda taulukoituja blokkeja, joissa on samasta perusmallista monta erilaista variaatiota, tai esimerkiksi blokin asetuskulma voi määräytyä automaattisesti jonkun jo olemassa olevan elementin perusteella. /5/

2.5 Kolmannen sektorin ohjelmat

Kolmannella sektorilla tarkoitetaan tahoa, joka ei hallinnoi tuotetta tai prosessia jonka suunnittelua ollaan automatisoimassa, eikä suunnitteluohjelmiston valmistajaa.

Kolmannesta sektorista oli jo muutama viittaus aiemmissa kappaleissa, todettakoon vielä, että kyseessä on aivan oma teollisuuden ala, joka on muodostunut cad-ohjelmien ympärille. Tämä ala tuottaa lisäarvoa ja lisäarvoa tuottavia palveluita cad-ohjelmiston peruskäyttäjille. Pääasiallisesti cad-ohjelmien tekijän eivät ole kiinnostuneita näistä markkinoista, vaan kolmas sektori on hajautunut useiksi pieniksi tahoiksi.

Kolmas sektori tarjoaa monenlaisia asiantuntija- ja konsultointipalveluita. Usein yrityksellä on tarjottavana värikäs kirjo erilaisia palveluita sekä myös tuotteistettuja ohjelmistoja.

Yleisimmät tarjotut ohjelmistot ovat tiedonhallintaan tai tuotekonfiguraatioihin liittyviä. Tiedonhallinnalla tai sen automatisoinnilla on oma erityinen roolinsa yrityksen sisäisessä työskentelyssä. Kuitenkin tiedon ja projektien hallinta on useimmiten sellainen asia, johon on jo paneuduttu tai se on järjestetty jotenkin. Tuotekonfiguraattorit taasen ovat sellaisia yleispäteviä ohjelmia, joilla voidaan hallita tuotetta automaation eri tasoilla. Konfiguraattorit mahdollistavat tuotesarjojen teon, attribuuttihallinnan, piirustusten teon ja monet muut rutiinityöt. Parhaimmillaan tällaiset ohjelmistot ovat vakioitujen tuotteiden suunnittelun räätälöintivaiheessa ja tuotteen hallinnassa, jolloin automatisoidaan mitoitukset ja jälkikäsittelyvaiheet. Monet konfiguraattorityyppiset ohjelmat käyttävät apunaan Exceliä, niin käyttöliittymänä kuin matemaattisena apuvälineenä.

Moniin cad-ohjelmiin tarjotaan myös pitkälle räätälöityjä erikoissovelluksia, osa valmistajista jakaa oman tuotteensa ympärille liittyviä apuohjelmia, jotka voidaan integroida suoraan ohjelmaan. Esimerkkinä Anstarin tuottama Autocad-laajennus, joka helpottaa suunnittelua, jossa käytetään Anstarin tuotteita. Laajennus lisää ohjelmaan valikon, jonka avulla pystytään lisäämään symboleita piirustukseen.

SolidWorksin internet-sivuilla on erityinen haku kumppaniohjelmille, haku löytää yli 350 sovellusta tai tuotetta, jotka liittyvät itse päätuotteeseen mutta ovat kolmannen sektorin tekemiä.

3 AUTOMATISOINNIN ERI TAVAT

3.1 Makrot

Makroihin viitattiin jo ohjelmien esittelyissä useaan otteeseen, nyt lienee syytä määritellä hieman tarkemmin muutama ohjelmointiin liittyvä termi.

Eräs tietotekniikasta paljon kirjoittanut kirjailija määritteli algoritmin seuraavasti. J.G Brookshear ”Tarkasti ottaen algoritmi on äärellinen joukko täsmällisiä, suoritettavissa olevia ohjeita, jotka ohjaavat päättyvää tehtävän suoritusta” /10/

Algoritmin ei yleisesti ajatella olevan sidottu ohjelmointiin, eikä missään tapauksessa sidota mihinkään ohjelmointikieleen. Algoritmi on siis vain yleismaailmallinen ohje tai resepti jonkun asian suorittamiselle, käytännön toteutus voi erota tästä.

Miten makro sitten eroaa algoritmista? Makro edustaa omaa lajiaan tietokoneohjelmien maailmassa, sen ero yksittäiseen algoritmiin tulee siinä, että makro on sidottu johonkin alustaan tai ohjelmaan. Ohjelmiin sisäänrakennetut pikanäppäin toiminnotkin ovat omalla tapaa makroja, ja ennen kaikkea niillä toteutetaan makron ydintarkoitusta, helpotetaan käyttäjän työskentelyä.

Makro voi koostua siis eri tavoin määritetyistä käskyistä tai komennoista, jotka suoritetaan yhdellä kerralla. Pääasiallisesti makron suoritus on nopeampi tapa tehdä määrätyt asiat kuin ne yksi kerrallaan manuaalisesti tehden. Makrojen kohderyhmä onkin yleensä yksittäinen suunnittelija tai useampi suunnittelija, jos heidän työkalunsa tai työnsä on riittävän lähellä toisiaan. Usein makrojen tarve havaitaan, kun tiettyjä asioita toistetaan päivästä toiseen tai ainakin riittävän säännöllisin väliajoin.

Esimerkkejä makroista:

- Viivan koon automaattinen muutos piirustuksissa,
- piirustuksien tallentaminen haluttuun tiedostomuotoon,
- määrätyn asian käsittely yhdellä kertaa useaan tiedostoon,

- toiminto, joka sulkee kaikki auki olevat tiedostot ilman tallentamista ja sulkee ohjelman tämän jälkeen ja
- määrättyjen käskyjen lyhentäminen helpommin käytettäviksi komentoriviltä, esimerkiksi zoom 0.5x voisi olla vain zo, ja zoom 2x voisi olla zi.

Tässä vain muutamia esimerkkejä makroista, jotka voisivat helpottaa monen suunnittelijan arkirutiineja.

Makrojen tekoon on kaksi pääasiallista keinoa. Ensimmäinen on kirjoittaa skripti tai koodi käsin ja sen jälkeen suorittaa muodostettua ohjelman pätkää. Toinen vaihtoehto on nauhoittaa käyttäjän tekemät toiminnot ja tällöin ohjelman annetaan hoitaa koodin kirjoitus. Tapoja voidaan myös tehokkaasti yhdistää, tietyissä ohjelmissa voidaan nauhoittaa makroja niin, että niistä saadaan puhtaat ohjelmointikieleen sidotut lähdekoodit esille, ja näin ollen nauhoitettua makroa voidaan käyttää pohjana, josta muokataan haluttu makro.

3.2 Konfigurointi

Jukka Korpela määrittelee pienehkössä sivistyssanakirjassaan sanan konfigurointi seuraavasti:

> ”Tehdä valinnat vaihtoehtojen välillä esimerkiksi laitteistoa tilattaessa tai ohjelman asetuksissa. Laitteiston konfigurointi tarkoittaa ennen muuta sen päättämistä, mitä laitteita siihen hankitaan, esim. hankitaanko tietokoneeseen kirjoitin ja jos niin millainen. Ohjelman konfigurointi tehdään (ja on ehkä pakkokin tehdä) ohjelmaa tietokoneeseen asennettaessa, mutta useimmiten asetuksia voi myöhemmin muuttaa eli ohjelma voidaan konfiguroida uudelleen.” /6/

Suunnitteluautomaatioon viittaavissa lähteissä käytetään yleisesti termiä konfiguraattori, jolla automatisoidaan tuotteen tilauksen, suunnittelun tai jälkikäsittelyn eri vaiheita. Vastaavia työkaluja käytetään monella muullakin alalla, uutta tietokonetta tilatessa asiakas tekee tilauksen internetissä ja ohjelma ohjaa häntä tekemään sopivia

valintoja, eli asiakas tilaa haluamansa konfiguraation tietyistä vaihtoehdoista kooten. Autokaupassa toimitaan aivan samalla tavalla, uusi auto konfiguroidaan asiakkaan toivomusten mukaan vastaavalla työkalulla, ja tilaustieto kulkee sähköisesti aina tehtaalle asti.

Tietokoneavusteista suunnittelua voidaan konfiguroida ohjelmistojen omilla toimenpiteillä melko tehokkaasti, nämä tavat ovat taulukkolaskentatyyppisiä eivätkä vaadi ohjelmointitaitoja. SW:ssä konfiguraatioita voidaan hallita sellaisinaan manuaalisesti tai *design table* nimisen taulukon avulla, Vertexissä konfiguraatioita kutsutaan *ilmiasuiksi* ja taulukkoa, johon voidaan luoda mm. konfiguraatioita mitoituksista, *mittataulukoksi*. Tyypillisiä esimerkkejä ohjelman sisäisistä konfiguraatioita hyödyntävistä malleista on standardikomponentit, esimerkiksi tiettyä standardia noudattavat pultit on rakennettu yhteen malliin, ja mittoja ohjataan konfiguraatioiden avulla. Liitteessä 4 on esitetty esimerkki SolidWorksin design tablesta, jossa on erilaisia konfiguraatioita tuotteesta.

Suunnitteluautomaatio yhdistyy konfigurointiin sillä, että edellä mainittujen konfiguraatioiden tekoa ja hallintaa hoidetaan jollain siihen erikseen tehdyllä ohjelmistolla. Kolmannen sektorin konfiguraattoriohjelmat on yleensä tehty melko älykkäiksi ja helppokäyttöisiksi. Valmiissa tuotteissa on kuitenkin vain vakioitu määrä toimintoja, ja aina nämä eivät ole riittäviä. Konfiguraattorit myös hoitavat tuotteen jälkikäsittelyä erinäisin integroiduin makroin. Valmiiden ohjelmistojen suurin hyöty on siinä, etteivät ne vaadi ohjelmointitaitoa toimiakseen. Yrityksen sisäisiä räätälöityjä toimintoja pystyy rakentamaan ilman valmiita ohjelmistoja, jos henkilöstön osaaminen sen mahdollistaa. Kolmannen sektorin ohjelmat ovat melko tehokas tapa ostaa tätä tietotaitoa.

Konfiguraattorin yksinkertaistettu toimintaperiaate on seuraavanlainen:

- Konfiguraattori linkitetään malliin sekä siihen liittyviin dokumentteihin.
- Käyttöliittymän avulla annetaan lähtötiedot halutusta lopputuloksesta.
- Suoritetaan automaatti.
- Tarkastellaan lopputulosta ja tehdään tarvittavia manuaalisia toimintoja.

Konfiguroinnissa perusperiaatteena on olemassa oleva suunniteltu malli, tätä ohjataan ennalta määriteltyjen ehtojen perusteella haluttuun suuntaan. Tuotteelle asetetuista vaatimuksista on kirjoitettu tarkemmin luvussa 4. Luvussa 6 esiteltävä ensimmäinen esimerkki on eräänlainen mallisarjan tuottava konfiguraattori, siinä hallitaan hydraulisylinteri mallisarjaa automaattisesti ohjelmalla. Autocadissa ja teknisessä piirtämisessä konfiguraattorien toimintaperiaate on erilainen, ne perustuvat yleensä blokkien tekoon, ohjelmalle niiden opettamiseen ja niistä koottavaan piirustukseen. Esimerkkinä sähkökaavio, joka muodostetaan ennalta tunnetuista blokeista. 3D-pohjaisissa ohjelmissa pyritään päinvastaiseen toimintaan, eli tehdään valmis malli mahdollisimman valmiiksi, josta muokataan haluttu konfiguraatio.

Monet konfiguraattorin yksittäiset ominaisuudet ovat makroja, konfiguraattorin voidaankin ajatella koostuvan osittain isosta osasta erilaisia makroja, joista on tehty älykkäitä. Tällaisia makroja esimerkiksi ovat piirustuksien käsittely, piirustuksien luonti, tiedostojen tallennus haluttuun formaattiin jne.

3.3 Kokonaisuudet

Kokonaisuus laajentaa konfiguraattoria edelleen. Kokonaisautomaatiossa järjestelmään linkitetään muitakin henkilöstöryhmiä kuin suunnittelijoita. Järjestelmä suunnitellaan myös valmistus-, tarjous- ja myyntihenkilöstön työkaluksi, tällöin sovellusta voidaan hallita niin, ettei edes varsinaista suunnitteluohjelmistoa ole enää tietokoneella vaan toimintoja hoidetaan yli verkon. Isoja kokonaisuuksia käytetään mm. silloin, kun myyntihenkilöstö on hajautunut pitkin maailmaa, suunnitteluhenkilöstö on toisessa paikkaa ja valmistus tapahtuu vielä kolmannessa paikassa. Yhdellä järjestelmällä linkitetään nämä kaikki toisiinsa, ja samalla tuotteen suunnittelu automatisoidaan niin, että myyjältä päätyy valmistukseen asti suunnittelutieto.

Suunnittelujärjestelmän ja valmistustekniikan linkityksellä on huikeita mahdollisuuksia tehostaa tiedonvälitystä ja tuotannon tehoa. Pääasiallisesti valmistustekniikat on muutettu jo ajat sitten tietotekniikkaa hyödyntäviksi, monessa pienessä ja keskisuuressa konepajassa tieto näille pajan tietokoneille ei kuitenkaan kulje automaatti-

sesti suunnittelijan pöydältä. Ensimmäinen askel, kun järjestelmiä liitetään yhteen, on tiedonsiirto niiden välillä.

Aiemminkin viitattu tiedonhallinta- ja projektinhallintajärjestelmän liittäminen suunnittelujärjestelmään ja suunnitteluautomaattiin on myös yksi mahdollisuus. Automaatista voidaan siirtää tiedot tehdyistä malleista ja piirustuksista suoraan tiedonhallintajärjestelmään. Erilaisten järjestelmien keskenään linkitykset vaativat joko sen, että kaikki käytössä olevat ohjelmat on yhden toimittajan tekemiä tai sitten erillistä räätälöintiä. Kovin pienissä yrityksissä tällaiset järjestelmät käyvät liian isoiksi ja kankeiksi eivätkä palvele siinä tarkoituksessa mihin soveltuvat.

4 AUTOMATISOINNIN ASETTAMAT VAATIMUKSET

Tämän luvun kaksi ensimmäistä kappaletta ovat teoriapitoisia ollen hyvin pitkälti irti cad-ohjelmista. Kahdessa viimeisessä kappaleessa keskitytään konkreettisimmin asioihin, jotka vaikuttavat suunnitteluautomaation käytäntöön.

4.1 Standardisointi

”Standardi on toistuvaan tapaukseen tarkoitettu yhdenmukainen ratkaisu.” /7/

Niin yrityksen kuin tuotteen standardisointi on yritykselle tärkeä strateginen asia. Yritysstandardisointi tarkoittaa huomattavasti laajempaa menettelytapaa kuin pelkkä tuotteeseen kohdistuva standardisointi. Yrityksen standardisointi voi perustua ylikansallisiin tai kansallisiin standardeihin tai yrityksen omiin standardeihin. Normaalilla standardisoinnilla tavoitellaan hyötyä kaikille kohderyhmille, yrityksen sisäisillä toimilla tavoitellaan kuitenkin hyötyä lähinnä yritykselle itselleen, kuitenkin ilman, että loppukäyttäjä kärsii siitä.

Tuotteeseen liittyvä standardisointi voidaan jakaa kolmeen tasoon: /8/

- Alimmalla tasolla standardisoidaan materiaalit, tarvikkeet, suunnitteluohjeet sekä menetelmästandardit. Tähän tasoon lukeutuu pientarvikkeet kuten pultit ja mutterit, sekä materiaalivalinnat.
- Keskimmäisellä tasolla standardisoidaan komponentit sekä niiden valinta. Tällä tasolla määritetään tarkasti komponenttien laskenta- ja valintaperusteet sekä komponentit, jotka ostetaan ulkoa.
- Ylimmällä tasolla standardisoidaan tuote, josta syntyy tuotteen suunnittelustandardi.

Monessa yrityksessä tehdään kahden alimman tason standardisointia ilman, että siihen kiinnitetään minkäänlaista huomiota. Monille tarvikkeille on vakiotoimittajat, kuten myös materiaaleille. Myös komponenttien osalta käytetään niitä yleisesti hyviksi todettuja; tämä on piilossa tapahtuvaa standardisointia.

Ylin taso eli tuotestandardi on se taso, jolla on suunnittelun automatisoinnin kanssa merkitystä. Tuotestandardissa määritellään tuotteen fyysisiä rakenteita ja suoritusarvoja, ja kuvataan näille suunnitteluohjeisto. Tällä tasolla ei kuvata kaikkia tuotteita, jos tuotteiden jakauma ajatellaan Gaussin käyrälle. Standardoidut tuotteet ovat siinä keskellä, ja selkeästi räätälöidyt ja vähemmän kysytyt tuotteet ovat käyrän laitamilla.

Standardointiasteen mukaan tuotteet voidaan jakaa neljään kategoriaan: /8/

- Tasolla 1 on täysin määritellyt tuotteet, asiakkaalla on ainoastaan mahdollisuus valita haluamansa tuote. Tyypilliset kuluttajien kestohyödykkeet ovat tällaisia tuotteita, kulutuselektroniikka yms.
- Tasolla 2 on moduuleihin perustuvat tuotteet, konfiguraattorissakin käytetyt tietokoneet ja autot voivat kuulua tähän kategoriaan.
- Tasolla 3 on osittain standardoitu tuote, jossa on tietyt osat vakioituja mutta lopputuote räätälöidään vakioidun osan ympärille asiakkaan toiveiden mukaan. Räätälöinti voi olla osakonstruktion suunnittelua täysin tai vakio-osan parametrista suunnittelua.
- Tasolla 4 on täysin räätälöity tuote, jossa kuitenkin käytetään vakioituja komponentteja. Esimerkiksi painevesijärjestelmä, joka kootaan valmiista komponenteista täysin asiakkaan vaatimuksien perusteella, vaatii projektikohtaisen suunnittelun.

Standardisoinnin tarkoitus on määrätietoinen yrityksen toimintojen kehittäminen yrityksen kilpailukyvyn ylläpitämiseksi sekä kannattavuuden turvaamiseksi. Standardisointi pyrkii myös vakioimaan asioita, sen ei tule sekoittaa asioita eikä olla rajattoman joustava.

Liikaa ei saa innostua tästäkään, on olemassa selkeästi tietty määrä tuotetyyppejä, joita ei voi tai ei taloudellisesti kannata standardisoida. Yksittäiskappaleet, joita tehdään vuodessa vain muutama, eivät saavuta merkittävää hyötyä täydestä standardisoinnista. Näihinkin tuotteisiin voidaan soveltaa kevyemmin osin jotain osia prosessista. Myös sellaiset tuotteet, joissa joku ominaisuus on aina tapauskohtaisesti optimoitu, eivät sovellu hyvin automatisointiin, jos ominaisuus määrittää pääosan rakenteesta.

4.2 Modulointi

Modulointi on yksi standardisoinnin sovellus, jolla on kuitenkin hieman toisenlainen pyrkimys. Modulointi ei tavoittele tarjottavan tuotevalikoiman pienentämistä vaan pyritään luomaan sellainen tuotevalikoima, joka vastaa markkinoiden tarpeita rajaamalla konfiguroinnin mahdollisuudet vain tärkeimpiin ominaisuuksiin.

Modulaarinen tuoterakenne on järjestelmä, joka muodostuu ennalta määritetyistä osista, moduuleista. Moduuli on osa, jolla on tietty määrätty käyttö- tai sovellusalue ja rajapinta, josta se liittyy kokonaisuuteen tai toiseen moduuliin. Yksittäisistä moduuleista kootaan varsinainen lopputuote yhdistelemällä niitä asiakkaan vaatimuksien mukaan. Moduulien väliset rajapinnat on suunnittelunäkökulmasta katsottuna tärkeitä asioita, niiden avulla mahdollistetaan moduulien vaihdettavuus ja tuotteen varioinnin mahdollisuudet kasvavat. Yksittäinen moduuli voi olla alikokoonpano tai osa. Moduulien suunnittelu voidaan hajauttaa todella tehokkaasti, kunhan rajapinnoista on olemassa selkeät säännöt ja ohjeet.

Moduulisarjoja kutsutaan avoimiksi, jos erilaisten yhdistelmien lukumäärä on ääretön tai suuri, sekä suljetuksi, kun yhdistelmien lukumäärä on äärellinen. Moduloidun tuotesarjan kaksi perimmäistä etua on asiakasvarianttien hallittu määrä sekä lyhyempi tuotteen läpimenoaika suunnittelupöydältä kaupan hyllyyn. Moduloitaessa on syytä tavoitella sellaista moduulikokonaisuutta, että asiakkaiden haluamat muutokset on mahdollisimman helppo huomioida kokonaisuudessa, kuitenkin niin, että tuoteperheen sisällä olisi silti paljon yhtenäisiä osia. Ominaisuuksista on tärkeä määritellä selkeästi ne, joihin asiakas saa vaikuttaa, ja myös ne, joihin asiakas ei saa vaikuttaa. Osan ominaisuuksista määrittää yrityksen omat sidosryhmät, näihin ei välttämättä voida tai ole järkevä vaikuttaa edes yrityksen sisäisesti.

Moduloitavan tai ylipäänsä standardisoitavan tuotteen on täytettävä tiettyjä ominaisuuksia, että projekti voisi olla hyödyllinen:

- Tuote on elinkaarensa alkuvaiheilla.
- Tuote on taloudellisesti kannattava, merkittävä tai sen volyymi on riittävä.
- Tuote on ohittanut jo prototyyppi-asteen ja on valmis tuote.

- Tuotteen on oltava riittävän samankaltainen rakenteiden osalta sekä mielellään edustaa nykytekniikkaa.

/8,9/

4.3 Tuotteen vaatimukset

Riippumatta siitä, onko tuote standardisoitu tai moduloitu, voidaan sitä automatisoida. Kuitenkin, jos aletaan uutta tuotetta suunnittelun osalta automatisoida, on syytä pohtia standardisoinnin mahdollisuudet tarkasti läpi. Tuoteperheen ja yksittäisen konfigurointiin kelpaavan tuotteen raja on hyvin häilyvä, siksi automatisointia ei tule unohtaa ajatuksista, vaikka kyse olisikin täysin yksittäisestä tuotteesta. Projektiluontoisesti toimitettavat tuotteetkin voi olla osin tai suurelta osin automatisoitavissa. Isot projektitkin voivat noudattaa logiikkaa, kunhan suunnitteluohjeisto luodaan riittävän hyvin. Monesti asiakkaan ei tarvitse projekteissa vaikuttaa pääosaan yksityiskohtia, vaan ne ovat suunnittelijan mielivaltaisesti luomia.

Automaattista tuotannon kokoonpanoa ja suunnittelun automatisointia voi tietyiltä osin rinnastaa, erityisesti on syytä ymmärtää, että kaiken voi automatisoida, mutta kaikki ei ole kannattavaa. Pidä mielessä alusta alkaen ajatus, että automatisoidaan vain se osa, joka on taloudellisesti kannattavaa, vaihtoehtojen ei tule olla kaikki tai ei mitään. Alan konsultit ja kolmannen sektorin ohjelmistoja tarjoavat yritykset kyllä auttavat tuotteen arvioinnissa, jos epäilet automatisoinnin mahdollisuuksia. Automaattisessa kokoonpanossa korostetaan rakenteen yksinkertaisuutta ja osien vähyyttä, cad-ohjelmassa nämä eivät ole välttämättä ne vaikeimmat asiat. Samankaltaisia lain alaisuuksia kuitenkin tuotteelle on pystyttävä määrittämään, jotta automatisoinnissa olisi mitään ajatusta.

”*logiikka = järjellisen päättelyn yleiset säännöt; niitä koskeva tieteenala; niiden noudattaminen, johdonmukaisuus*” /6/

Kun itse tuotteesta halutaan tehdä automaattisesti hallittava, pitäisi sanan logiikka osua siihen jollain tavalla. Kun pystyt kirjoittamaan paperille tuotteen suunnittelun

perusteet, pystyt luultavasti rakentamaan mallinnuksen jälkeen tuotteesta myös automaatin tulevaisuuden variointeja varten. Tuotteen esisuunnittelussa pitäisi pystyä päättelemään tuotteen mahdollisuudet automatisointiin sekä tarpeet siihen. Esisuunnitteluvaiheessa tulisi pohtia automatisoinnin tasoa ja työkaluja, joita tuote voisi vaatia. Automatisointia voi myös hyödyntää aihion teossa. Tehdään raakilemalli, jota hallitaan automaattisesti, kuitenkin niin, että tavoitellaan lopputuloksena vain puoliksi suunniteltua tuotetta. Tälläkin voidaan pystyä vähentämään suunnitteluun kuluvia työtunteja paljon.

Makrojen mahdollisuudet on syytä myös pohtia jo esisuunnittelussa; niiden käytöstä yrityksen suunnittelussa olisi hyvä muodostaa vakioitu käytäntö.

4.4 Suunnittelutyön vaatimukset

Suunnittelutyön vaatimuksilla tarkoitetaan sellaisia asioita, jotka tulee tiedostaa tehtäessä mekaanista työtä cad-ohjelmien kanssa. Ensimmäiseksi lienee tarpeen esitellä kaksi ääripäätä edustavaa suunnitteluteknikkaa. Molemmat menetelmät esiintyvät lähinnä kokoonpanojen teossa, eivät niinkään yksittäisen osan suunnittelussa.

Bottom-Up suunnittelutapa tarkoittaa yksittäisten osien mallintamista sellaisinaan etukäteen, ja tämän jälkeen kokoonpano kootaan aiemmin tehdyistä valmiista osista. Tämä tapa soveltuu, jos osat on hyvin vakioituja tai ennalta tunnettuja, tai jos suunnittelutyö on hyvin organisoitua ryhmätyöskentelyä. /3/

Top-Down suunnittelutapa on päinvastainen: Osat mallinnetaan suoraan kokoonpanoon ja sidotaan kiinteisiin paikkoihin kokoonpanossa. Tällä periaatteella etuina tulee kokoonpanon olemassa olevan geometrian hyödyntäminen ja osien helpompi paikoilleen asettelu. /3/

Käytännön työssä yleensä kuitenkin tapoja yhdistellään kulloisenkin tarpeen ja olemassa olevan suunnittelutiedon perusteella. Suunnitteluautomaation näkökulmasta top-down menetelmä on parempi. Vähemmän esitetty tekniikka on myös ns. skeleton-tekniikka tai luurankonakin nimitetty, tässä tekniikassa hyödynnetään luuranko-

tyyppistä apugeometriaa kokoonpanon luonnissa, yleisimmin tätä tekniikkaa käytetään, kun kootaan profiilirakenteisia kokoonpanoja. Tekniikka soveltuu myös muunlaisten kokoonpanojen kokoamiseen.

Sidokset, geometriset ehdot, relaatiot ja symmetria - siinä lista tärkeitä sanoja ja asioita, joilla on automaatin toiminnan kannalta todella tärkeä rooli. Kaikille suunnittelijoille on varmasti tuttu ilmiö, että malli ns. räjähtää käsiin. Jokin osa ei pysy paikallaan tai siirtyy holtittoman tuntuisesti paikasta toiseen, näihin asioihin reagoidaan aiemmin mainituilla taikasanoilla.

Sidoksia, relaatioita ja symmetriaa tarvitaan niin 2d-luonnoksissa kuin 3d-piirteiden ominaisuuksissa. Näiden ominaisuuksien käytöllä on ratkaiseva merkitys mallin toiminnalle. Automaatiossa käytettäviä muuttujia olisi hyvä olla mahdollisimman vähän, ja niitä muuttujia, joita käyttäjältä otetaan vastaan, tulisi käyttää tehokkaasti mallissa. Kun on mahdollista jättää joku asia mitoittamatta ja hoitaa paikoitus relaatiolla tai symmetrialla, se kannattaa hyödyntää. Mallinnusta aloittaessa on myös hyvä pohtia normaalien tasojen suunnat, sekä koordinaatiston ja origon sidonta. Sidoksien elementtien referensseinä ei saisi käyttää sellaisia elementtejä, jotka eivät ole varmuudella vakioita; älä siis käytä katoavia elementtejä tai sellaisia, jotka voivat liikkua tavalla, jota et halua. Tämän suunnittelun kun tekee ennen mallinnusta, on symmetrioiden käyttö huomattavasti helpompaa. Hyvään suunnittelutapaan kuuluu myös yksinkertaisuus, älä tee ylimääräisiä piirteitä sekä ajattele mallin ylläpidettävyyttä toisen ihmisen näkökulmasta oman työsi sijaan.

Automaation näkökulmasta optimaalisesti katsottuna malleja ei tarvitse suunnitella mitenkään erilaisella tavalla kuin tavallisessa suunnittelussa. Tämä voi kuitenkin olla hieman optimaalinen katsontakanta asiaan.

5 AUTOMATISOINNIN HYÖDYT JA VAARAT

Ymmärtäminen vaatii aikaa. Suorittaminen on nopeaa.

Suunnitteluautomaatio on kallista?

Suunnitteluautomaatioprojektit ovat aivan samanlaisia projektia muiden tehostamisprojektien kanssa. Niillä on oltava selkeä tavoite, ja tavoitteen saavuttamiselle on asetettava joku hintalappu. Hintalapulla on sitten kääntöpuoli, jossa on arvioitu investoinnin kannattavuutta. Tärkeää hintalapun muodostumisen kannalta onkin ymmärtää mitä voidaan automatisoida ja mitä ei missään nimessä kannata. Käsityön ja automaattisen työn kustannuserot ovat hyvin riippuvaisia työn vaativuudesta.

Kannattavuuden arvioinnissa sekä mahdollisuuksien pohtimisessa voidaan turvautua alan yrityksien apuun. Yrityksen oman ammattitaidon loppuessa onkin tärkeä muodostaa puolueeton kokonaiskuva investoinnin kannattavuudesta sekä realistisista mahdollisuuksista. Hyvin suunnittelut ja toteutetut projektit ovat äärimmäisen kannattavia investointeja parhaimmillaan. Suunnitteluun kuluvat työmäärät voivat vähentyä jopa 80-90% alkuperäiseen käsityöhön verrattuna.

Suunnitteluautomaatio on joustamatonta?

Varmasti on, jos järjestelmä on sellaiseksi rakennettu. Logiikkaa noudattavat rakenteet ja järjestelmät ovat juuri niin älykkäitä kuin ne on tehty. Järjestelmien suunnittelussa onkin tärkeää pitää kanavat auki aina tulevaisuutta ajatellen. Järjestelmiä pitää pystyä päivittämään käyttöönoton jälkeenkin käyttäjien toiveiden ja vaatimuksien perusteella. Yksinkertaiset ja loppuun asti mietityt tuoteperheet on helpompi tehdä joustaviksi kuin yksittäiset tuotteet, joista ei ole kokonaisnäkemystä. Järjestelmää tehdessä tulisi pitää mielessä aivan alusta alkaen se mahdollisuus, että järjestelmään halutaan lisätä useita erilaisia pieniä ominaisuuksia jälkeenpäin. Huonosti toteutetussa järjestelmässä tämä on vaikeaa tai jopa mahdotonta. Erityisesti on syytä pohtia toimitaanko avoimessa vai suljetussa järjestelmässä, eli jääkö järjestelmän tehneelle taholle kaikki oikeudet tuotteeseen, vai voidaanko lisäpalveluita mahdollisesti tehdä myöhemmin itse tai hankkia muualta.

Joustoja pohtiessa on hyvä vetää konkreettisia rajoja sille, mitä automaatti tekee ja mitä ei. Näin voidaan jättää käsityölle se osuus, jossa todella tarvitaan suurta määrää variaatioita. Suunnitteluautomaation joustoja on helppo ajatella, kun pohtii automaattista valmistusta ja käsityötä. Ihminen on äärimmäisen joustava valmistaja verrattuna CNC-ohjattuun koneeseen.

Suunnitteluautomaatio on häiriöaltista?

Siitä ajasta on tultu jo kauas, jolloin tietokoneet kaatuilivat päivittäin ja ilmiö oli arkipäiväinen. Nykyään ei enää murehdita niinkään tietokoneiden tai järjestelmien pystyssä pysymistä vaan sitä, että niissä on joku rakenteellinen vika. Suunnitteluautomaatio poistaa ihmiseen verrattuna inhimilliset virheet tehokkaasti. Logiikkaa noudattava järjestelmä ei poikkea siitä, ilman sille opetettua poikkeusta. Ihminen tekee säännöllisesti inhimillisiä virheitä, joilla osalla on merkitystä ja osalla ei.

Suunnitteluautomaatio jättää kuitenkin huomattavasti vaarallisemman virheen mahdollisuuden järjestelmään. Näitä ovat systemaattiset virheet. Systemaattisella virheellä tarkoitetaan virhettä, joka tulee itse järjestelmästä. Esimerkiksi mittaustekniikassa systemaattinen virhe johtuisi käytetystä mittalaitteesta. Systemaattiset virheet ja niiden mahdollisuudet pitäisi tunnistaa jo järjestelmää luodessa, ettei sellaisia voi jäädä mihinkään kohtaan järjestelmää.

Hyvin testattu ja toteutettu automaatiojärjestelmä ei ole häiriöaltis. Ohjelmistoteknisesti cad-ohjelmien automatisointi ei ole kovin vaativaa eikä korkeatasoista ohjelmointityötä. Tämän takia toimiva logiikka ja hyvin määritelty tuote on paljon tärkeämpiä kuin viimeisintä huutoa olevan tekniikan käyttö automaatin toteutuksessa.

6 CASE-ESIMERKKEJÄ

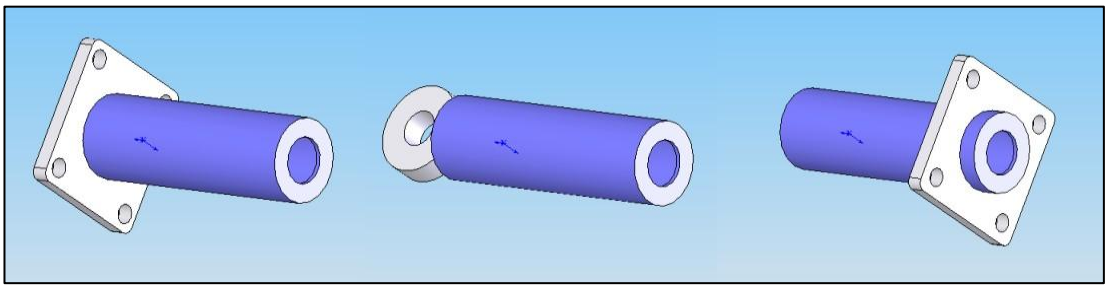
Tässä luvussa esitettävät case-esimerkit on julkaistu erillään muusta teoksesta, näihin esimerkkeihin sovelletaan Creative Commons Nimeä lisenssiä. Saat vapaasti levittää ja käyttää esimerkkejä, kunhan mainitset niiden käytön yhteydessä tekijän nimen.

Esimerkit on myös ladattavissa internetistä osoitteesta http://www.iki.fi/juhoa/insinoorityo. Siellä voit tutustua myös lisenssiehtoihin.

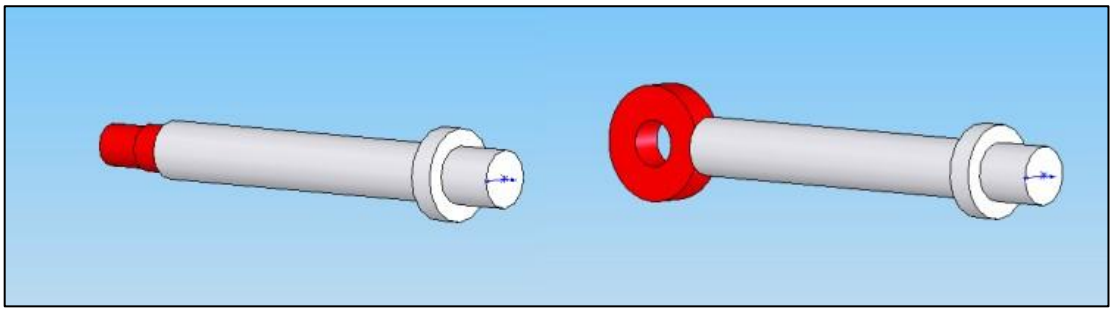
6.1 Hydraulisylinteri mallisarja

Tässä esimerkissä esitellään hydraulisylinterien tuoteperhettä varten tehtyä automaattia. Ohjelma voisi olla esimerkiksi sidosryhmille jaettava helppokäyttöinen sovellus, tätä esimerkkiä voisi hyvin jatkojalostaa hoitamaan tilauksia suoraan tehtaalle, ja lisätä siihen suunnittelua helpottavia tietoja suoraan cad-ohjelman sisään.

Esimerkkiä varten mallinnettiin hydraulisylinteri yksinkertaistettuna, se tehtiin kahdesta erillisestä osasta. Runko ja mäntä tehtiin omina osina, ja näistä luotiin varsinainen kokoonpano. Kumpaankin osaan tehtiin konfiguraatioita, konfiguraatiot tehtiin sylinterin kiinnitystapoja varten. Niihin mallinnettiin erilaiset kiinnitystavat, joita on rungossa kolme ja varressa kaksi. Näitä yhdistelemällä tulee kuusi erilaista variaatiota, aiemmin viitattiin liitteeseen 4, jossa on esitetty tämän esimerkin design table näistä konfiguraatioista.



Kuva 3, erilaiset variaatiot rungon kiinnityksistä.



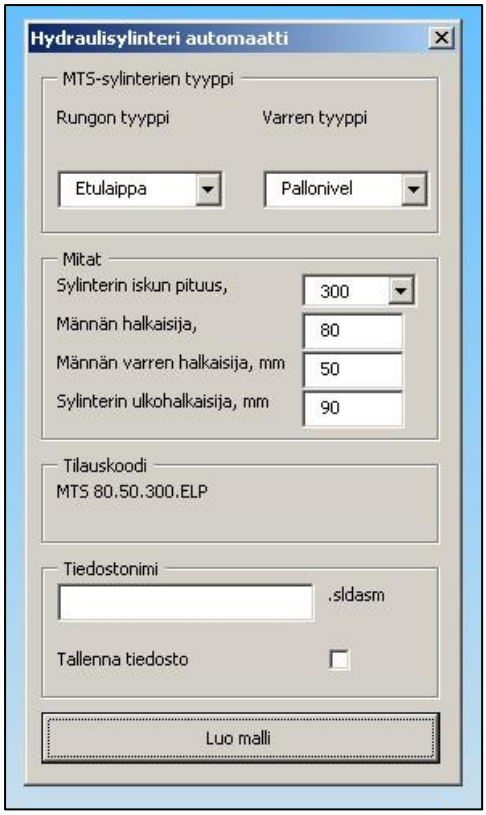
Kuva 4, erilaiset variaatiot männänvarren kiinnityksistä.

Ohjelman toiminta-ajatukseksi hahmoteltiin seuraavanlaiset toiminnot:

- Sylinterin mittoja voidaan muuttaa vapaasti, näiden väliset riippuvuudet opetetaan ohjelmalle.
- Haluttu konfiguraatio pystytään valitsemaan suoraan valikoista.
- Mallisto antaa tilauskoodin valittujen mittojen perusteella.
- Ohjelma tallentaa, käyttäjän niin halutessaan, kokoonpanon paikkaan x.

Liitteessä 2 on esitettynä sisäänrakennettu käyttöliittymä tähän ohjelmaan liittyen, sekä kuvassa on myös itse kokoonpano hydraulisylinteristä. Kuvassa 5 esitetään ulkoinen käyttöliittymä. Sylinterien pituudet asetettiin alasvetovalikosta valittaviksi mutta halkaisijat on vapaasti kirjoitettavissa. Todellisuudessa varmaan kaikki mitat olisi tiettyjen taulukoiden perusteella valikoitavissa. Tähän yksinkertaistettuun esimerkkiin ohjelmoitiin vain pituuden muutokset ja konfiguraatioiden valinta, sekä tilauskoodin muodostus ja tiedoston tallennus.

Ohjelmaa käytetään niin, että se käynnistetään SolidWorksin makrojen kautta, ja tämän jälkeen sillä voidaan reaaliaikaisesti muokata auki olevaa mallia. Tämä on yksi hienoimpia ominaisuuksia SolidWorks ja VBA yhdistelmässä. Varsinainen lähdekoodi ohjelmaan on esitetty liitteessä 5, koodin sekaan on esitetty kommentit ja ajatukset liittyen itse koodiin.



Kuva 5: Ulkoinen käyttöliittymä automaatista.

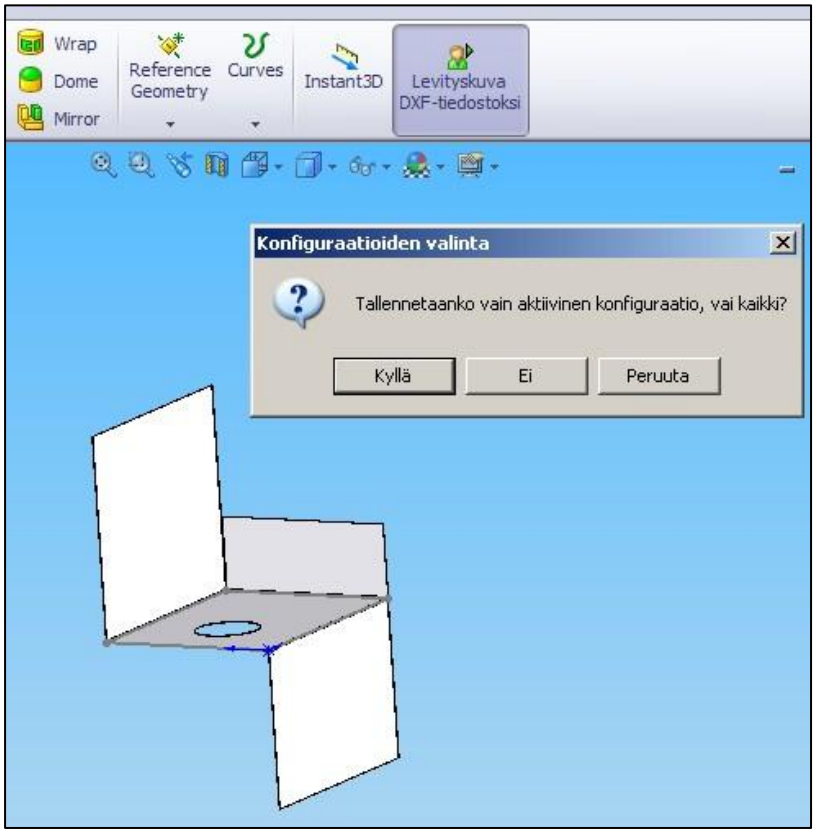
Mitä tällä sitten saavutettiin? Sylinterin pituusvariaatioita oli 13 kappaletta, erilaisia kiinnityksiä oli 6 kappaletta. Tästä mallisarjasta tällä halkaisijalla pelkästään voisi tilata siis 78 erilaista sylinteriä. Tähän jos lisätään vaikka 10 erilaista halkaisijavariaatiota, niin mallisarjaan kuuluu jo lukematon määrä erilaisia variaatioita. Tähän yksinkertaistettuun esimerkkiin piirrettiin vain muutama konfiguraatio ja annettiin automaation hoitaa loput variaatiot. Tämän kaltaisen automaatin ylläpito jatkossa olisi paljon helpompaa. Tämän avulla niin omaan suunnitteluun kuin sidosryhmille saataisiin helposti jaettava ohjelma, josta saadaan todellisuutta vastaavat mallit.

Vastaavia käytännön esimerkkejä löytyy cad-ohjelmista ja niiden työkalupakeista. Standardiosia jaetaan tämän kaltaisin ohjelmin varustettuna yleisesti. Yrityksen sisäisessä käytössä tällaiseen ohjelmaan pystyttäisiin sisällyttämään järkeä huomattavasti enemmän ja helpommin kuin manuaalisessa työssä voidaan.

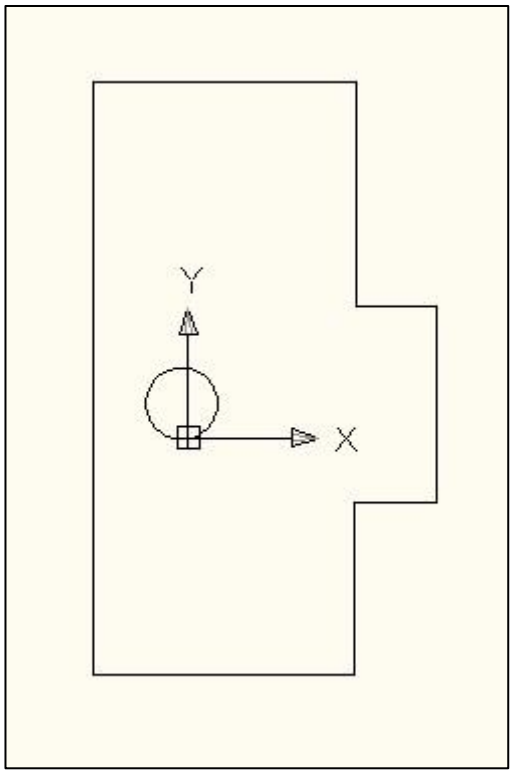
6.2 Levityskuvan automaattinen tallentaminen dxf-muotoon

Tämä esimerkki kuuluu makrojen kategoriaan. Makro on suunniteltu nopeuttamaan levyjä suunnittelevan cad-piirtäjän arkirutiineja. Tähän ohjelmaan ei tehty mitään erillistä käyttöliittymää, vaan se suoritetaan sille luodusta pikakuvakkeesta Solid-Worksin valikosta suoraan, ja se toteuttaa orjallisesti opetetun logiikan mukaisesti ainoan toimenpiteen, jota varten se on olemassa.

Kun makro ajetaan, se kysyy käyttäjältä vain yhden kysymyksen ja antaa palautteen suorituksen perusteella.



Kuva 6, pikakuvakkeesta käynnistetty makro.



Kuva 7, Kuvan 6 kappaleen levityskuva ilman taivutusviivoja

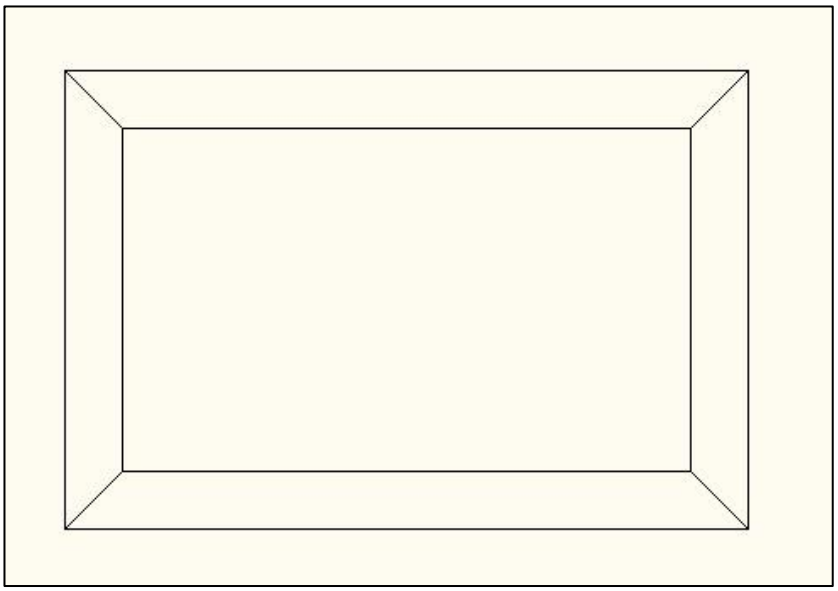
Mikäli käyttäjä vastaa kyllä, niin ohjelma tekee vain aktiivisen konfiguraation levityskuvasta dxf-tiedoston. Käyttäjän vastatessa ei, ohjelma tekee levityskuvan kaikista konfiguraatioista. Peruuttamalla ohjelman ajo keskeytetään. Lopputuloksena, samaan hakemistoon missä malli on tallennettuna, syntyy dxf-tiedostoja 1:1 mittakaavalla ilman mitään ylimääräistä. Taivutusviivojen tuleminen dxf-tiedostoon riippuu tallennusasetuksista.

Samaa ideaa noudattavia makroja voidaan tehdä hyvinkin paljon nopeuttamaan suunnittelijan työtä. Monesti tämän kaltaiset esimerkit toimivat todella hyvin, kun tehdään jonkinlaista sarjatyötä. Oli se sitten tulostamista, tallentamista tai jotain muuta.

Esimerkin lähdekoodi on kokonaisuudessaan kommentoituna liitteessä 6.

6.3 Ikkunaraamin piirtäminen Autocadissa AutoLispin avulla

Tämä esimerkki on kahteen edelliseen verrattuna erilainen. Ensinnäkin tämä on tehty Autocadiin, ja tässä piirretään 2d-kuvaa alkeistoiminnoin. Tämä ohjelma havainnollistaa sitä, mihin Autodesk alun perin Lisp-kielen käytön ajatteli, monimutkaisempien käskyjen tai niiden yhdistelmien tekemiseen.



Kuva 8, Ohjelman lopputulos.

```
Regenerating model.
AutoCAD Express Tools Copyright © 2002-2004 Autodesk, Inc.
AutoCAD menu utilities loaded.*Cancel*
Command:
Command:
Command: _appload ikkunapoka.lsp successfully loaded.
Command:
Command:
Command: ikkunapoka
Anna ikkunapokan nollapiste:
Anna pokan leveys: 1200
Anna pokan korkeus: 800
Anna pokan paksuus: 100
```

Kuva 9, ohjelman suorittaminen käskyrivillä ja annetut parametrit.

Ohjelma piirtää siis kuvitteellista jiirin sahatuista laudoista tehtävää ikkunaraamia. Ohjelma vaatii parametreina nollapisteen, joka on vasen alanurkka, sekä leveyden, korkeuden ja laudan paksuuden.

Ohjelman lähdekoodi on kokonaisuudessaan esitettynä liitteessä 7. Lähdekoodissa määritellään aliohjelma *keha*, joka vastaa Autocadin käskyä *rectang*. Tämä havain-

nollistaa Lispin käyttökelpoisuutta tehdä alkeistoimintoja hyväksi käyttäen laajempia toimivia kokonaisuuksia kulloisenkin käyttötarkoituksen mukaan.

Tämän esimerkin kaltaisia laajempia sovelluksia käytetään esimerkiksi rakennusteollisuuden cad-apuohjelmissa.

6.4 Esimerkki aikataulun laatimisesta

Kuvitellaan teoreettista esimerkkiä tuotteesta, joka muodostuu muutamasta alikokoonpanosta, jotka edelleen koostuvat muutamasta osasta. Noin puolet osista on sellaisia, joihin asiakkaalla on mahdollisuus vaikuttaa tuotteen tilauksen yhteydessä. Tuote muodostaa myös tuoteperheen edellä mainituin ehdoin niin, että siitä myydään suurta määrää erilaisia mitta- ja koko-variaatioita. Tuote on ollut tuotannossa jo hyvän tovin ja valmistus- sekä myyntimäärät ovat vakiintuneita. Käytännössä on havaittu asiakkaiden vaatimuksien aiheuttavan paljon manuaalista suunnittelutyötä yksittäisiin osiin, mikä on edelleen vaikuttanut siihen, että mittaporrastus tuoteperheessä ei toimi niin kuin sen alun perin ajateltiin.

Yritys on päättänyt saada suunnittelukustannukset kuriin, kuitenkin uhraamatta asiakkaiden mahdollisuutta varioida tuotetta, koska on havaittu sen olevan kriittinen tekijä tuotteen myynnissä verrattuna kilpailijoiden tuotteisiin. Tuotetta aletaan siis tutkia automatisoinnin lähtökohdista ja pohtia, mitä projekti vaatii omalta väeltä ja mihin tarvitaan ulkopuolisia toimijoita. Tavoitteita projektille asetetaan kaksi. Ensimmäisenä halutaan saada tuotteen myyntikate nousemaan vähentämällä suunnittelutyötä tuotetta kohden, toisena pyritään parantamaan suunnittelun laatua, jonka on havaittu johtuvan asiakasvariaatioiden suunnittelusta. Suunnittelun laadullisista puutteista on aiheutunut ylimääräisiä kustannuksia valmistuksessa.

Projektiryhmään valittiin suunnittelupäällikkö, kaksi suunnittelijaa ja työnjohtaja valmistuksen puolelta. Ryhmä aloitti työnsä selvittämällä käytössä oleva 3d-cad ohjelmiston automatisointimahdollisuuksia. Esiselvitystä tehtiin yhdessä ohjelmiston jälleenmyyjän kanssa.

Todettuaan käytössä olevan ohjelmiston potentiaalin ja nykyisten mallien käytön olevan mahdollista, aloitti ryhmä speksin määrittelemisen. Ajankäytöllisesti esiselvi-

tys projektin mahdollisuuksista ei vienyt kauaa, päivässä ryhmä pystyi selvittämään teoreettiset mahdollisuudet. Speksin määrittelyyn uhrattiin aikaa jo paljon enemmän. Koko ryhmä, välillä täydennettynä muutamilla ulkopuolisilla, kokoontui useamman päivän ajan pohtimaan mahdollisia variaatioita, sallittuja tapauksia ja asioiden välisiä riippuvuuksia.

Speksin pääteemoiksi nousi mittojen vapaa muunneltavuus, koska automaatti hoitaisi piirustuksien teon, joten tämä ei aiheuttanut merkittäviä kustannuksia tulevaisuudessa, ja muutaman osan irrottaminen automaatista manuaalisesti tehtäviksi osiksi.

Muutaman päivän määrittelyiden jälkeen suunnitelmaa esiteltiin kolmannelle osapuolelle, joka tulisi toteuttamaan varsinaisen automaatin. Tässä yhteydessä kolmannelle osapuolelle esiteltiin tarkasti tuote, sen nykytila ja tuotteen valmistus. Kolmannen osapuolen kanssa istuttiin pari päivää tutkimassa nykyisiä rakenteita ja sovittamassa speksiä käytäntöön.

Muutaman päivän jälkeen, kun kolmas osapuoli aloitti käytännön työnsä, heille heräsi lisää määreitä, joita piti tarkentaa. Pieniä yksityiskohtia nousi pinnalle koko ajan läpi projektin, ei niinkään sen takia, ettei projektia olisi voitu toteuttaa ilman tietoa niistä. Tarkentavilla kysymyksillä tuotteesta tehtiin kuitenkin entistä parempi, tämä lisätyö kuitenkin aiheutti lisäkustannuksia.

Muutaman viikon työstämisen jälkeen automaatin ensimmäistä versiota esiteltiin jo yrityksessä. Käyttäjiltä saadun palautteen perusteella tehtiin vielä viikon ajan korjauksia ja ennen kaikkea muutoksia käyttöliittymään. Tämän jälkeen järjestelmä toimitettiin käyttäjille ensimmäisiä varsinaisia koekäyttöjä varten.

Ensikosketuksen jälkeen käyttäjille ja projektissa työskennelleille järjestettiin käyttökoulutusta automaattiin. Tähän varattiin ensi alkuun päivä, jonka jälkeen käyttäjät saivat omatoimisesti hautoa viikon verran automaatin käyttöä. Tässä kohtaa automaatilla ei kuitenkaan vielä tehty mitään todellista työtä. Tämän jälkeen pidettiin vielä toinen koulutuspäivä, jossa keskusteltiin paljon dokumentoinnista, käyttöliittymästä ja pinnalle nousseista ajatuksista edellisen viikon aikana.

Aktiivisen käytön alettua yhteydenpito kolmanteen osapuoleen jatkui tiiviinä. Järjestelmää kehitettiin melko reaaliaikaisesti aina, kun käyttäjät havahtuivat johonkin epäloogisuuteen tai käytännön sanelemaan virheeseen.

Muutaman kuukauden aktiivisen käytön jälkeen yritys oli niin tyytyväinen järjestelmään, että halusi jatkojalostaa sitä edelleen. Yhdessä kolmannen osapuolen kanssa järjestelmään ideoitiin lisätoimintoja, jolla se liitettiin projektinhallintajärjestelmään ja valmistusteknisiin järjestelmiin.

7 JOHTOPÄÄTÖKSET

Tässä työssä on esitetty ja pohdittu yleisesti suunnittelutyön automatisoinnin erilaisia mahdollisuuksia ja tekniikoita. Kun palataan alkuperäiseen ongelmaan, joka ensimmäisessä luvussa asetettiin työlle, voidaan todeta, että asiaa on käsitelty monelta eri kantilta. Käytännössä kun aihetta yritysmaailmassa aletaan lähestyä, selvitystyöt kohdistuvat ensin olemassa oleviin työkaluihin. Tämä työ vastaa kysymyksiin liittyen cad-ohjelmien työkaluihin. Cad-ohjelmien päivityksissä ja vaihdoissa kannattaa perehtyä automatisoinnin mahdollisuuksiin.

Työkalujen potentiaalin selvittyä siirrytään tuotteen pohdiskeluun. Tuotetta voidaan, ja täytyy pyöritellä monesta eri näkökulmasta kun sen suunnittelua halutaan automatisoida. Työ vastaa perustaviin kysymyksiin standardoinneista, moduloinneista, tuotteen perusvaatimuksista ja huomioista käytännön cad-työskentelyssä. Mahdollisen projektin onnistumisaste määräytyy pitkälti tämän vaiheen onnistumisen myötä.

Automatisoinnin eri tavat käydään kirjassa läpi niin teoriassa, kuin niitä tukevin esimerkein. Kirjassa olevat kuvat tukevat lukijan hahmotuskykyä esimerkkien suhteen hyvin. Liitteissä esitetyt esimerkit voivat olla avuksi ja tueksi aloittelevalle ohjelmoijalle. Esimerkit on tarkoituksella julkaistu eri lisenssillä muusta teoksesta, jolloin niitä voi vapaasti käyttää parhaaksi katsomallaan tavalla.

Minä itse uskon ja väitän, että tämän alan liikevaihto tulee tulevina vuosina kasvamaan roimaa vuosivauhtia. Alan potentiaali on kasvanut niin merkittävästi cad-ohjelmien kehityksen ja 3d-cadien läpilyönnin myötä. Tämän alan kehityksessä kannattaa pysyä mukana ja seurata alaa. Ne alat, joiden tuotteet ovat suunnitteluvaltaisia eivätkä hinnat riipu pelkästään esimerkiksi materiaalin hinnasta, voivat saavuttaa suurta kilpailuetua suunnittelun automatisoinnilla.

LÄHTEET

1. Lukka K., 2001, Konstruktiivinen tutkimusote, http://www.metodix.com.lillukka.samk.fi/fi/sisallys/01_menetelmat/02_metodiartikkelit/lukka_const_research_app/?tree:D=168562&tree:selres=&hrpDelimChar=%3B&parentCount=1, viitattu 17.12.2009

2. Visual Basic For Applications, Wikipedia. Viitattu 22.12.2009, http://en.wikipedia.org/wiki/Visual_basic_for_applications

3. Hietikko E., 2007. SolidWorks – Tietokoneavusteinen suunnittelu, 2painos, Mikkeli, Savonia-ammattikorkeakoulu

4. Vertex Systems Oy – Suunnitteluautomaatit, viitattu 23.12.2009, http://www2.vertex.fi/web/fi/suunn_autom

5. Lasse Home, 2005, Pikatestit, Dynaaminen cad-konkari, MikroPC nro. 8, sivu 64

6. Korpela J, Pienehkö sivistyssanakirja, viitattu 27.12.2009, http://www.cs.tut.fi/~jkorpela/siv/

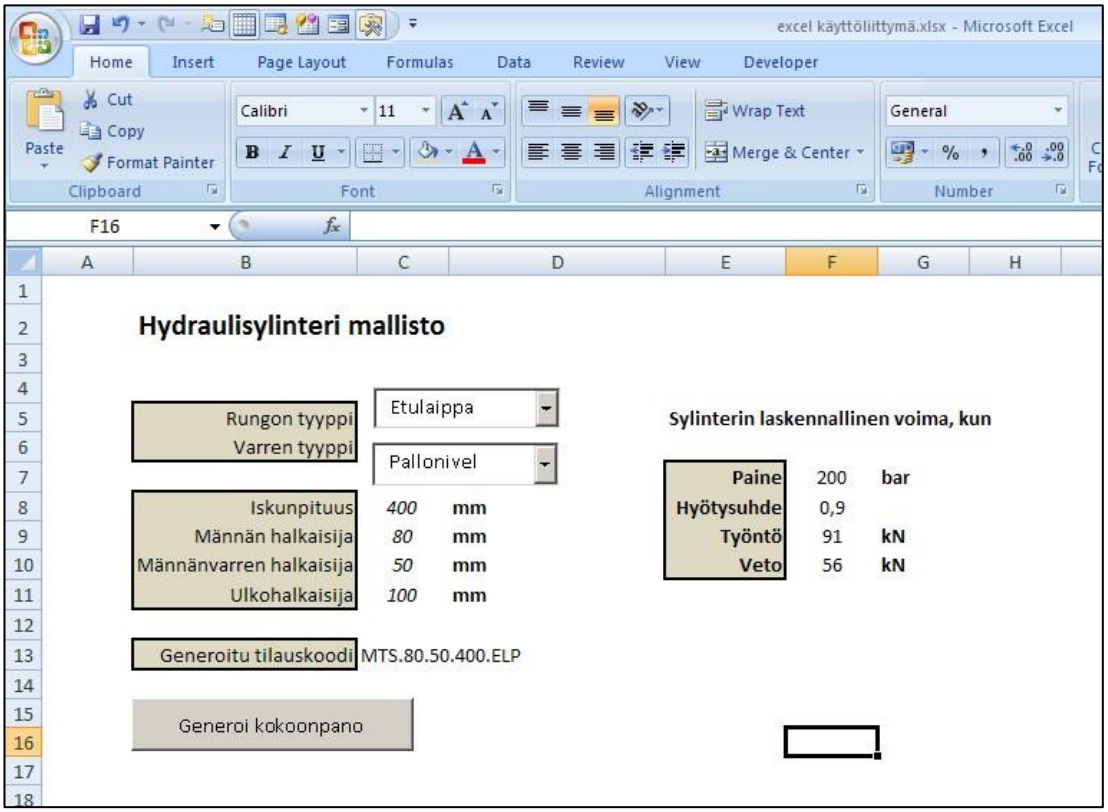
7. Standardit ja standardisointi SFS-käsikirja 1 (2006), viitattu 28.12.2009, www.sfs.fi/files/kk1.ppt

8. Laurila, T., 1987, Tuotestandardisointi, Helsinki, Metalliteollisuuden kustannus Oy

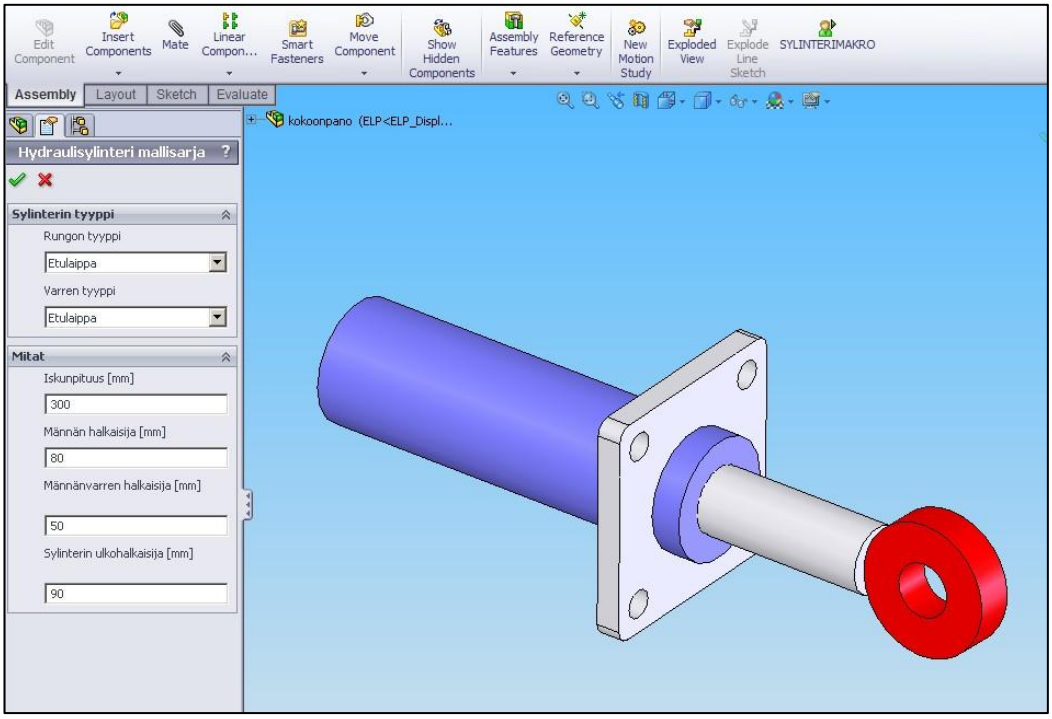
9. Österholm, J. & Tuokko, R., 2001, Systemaattinen menetelmä TUOTE-MODULOINTIIN, Helsinki, Metalliteollisuuden Kustannus Oy

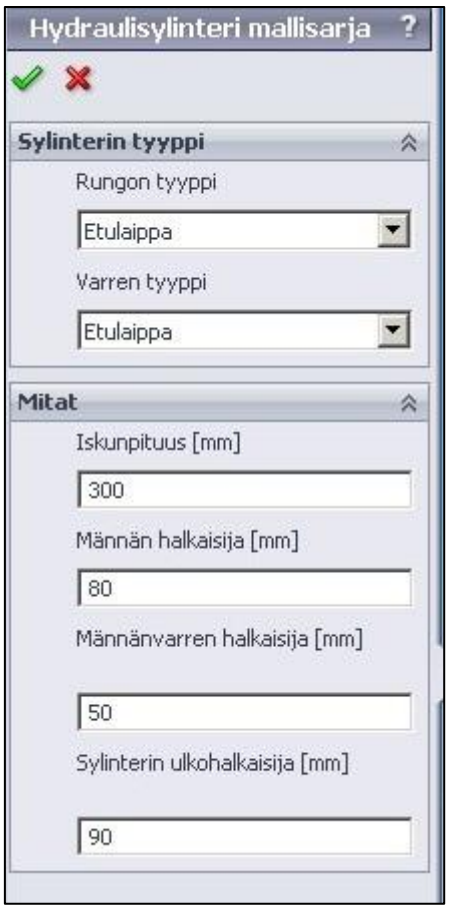
10. Algoritmi, Wikipedia. Viitattu 7.1.2010, http://fi.wikipedia.org/wiki/Algoritmi

Kuvakaappaus Exceliin tehdystä käyttöliittymästä case-esimerkki ykköseen.



Kuvakaappaus kokoonpanosta ja sisäisestä käyttöliittymästä, joka on tehty liittyen case-esimerkki ykköseen.

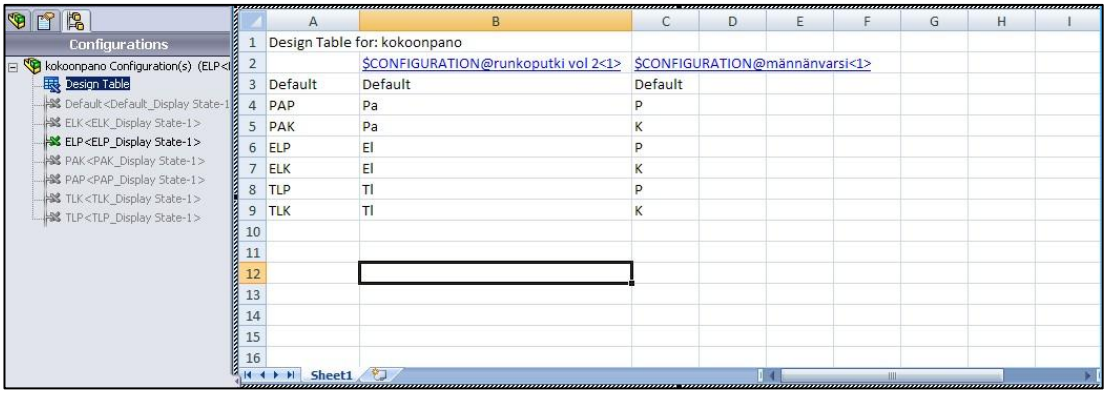




Esimerkki listaus Vertexin käskytiedostosta. Rivit, jotka alkavat ! merkillä, ovat kommentteja.

```
{
! Perustetaan käskyoperaattori, oltava sama kaikissa riveissä.
#Task = VXTASK(VXG4_AUTOASSY,0);
#Task.SETMEAS(Alapaarre.h, 120);
#Task.SETMEAS(Alapaarre.b, 120);
#Task.SETMEAS(Alapaarre.s, 4);
! Asetetaan kolme nimettyä mittaa mallista tiettyihin arvoihin.
#Task.SETATTRIB(Alapaarre,11,0,CODE=1000;
DESCRIPTION=Neliöputkipalkki RHS 120x120x4);
! Asetetaan profiilille määrätty ominaisuus
#Task.SETMEAS(Ylapaarre.h, 120);
#Task.SETMEAS(Ylapaarre.b, 120);
#Task.SETMEAS(Ylapaarre.s, 4);
#Task.SETATTRIB(Ylapaarre,11,0,CODE=1000;
DESCRIPTION=Neliöputkipalkki RHS 120x120x4);
! Ratkaistaan malli täydellisesti uudelleen.
#Task.SOLVE(71)
}
```

Kuvakaappaus esimerkki ykkösen Design tablesta ja konfiguraatioista.



Hydraulisylinteri esimerkin lähdekoodit. Rivit, jotka alkavat ’ merkillä, ovat kommentteja. Tämä makro muodostuu yhdestä moduulista, jossa käsketään vain näyttää lomake. Lomakkeessa on sitten käyttöliittymä ja varsinainen toiminnallinen lähdekoodi on lomakkeessa olevan napin takana.

```
Sub main()
' asetetaan käyttöliittymän lomake näkyville.
   UserForm1.Show
End Sub

Private Sub LuoMalli_Click()
' esitellään muuttujat
Dim Polku As String
Dim runko As String
Dim varsi As String
Dim isku As Integer
Dim tilauskoodi As String
Dim tyyppi As String
Dim Konfiguraatio_nykyinen  As String
Dim Polku_malli As String
Dim swApp As Object
Dim swMalli As Object
Dim palautus As Boolean
Dim boolstatus As Boolean
Dim longstatus As Long
Dim longwarnings As Long
Dim errors As Long
Dim warnings As Long


' tarkastellaan valittu runkotyyppi
If Runkotyyppi.Value = "Pallonivel" Then
```

```
    runko = "PA"
ElseIf Runkotyyppi.Value = "Etulaippa" Then
    runko = "EL"
ElseIf Runkotyyppi.Value = "Takalaippa" Then
    runko = "TL"
End If

' tarkastellaan männänvarren valittu tyyppi
If Varsityyppi.Value = "Kierre" Then
    varsi = "K"
Else
    varsi = "P"
End If
' yhdistetään edellä valitut tiedot yhteen, tämän tiedon perusteella valitaan käytössä oleva konfiguraatio.
tyyppi = runko & varsi
' luetaan iskunpituus
isku = iskupituus.Value

' alustetaan yhteys vba:n ja solidworksin välille.
Set swApp = Application.SldWorks

' alla oleva rivi on kommentoituna tarkoituksella, tämän kaltaisella rivillä voisit avata mallin jos
' se ei olisi jo valmiiksi auki.
'Set swMalli = swApp.OpenDoc6("D:\sw hydr sylinteri\opettajan ver-sio\kokoonpano.SLDASM", 2, 0, "", longstatus, longwarnings)

' aktivoidaan malli. tämä on riippuvainen tiedostonimestä tällä tavalla tehtynä.
Set swMalli = swApp.ActivateDoc2("kokoonpano.SLDASM", False, longstatus)


If swMalli Is Nothing Then
 MsgBox "Virhe mallin aktivoinnissa, makron ajo keskeytetään"
```

```
 Exit Sub
Else
 ' ei virheitä, ajo voi jatkua
End If

' tämä on vasta virheentarkastelun jälkeen, eli jos makroa ajetaan ilman mallin aukiolemista
' ei tule tilauskoodiakaan.
' generoidaan tilauskoodi tunnettujen parametrien perusteella.
tilauskoodi = "MTS " & männänhalkaisija.Value & "." & männänvarrenhalkaisija.Value & "." & isku & "." & tyyppi
' kirjoitetaan tilauskoodi sille varattuun kohtaan käyttöliittymässä
koodi.Caption = tilauskoodi
' tulostetaan myös tilauskoodi käyttäjälle
' tähän voisi lisätä esimerkiksi leikepöydälle kopiointimahdollisuuden.
MsgBox "Tilauskoodisi on " & tilauskoodi, vbOKCancel


swMalli.EditPart
' sörkitään muuttujaa nimeltään rungonpituus joka on Sketch ykkösessä osassa runkoputki vol 2.part
swMalli.Parameter("rungonpituus@Sketch1@runkoputki vol 2.Part").SystemValue = iskupituus.Value / 1000
boolstatus = swMalli.EditRebuild3
swMalli.ClearSelection2 True
swMalli.EditAssembly
swMalli.ShowConfiguration tyyppi
' toistetaan edellä mainitut männänvarrelle.
swMalli.EditPart
swMalli.Parameter("varrenpituus@Sketch1@männänvarsi.Part").SystemValue = iskupituus.Value / 1000
boolstatus = swMalli.EditRebuild3
swMalli.ClearSelection2 True
swMalli.EditAssembly
```

```
swMalli.ShowConfiguration tyyppi

' näytetään oikea konfiguraatio
' konfiguraatiot on tässä kuvassa: ELK, ELP, PAK, PAP, TLK, TLP

' jos tallennusruksi valittuna tallennetaan tiedosto.
If tallenna.Value = True Then
   Polku = tiedostonimi.Text & ".sldasm"
   palautus = swMalli.SaveAs4(Polku, swSaveAsCurrentVersion, swSaveAsOptions_Silent, errors, warnings)
End If

End Sub

Private Sub UserForm_Activate()
' lomakkeen tullessa esiin asetetaan alasvetovalikoihin eri vaihtoehdot.
Runkotyyppi.Clear
Runkotyyppi.AddItem "Pallonivel"
Runkotyyppi.AddItem "Etulaippa"
Runkotyyppi.AddItem "Takalaippa"

Varsityyppi.Clear
Varsityyppi.AddItem "Pallonivel"
Varsityyppi.AddItem "Kierre"

iskupituus.Clear
iskupituus.AddItem "100"
iskupituus.AddItem "150"
iskupituus.AddItem "200"
iskupituus.AddItem "250"
iskupituus.AddItem "300"
iskupituus.AddItem "350"
iskupituus.AddItem "400"
iskupituus.AddItem "450"
```

```
iskupituus.AddItem "500"
iskupituus.AddItem "550"
iskupituus.AddItem "600"
iskupituus.AddItem "650"
iskupituus.AddItem "700"

End Sub
```

Levityskuva DXF-makro

Käyttöönotto tapahtuu luomalla uuden makron, johon kopioit alla olevan lähdekoodin. Sen jälkeen tallenna tiedosto johonkin ja tee siitä pikakuvake työkalurivistöön, tai suorita sitä Tools / Macro –valikon kautta. Lähdekoodissa rivit, jotka alkavat ’ merkillä ovat kommentteja.

```
Option Explicit

Sub main()
' esitellään tarvittavia muuttujia
    Dim swApp                As SldWorks.SldWorks
    Dim swMalli              As SldWorks.ModelDoc2
    Dim Konfiguraatiot       As Variant
    Dim Konfiguraatio_nykyinen  As String
    Dim Konfiguraatio_nimi      As String
    Dim i                    As Integer
    Dim valinta              As Integer
    Dim valitut              As Boolean
    Dim ohitus               As Boolean
    Dim Polku_malli          As String
    Dim Polku_tallenna       As String

' alustetaan yhteys ohjelmaan ja malliin
Set swApp = CreateObject("SldWorks.Application")
Set swMalli = swApp.ActiveDoc
' otetaan nykyisen tiedoston hakemisto muuttujaan
Polku_malli = swMalli.GetPathName

' tehdään virheen tarkastelu sille onko mallia avoinna ylipäänsä.
If swMalli Is Nothing Then
    MsgBox "Virhe. Ei mallia avattuna", vbCritical
```

```
    Exit Sub
End If

' haetaan taulukkoon konfiguraatioiden nimet
Konfiguraatiot = swMalli.GetConfigurationNames
' haetaan muuttujaan aktiivinen konfiguraatio
Konfiguraatio_nykyinen = swApp.GetActiveConfigurationName(Polku_malli)

' annetaan virhe jos mallia ei ole tallennettu.
If Polku_malli = "" Then
    MsgBox "Tiedosto pitää tallentaa ennen levityskuvan tekoa", vbInformation
    Exit Sub
End If

' esitetään kysymys, notta mitkä konfiguraatiot muunnetaan
valinta = MsgBox("Tallennetaanko vain aktiivinen konfiguraatio, vai kaikki?", vbYesNoCancel + vbQuestion, "Konfiguraatioiden valinta")

If valinta = vbYes Then
    valitut = True
ElseIf valinta = vbNo Then
    valitut = False
Else
' cancel peruuttaa makron ajon
    Exit Sub
End If

' loopataan kaikki konfiguraatiot läpi, jos aiemmassa kysymyksessä valittiin
' että käsitellään vain nykyinen konfiguraatio se hoidetaan sisällä olevalla ehtolau-
seella.

For i = 0 To UBound(Konfiguraatiot)
Konfiguraatio_nimi = Konfiguraatiot(i)
```

```
' tarkastellaan konfiguraatiota, jos tallennetaan kaikki konfiguraatiot niin annetaan jatkua
If valitut = False Then
    ' jos konfiguraation nimessä on valmiiksi jo FLAT-PATTERN niin se hylätään
    ' tämä koska solidworks tekee tämän konfiguraation automaattisesti
    ' eli tällä tulisi tupla DXF joka konfiguraatiosta
    If Not Strings.InStr(1, Konfiguraatio_nimi, "FLAT-PATTERN", vbTextCompare) = 0 Then
        ohitus = True
    Else
        ohitus = False
    End If

' jos valittu vain yhden konfiguraation tallennus, tehdään tarkastelu ollaanko nyt juuri siinä.
ElseIf valitut = True And Konfiguraatio_nimi = Konfiguraatio_nykyinen Then
    ohitus = False

' muissa tapauksissa hypätään loopissa eteenpäin
Else
    ohitus = True
End If

If ohitus = False Then

' asetetaan seuraava konfiguraatio aktiiviseksi, jos nykyinen on ainoa tallennettava tämä on ohitettava
' muuten suoritus loppuu tuohon elsen virheeseen
If Not Konfiguraatio_nimi = Konfiguraatio_nykyinen Then
    If swMalli.ShowConfiguration2(Konfiguraatio_nimi) Then
    ' onnistui, ei toimenpiteitä
    Else
        MsgBox "Konfiguraation valinnassa virhe, makron ajo keskeytetään"
        Exit Sub
```

```
    End If
End If

' false optiolla tehdään täydellinen uudelleenpiirto.
If swMalli.ForceRebuild3(False) Then
    ' pakotetaan uudelleenpiirto varmuudeksi, onnistui eli ei toimenpiteitä
Else
    MsgBox "Virhe mallin uudelleenpiirrossa, makron ajo keskeytetään"
    Exit Sub
End If

' konfiguraatio saatu valittua onnistuneesti päälle, siirrytään tallentamiseen
' poistetaan nykyisen tiedoston polusta tiedostopääte SLDPRT
Polku_tallenna = Strings.Left(Polku_malli, Strings.Len(Polku_malli) - 6) + Konfigu-
raatio_nimi & ".dxf"

' Varsinainen tiedoston muunto flatteniksi ja dxf-formaattiin.
' mukana virheen tarkastus onnistumisesta. Toinen tallennusvaihtoehto
' olisi käyttää SaveAs3 metodia ja siellä FLAT-PATTERN konfiguraatiota.
If swMalli.ExportFlatPatternView(Polku_tallenna, 1) Then
    MsgBox "Konfiguraation: " & Konfiguraatio_nimi & " muunto DXF-tiedostoksi
onnistui!"
Else
    MsgBox "Virhe DXF-tiedoston tallennuksessa"
End If
' ohitus-iffin lopetus
End If

' jatketaan konfiguraatio-looppia
Next i

End Sub
```

Ikkunapoka.lsp lähdekoodi

Rivit, jotka alkavat ; merkillä, ovat kommentteja. Ohjelmaa voi testata ja ottaa käyttöön kopioimalla alla olevan lähdekoodin Lisp-editoriin ja tallentamalla tiedoston .lsp-päätteellä. Tämän jälkeen antaa komentoriville käskyn *APPLOAD* ja hakee skriptin, lataamisen jälkeen komentorivi tottelee käskyä *IKKUNAPOKA*.

```
; määritellään aliohjelma keha joka ottaa 3 parametria
; nollapisteen, leveyden ja korkeuden
(defun keha(piste1 lev kork)
  ; luetaan OSNAPin nykyinen arvo muuttujaan
  (setq os (getvar "OSMODE"))
  ;poistetaan OSNAP toiminto poiskäytöstä, aiheuttaa pienillä etäisyyksillä ongelmia.
  (setvar "OSMODE" 0)
  ; lasketaan kehän muut pisteet.
  (setq piste2 (list (+ (car piste1) lev) (cadr piste1)))
  (setq piste3 (list (car piste2) (+ (cadr piste2) kork)))
  (setq piste4 (list (car piste1) (+ (cadr piste1) kork)))
  ; annetaan alkeiskomento polyline edellä lasketuilla pisteillä.
  (command "._PLINE" piste1 piste2 piste3 piste4 "C")
  ; asetetaan OSNAP takaisin tilaan jossa se oli ennen aliohjelman ajoa
  (setvar "OSMODE" os)
)

; esitellään itse pääohjelma. C: ennen aliohjelman nimeä määrittää sen
; että ohjelmaa voidaan kutsua autocadin komentoriviltä suoraan.
(defun C:IKKUNAPOKA()
  ; poistetaan kaiku käskyriviltä pois.
  (setq old_cmdecho (getvar "CMDECHO"))
  (setvar "CMDECHO" 0)
  ; kysytään käyttäjältä suoritukseen vaadittavat tiedot
  (setq nollapiste (getpoint "\nAnna ikkunapokan nollapiste: "))
```

```
  (setq leveys (getdist "\nAnna pokan leveys: "))
  (setq korkeus (getdist "\nAnna pokan korkeus: "))
  (setq paksuus (getdist "\nAnna pokan paksuus: "))
  ; pieni virheentarkastus, exit tekee sellaisenaan saman kuin CTRL+C
  (IF (= nollapiste nil) (exit))
  (IF (= leveys nil) (exit))
  (IF (= korkeus nil) (exit))
  (IF (= paksuus nil) (exit))
  ; ulkokehän piirto, suorittaa aliohjelman määrätyillä parametreilla.
  (keha nollapiste leveys korkeus)
  ; lasketaan sisäkehän nollapiste
  (setq sisapiste (list (+ (car nollapiste) paksuus) (+ (cadr nollapiste) paksuus)))
  ; sisäkehän piirto.
  (keha sisapiste (- leveys (* 2 paksuus)) (- korkeus (* 2 paksuus)))
  ; jiirit
  ; alavasen
  (command "._LINE" nollapiste (list (+ (car nollapiste) paksuus) (+ (cadr nollapiste)
paksuus)) "")
  ;alaoikea
  (command "._LINE" (list (- (+ (car nollapiste) leveys) paksuus) (+ (cadr nollapiste)
paksuus)) (list (+ (car nollapiste) leveys) (cadr nollapiste)) "")
  ;ylävasen
  (command "._LINE" (list (car nollapiste) (+ (cadr nollapiste) korkeus)) (list (+ (car
nollapiste) paksuus) (- (+ (cadr nollapiste) korkeus) paksuus)) "")
  ;yläoikea
  (command "._LINE" (list (+ (car nollapiste) leveys) (+ (cadr nollapiste) korkeus))
(list (- (+ (car nollapiste) leveys) paksuus) (- (+ (cadr nollapiste) korkeus) paksuus))
"")
  ; palautetaan echo päälle
  (setvar "CMDECHO" old_cmdecho)
  ; viimeinen tyhjä princ käsky tekee ohjelman lopetuksesta ns. hiljaisen
  ; eli viimeistä NIL riviä ei palauteta käyttäjän näkyville.
  (princ)
)
```