Opinnäytetyö (AMK)

Kone- ja tuotantotekniikka

Koneautomaatiotekniikka

2017

Esa Kekäläinen

CAD-MALLIN MUOTOTIEDON HYÖDYNTÄMINEN YHTEISTYÖROBOTIN OHJELMOINNISSA

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Kone- ja tuotantotekniikka | Koneautomaatio

Elokuu 2017 | Sivumäärä 36

Esa Kekäläinen

CAD-MALLIN MUOTOTIEDON HYÖDYNTÄMINEN YHTEISTYÖROBOTIN OHJELMOINNISSA

Opinnäytetyön tavoitteena oli tutustua yhteistyörobotiikan perusteisiin ja tapaan kerätä muototietoa CAD-mallista robotin ohjelmointia varten. Työtä on tarkoitus hyödyntää toimeksiantajan valmisteilla olevassa robottisolussa.

Paikoituspisteiden opettamiseen robottia ohjelmoitaessa kuluu paljon aikaa, ja ohjelmointiin kuluva aika on pois varsinaisesta tuotannollisesta työstä. Etäohjelmoinnilla on mahdollista hyödyntää esimerkiksi työstettävän kappaleen CAD-mallin sisältämää tietoa. Etäohjelmointiin vaadittava ohjelmisto voi kuitenkin olla liian suuri investointi etenkin pienelle yritykselle. Tässä työssä pyritään hyödyntämään CAD-mallin muototietoa ilman etäohjelmointiohjelmistoa.

Yhteistyörobotiikkaan perehdyin kirjallisuuden ja verkkomateriaalin avulla, sekä konkreettisesti UR10-robottia käyttäen Automation Assistant Oy:n tiloissa. Ohjelmistoihin tutustuin lähinnä valmistajien tarjoamien käyttöohjeiden pohjalta.

Opinnäytetyön lopussa on esimerkki, jossa muototietoa kerätään SolidWorks-ohjelmistossa, ja siirretään robottisolun käyttöliittymään.

ASIASANAT:

yhteistyörobotiikka, SolidWorks, makro, API, CAD-malli

BACHELOR´S / MASTER'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Mechanical and Production Engineering | Machine Automation

August 2017 | Total number of pages 36

Esa Kekäläinen

UTILIZATION OF THE CAD DRAWING DATA IN COBOT PROGRAMMING

The aim of this thesis was to become acquainted with the basics of cobotics and learn how to collect data from the CAD model for robot programming. The learned information is intended to be utilized in the client's robotic cell under preparation.

It takes a lot of time to teach the position points when programming a robot, and the programming time is out of the actual production work. Offline programming can be used to utilize, for example, the information contained in the CAD model of a workpiece. However, the software required for offline programming may be too large an investment especially for a small business. This thesis is about using the CAD model design information without the offline programming software.

The cobotics was familiarized by reading literature and online material, and by training with the UR10 robot at the premises of Automation Assistant Oy. The different software that was studied was mainly based on the manufacturers' manuals.

At the end of this thesis a case example where data is collected in SolidWorks software and transferred to the robotic cell interface is presented.

KEYWORDS:

cobotics, SolidWorks, macro, API, CAD model

SISÄLTÖ

KUVAT

KAAVIOT

SANASTO JA LYHENTEET

API	Ohjelmointirajapinta (Application programming interface).
CAD	Tietokoneavusteinen suunnittelu (Computer-aided desing).
Inertia	Kappaleen taipumus jatkaa tasaisessa liiketilassaan, ellei siihen kohdistu kiihtyvyyttä aiheuttavaa voimaa.
Makro	Ohjelmoitu toimintasarja rutiininomaisten tehtävien suorittamista varten.
OLP	Etäohjelmointi (Off-line programming).
Robottisolu	Yhdestä tai useammasta robotista ja sovelluksessa tarvittavista lisä- ja turvalaitteista muodostuva kokonaisuus.
Singulariteettipiste	Paikka, jossa robottikäsivarsi menettää kykynsä siirtää työkalua halutussa asennossa ja halutulla radalla.
Sketsi	Mallinnusohjelmissa piirretty kaksiulotteinen luonnos, eli 2D-sketsi, joka saa kolmannen ulottuvuuden piirteitä luomalla.
Standardi	Jonkin organisaation esittämä määritelmä siitä, miten jokin asia tulisi tehdä.
TCP	Työkalupiste, työkalukoordinaatiston origo (Tool center point).

1 JOHDANTO

Opinnäytetyön toimeksiantaja on Automation Assistant Oy. Vuodesta 2003 toiminut yritys toteuttaa teollisuuden automaatioprojekteja alusta loppuun avaimet käteen -periaatteella. Robottisovelluksia yritys tekee sekä perinteisellä että yhteistyörobotiikalla. Referensseistä löytyy monipuolisesti myös muun muassa teollisuuden testilaitteita, prosessiteollisuuden ohjausjärjestelmiä ja terminaalien matkustajasiltojen automatisointeja.

Tämän opinnäytetyön tavoitteena on tutustua yhteistyörobotiikkaan ja sen ohjelmointiin sekä muototiedon keräämiseen tuotteen CAD-mallista. Opittua tietoa hyödyntämällä opinnäytetyössä toteutetaan esimerkki, jossa CAD-mallista kerättyjä koordinaattitietoja siirretään robottisolun käyttöliittymään. Solu muodostuu Universal Robot UR10 -yhteistyörobotista, kierteityskoneesta ja Indusoft Web Studiolla suunnitellusta käyttöliittymästä.

Yhteistyörobotiikka on verrattain uusi robotiikan suuntaus, ja on vasta hakemassa paikkaansa tuotannon yhtenä osatekijänä. Teollisuusrobotteja on yleisesti pidetty massatuotannon työkaluina hintansa ja korkeaa ammattitaitoa vaativan integrointiprosessinsa vuoksi. Yhteistyörobotiikalla on kuitenkin suunnittelun ja toteutuksen onnistuessa mahdollista taloudellisesti kannattavan robotisointihankkeen läpiviemiseen myös pienessä yrityksessä. Perinteistä teollisuusrobottisolua kevyemmät ja edullisemmat turvajärjestelmät ja myös maallikolle helpommin lähestyttävä käyttöliittymä avaavat paljon mahdollisuuksia. Ennen kaikkea pienten sarjojen ja joustavan tuotannon näkökulmasta yhteistyörobotit voivat tarjota tilaisuuden automatisointimahdollisuuksien uudelleenarviointiin.

Piensarjojen tuotevaihtotilanteisiin kuluvan ajan vähentäminen on mahdollista esimerkiksi etäohjelmointiohjelmistoilla. Niiden hintataso ja käytönopetteluun kuluva aika voivat olla liiallinen rasitus etenkin pienelle yritykselle. Tämän opinnäytetyön ajatuksena on löytää tapa hyödyntää CAD-malleissa olevaa mittatietoa yhteistyörobotin ohjelmoinnissa ilman uutta ohjelmistoa. Näin on mahdollista välttyä aikaa vievältä ja tarkkuutta vaativalta taluttamalla opettamiselta tuotevaihtojen yhteydessä. Se parantaa robottisolun kannattavuutta, eikä kalliille CAD-mallia hyödyntävälle offline-ohjelmistolle ole välttämättä tarvetta.

Tässä työssä käsitellään aluksi yleisesti yhteistyörobotiikan ominaispiirteitä lähinnä perinteiseen teollisuusrobotiikkaan verrattuna. Tämän jälkeen tutustutaan hieman robotin

ohjelmointiin ja tapaan hyödyntää CAD-mallia ohjelmoinnissa käytettävän paikoitustiedon lähteenä. Lopuksi esitellään esimerkki toimeksiantajan kehitteillä olevan robottisolun tarpeisiin suunnitellusta tiedonkeräys- ja hyödyntämismallista.

Opinnäytetyö on saanut alkunsa toimeksiantajayrityksen tarpeesta, ja se pyrkii sekä ratkaisemaan annetun tehtävän että olemaan myös hyödyksi vastaavanlaisten tilanteiden parissa työskenteleville.

2 YHTEISTYÖROBOTIIKKA

Perinteisesti teollisuusrobotti on työskennellyt aidattuna häkissään. Se on suhteellisen kustannustehokas ja selkeä tapa erottaa ihmisen ja robotin työskentelyalueet toisistaan ja helpottaa riskianalyysin tekemistä merkittävästi. Turvallisuustekniikan kehittyessä ja tuotannollisten vaatimusten kasvaessa on kuitenkin tullut ajankohtaiseksi yhdistää ihmisen ja robotin työskentelyalueita osittain tai kokonaan. (Malm 2008, 2.) Robotti on mahdollista tuoda tehdassaliin työntekijöiden joukkoon ilman aitoja turvallisuuden siitä kärsimättä. Tällaiseen robottiin viitataan englanninkielisellä termillä *collaborative robot* eli *cobot.* Suomeksi voi puhua yhteistoiminta- tai yhteistyörobotista. Tässä opinnäytetyössä käytetään jälkimmäistä termiä.

2.1 Historiaa

Yhteistyörobotiikan juuret juontavat 1990-luvun alkupuolelle. Suuri yhdysvaltalainen autonvalmistaja General Motors alkoi etsiä keinoja parantaa kokoonpanolinjastoilla työskentelevien työntekijöidensä ergonomisia olosuhteita. Autoteollisuus on ollut edelläkävijä teollisuusrobottien käyttöönotossa alusta alkaen, joten oli luontevaa etsiä ratkaisua ongelmaan robotiikasta. Usean yliopiston ja General Motorsin omien työntekijöiden yhteistyöstä syntyi ensin *Intelligent Assist Device (IAD).* Se oli lähinnä voiman tunteva apuväline taakkojen nostamiseen ja sellaisenaan käytössä yhä nykyisinkin. (Pittman 2016.)

Vuosia jatkuneen kehitystyön lopputuloksena Northwestern-yliopiston professorit J. Edward Colgate ja Michael Peshkin jättivät patenttihakemuksen yhteistyörobotista eli cobotista vuonna 1997. Samana vuonna he perustivat myös yrityksen nimeltä Cobotics (Pittman 2016). Colgaten ja Peshkinin suunnittelevat laitteet oli kuitenkin lähinnä tarkoitettu toimimaan ihmisen jatkuvassa ohjauksessa. Niiden autonomia oli vähäistä ja mekaniikka poikkesi vielä paljon nykyisen kaltaisista nivelvarrellisista yhteistyöroboteista.

Kolmen insinöörin Tanskassa perustama Universal Robots vei kehitystä eteenpäin tuomalla markkinoille vuonna 2009 ensimmäisen mallinsa (Universal Robots 2017). Yrityksen nimi on viittaus tieteiskirjallisuuden klassikkonäytelmään *Rossumovi Univerzální Roboti (Rossum's Universal Robots)*, jonka kirjoitti tšekkiläinen kirjailija Karel Čapek (Roberts 2006, 168). Tämä UR5:ksi nimetty kuusinivelinen käsivarsirobotti painaa 18 kg, sen kuormankäsittelykyky on 5 kg ja ulottuvuus 85 cm. Kokoluokka on siis varsin pieni

verrattuna perinteisiin teollisuusrobotteihin. Niistä pienimmätkin painavat satoja kiloja ja vastaavasti kykenevät käsittelemään painavampia kappaleita. UR5 ja sitä seuranneet UR10 ja UR3 synnyttivät osaltaan uuden robotiikan kategorian, jollaisena yhteistyörobotiikka juuri usein nykyisin ymmärretään. On huomionarvoista, että robottien ja ihmisten välinen yhteistoiminta on säädösten puitteissa mahdollista toteuttaa myös perinteisempiä teollisuusrobotteja ja turvallisuusjärjestelmiä oikein integroimalla.

Universal Robotsin jalanjälkiä seurasivat lähes kaikki vakiintuneet robottivalmistajat ja myös täysin uusia toimijoita tuli nopeasti kilpailemaan omilla tuotteillaan. Näistä voi mainita esimerkkinä Rethink Roboticsin kahdella käsivarrellaan ja näyttöruudun grafiikan ilmeillään inhimillistetyn Baxterin. Tällaisia alle 15 kg käsittelykyvyn omaavia yhteistyöhön kykeneviä robotteja on siis nykyisin markkinoilla moneen käyttötarpeeseen (Koukkari 2016, 2). On kuitenkin huomattava, että perinteisen teollisuusrobotiikan tarve ei ole kadonnut minnekään ja asennetuista roboteista edelleen pieni osa on yhteistyörobotteja.

2.2 Periaatteita

Teollisuusrobottien on tyypillisesti mielletty soveltuvan parhaiten suuren volyymin tuottajille kuten autotehtaille. Seuraavat syyt ovat puoltaneet näkemystä (Malm 2008.):

- Robotin integrointi osaksi tuotantoa, eli käyttökohteen suunnittelu ja käyttöönotto, on ollut kallista ja vaatinut erityistä osaamista.
- Ohjelmointi on ollut asiantuntijatyötä.
- Turvajärjestelmä aitoineen on vienyt paljon tilaa ja aiheuttanut kustannuksia.
- Mukautumiskyky pieniin sarjoihin tai laajaan tuotevalikoimaan on ollut heikko.
- Robottisoluun tehtävät muutokset ovat saattaneet vaatia koko tehtaan layoutin uudelleenjärjestelyä ja robotin käyttötarkoituksen muuttaminen on kiinteän asennuksen vuoksi suuren työn takana.

Näihin ongelmiin yhteistyörobotiikan toivotaan tuovan parannusta. Ennen kaikkea pienten ja keskisuurten yritysten kynnys ottaa robotti osaksi tuotantoaan voi madaltua, ja ne voivat saada osansa robotiikan tuomasta kilpailukyvyn parantumisesta, johon vain suurilla valmistajilla on tähän asti ollut mahdollisuus. Joustavuus, käyttöönoton ja ohjelmoinnin helppous, siirreltävyys, ja turvajärjestelmien tarpeettomuus ovat yhteistyörobottival-

mistajien lupauksia, mutta sovelluskohtaisen suunnittelun merkitystä ei voi korostaa liikaa. Kannattavan investoinnin perusta on huolellisesti valmisteltu ja läpiviety integrointiprosessi.

Jotta todella hyödyttäisiin uuden teknologian käyttöönotosta, on ymmärrettävä, mistä yhteistyörobotiikassa on oikeastaan kyse. Aiheeseen liittyy monia väärinkäsityksiä, joista yleisimpinä voidaan mainita seuraavat (TM Robotics 2017):

1. Väite, että yhteisen työtilan ihmisen kanssa voivat jakaa vain niin sanotut yhteistyörobotit, on virheellinen. Robotti itsessään ei takaa turvallista sovellusta. Lähes jokainen robotti taas on riittävin turvallisuusmekanismein kyvykäs yhteistyöhön, jos se täyttää standardin ISO 10218-1:2011 vaatimukset. Suuntaviivat turvallisen sovelluksen suunnitteluun löytyvät ISO:n teknisestä spesifikaatiosta ISO/TS15066:2016.
2. Se, että yhteistyörobotti ei koskaan tarvitse turva-aitaa ympärilleen, on myös virheellinen käsitys. Jokaisen robottisolun suunnitteluun kuuluu olennaisen osana riskianalyysi, jossa määritellään erilaisten turvaratkaisujen tarve. On täysin sovelluskohtaista, tarvitaanko turva-aitauksia vai ei. Kuten edellä mainittiin, robotti on vain yksi osa sovellusta, jonka turvallisuutta tarkastellaan aina kokonaisuutena.
3. Yhteistyörobotin kyvystä työskennellä aina ihmistä selvästi nopeammin ja tehokkaammin on myös harhakuvitelmia. Ihmisen ja robotin jakama työskentelyalue johtaa yhteentörmäyksen riskin kasvamiseen. Tällaisen kontaktin ihmiselle vahinkoa aiheuttavaa vaikutusta pienennetään robotin liikenopeutta vähentämällä tuottavuuden kustannuksella. Yhteistyörobotin ohjelmointi taas esimerkiksi ”talluttamalla” voi maallikon kannalta olla yksinkertaisempaa, mutta robotin ohjelmaan tulee helposti turhia liikkeitä, jotka lisäävät läpimenoaikaa.

On ilmeistä, että turvallisuuden merkitys on entistä suurempi, kun robotin halutaan työskentelevän samalla alueella ihmisen kanssa ilman turva-aitoja. Järjestelmää ja sen turvaratkaisuja suunniteltaessa avuksi tulevat aihepiiriä käsittelevät standardit.

2.3 Turvallisuus

Standardi on yhdenmukainen ja suositeltava toteutus- tai menettelytapa sille, miten sen käsittelemä asia tulisi tehdä. Standardien käyttö on periaatteessa vapaaehtoista, mutta

viranomaiset saattavat edellyttää niiden käyttämistä. Turvallinen ja yhteensopiva järjestelmä on helpoimmin suunniteltavissa standardien avulla. Suomessa standardointia organisoi Suomen Standardoimisliitto SFS ry. Vahvistettavista SFS-standardeista valtaosa on alkuperältään eurooppalaisia tai kansainvälisiä standardeja. Kansainvälinen kattojärjestö alan toimijoille on ISO (International Organization for Standardization), ja sen jäseniä ovat kansalliset standardisoimisjärjestöt 163 maasta. Sähkö- ja telealalla on omat järjestönsä, joista kansainvälisellä tasolla toimivat IEC (International Electrotechnical Comission) ja ITU (International Telecommunication Union) (Suomen Standardoimisliitto ry. 2017).

2.4 Standardi SFS-EN ISO 10218:2011

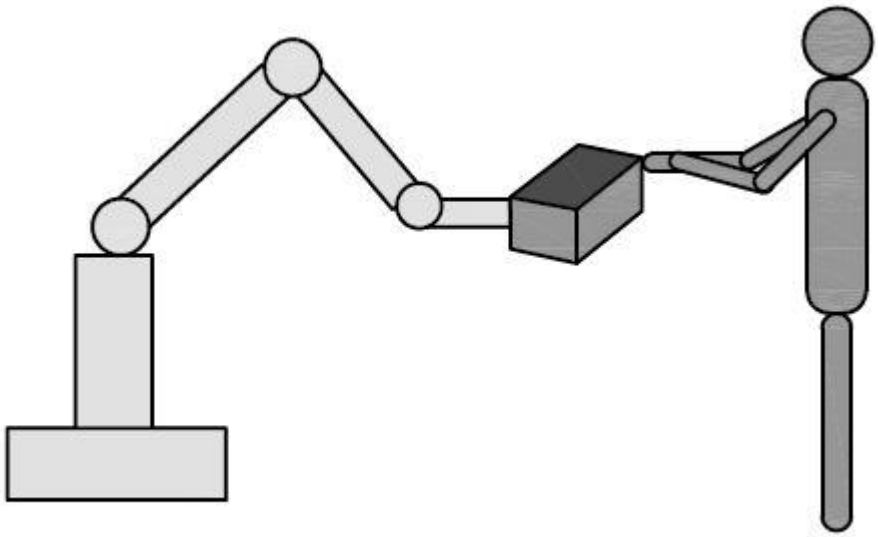
Yhteistyörobotiikka nykymuodossaan on varsin uusi asia. Robottien ja niiden laitteiden turvallisuusvaatimuksia käsittelevä kaksiosainen ISO:n laatima standardi 10218 uudistettiin vuonna 2011. Ensimmäinen osa ISO 10218-1:2011 käsittelee teollisuusrobotteja ja toinen osa ISO 10218-2:2011 robottijärjestelmiä ja integrointia. Standardia kirjoitettaessa kokemuksia yhteistyörobotiikasta oli vielä hyvin rajallisesti. Kokemusten karttuessa nähtiin tarpeelliseksi kirjoittaa tarkentava tekninen spesifikaatio ISO/TS 15066, jossa vaatimukset ovat entistä tarkemmin kohdistettuja. (Robotiq 2016, 3).

2.4.1 Työtila

Standardin ISO 10218-2 osio 5.11 käsittelee yhteistyörobotiikkaa. Kohdassa 5.11.3 on lueteltu ihmisen ja robotin jakaman yhteisen työtilan vaatimuksia (SFS-EN ISO 10218-2. 2011.):

- Tila, jossa suoraa vuorovaikutusta voi tapahtua, on merkittävä selvästi esimerkiksi lattiaan maalaamalla tai huomiokyltein.
- Työntekijä/operaattori on suojattava turvalaitteiden ja standardissa ISO-10218-1 hyväksyttyjen robotin turvaominaisuuksien yhdistelmällä kohdan 5.2.2 mukaisesti. Sen mukaan tehon syötön menettäminen tai tehon syötössä tapahtuvat muutokset eivät saa aiheuttaa vaaraa. Tämä koskee kaikkia tilassa työskenteleviä.
- Tila pitää suunnitella siten, että operaattori voi vaivattomasti suorittaa tehtävänsä. Koneiden ja laitteiden sijoittelusta ei saa aiheutua ylimääräistä vaaraa.

- Robotin käsivarren suurimman ulottuman ja kiinteiden esteiden väliin tulisi jäädä vähintään 500 mm, jotta puristumisvaaralta vältyttäisiin.
- Kuvassa 1. on yhteistyörobotin merkinnässä käytettäväksi suositeltu symboli.



Kuva 1. Yhteistyörobotin symboli (SFS-EN ISO 10218-2. 2011).

2.4.2 Yhteistyön toteuttaminen

Osiossa 5.11.5 esitellään neljä turvaominaisuutta, jolla yhteistyösovelluksen voi toteuttaa. Minkä tahansa ominaisuuden vikaantumisen on aiheuttava suojapysäytys kohdan 5.3.8.3 mukaisesti, jolloin robotin autonominen toiminta on käynnistettävissä uudelleen vain toiminta-alueen ulkopuolelta. Ominaisuudet ovat seuraavat (Bélanger-Barrette 2016, 13):

1. Turvaluokiteltu valvottu pysäytys, jossa robotin on pysähdyttävä, kun ihminen on yhteisessä työtilassa. Pysäytystoiminnon on täytettävä standardin ISO-10218-1 vaatimukset. Ihmisen poistuessa robotti voi jatkaa automaattista toimintaa.
2. Käsin ohjaaminen.
3. Nopeuden ja vähimmäisetäisyyden valvonta, jossa työtila on jaettu eri vyöhykkeisiin. Henkilön tullessa lähivyöhykkeelle robotin on ylläpidettävä määritettyä nopeutta ja vähimmäisetäisyyttä käyttäjästä. Häiriön havaitsemisen nopeuksien tai etäisyyden ylläpidossa on aiheutettava suojapysäytys.

4. Tehon ja voiman rajoittaminen luontaisesti turvallisella suunnittelulla tai ohjauksella, jolloin rajoitustoiminnon on oltava standardin ISO 10218-1 kohdan 5.4 mukainen. Jos jokin parametriraja ylittyy, seurauksena on oltava suojapysäytys. Kohta neljä määrittää turvaominaisuuden, johon nykyisten yhteistyörobottien toiminta pääasiassa perustuu. Niiden ohjausjärjestelmä kykenee tunnistamaan epänormaalin toiminnan aiheuttaman voiman vaikutuksen runkoonsa. Näissä kontaktitilanteissa robotti pyrkii aktiivisin toimin vähentämään iskun suuntaista inertiaansa. Osa robottimalleista pysähtyy ja osa vetäytyy iskusuunnasta poispäin. Vaihtoehdoista ainoastaan teho- ja voimarajoitetuissa järjestelmissä robotin ja ihmisen välinen toiminnallinen kontakti on siis mahdollinen.

2.5 ISO/TS 15066

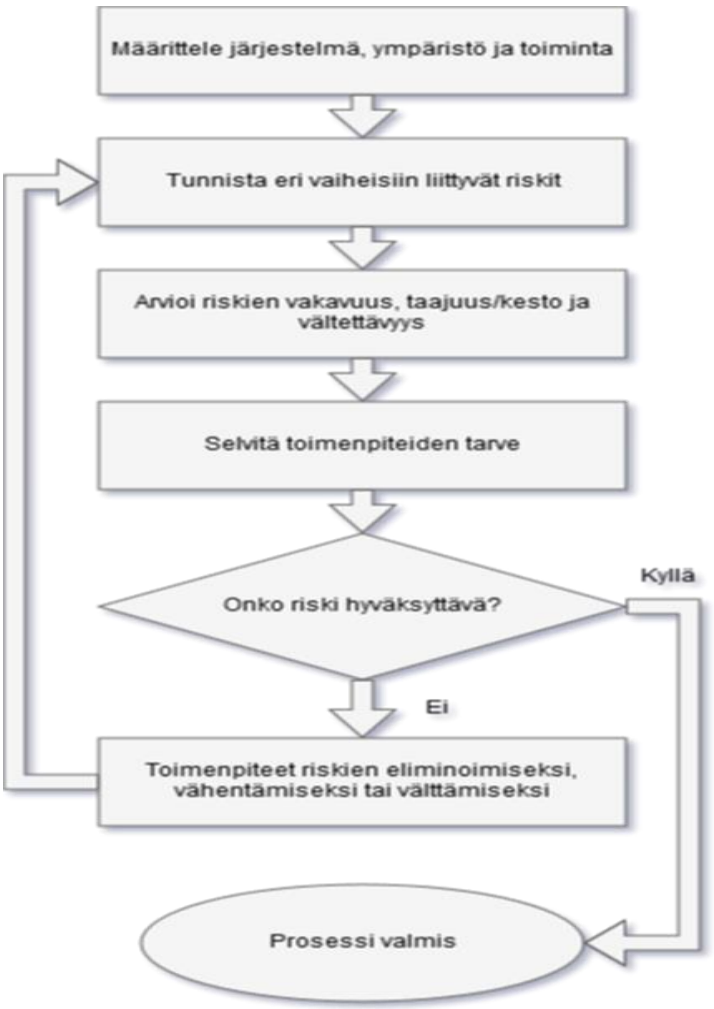
Standardia ISO 10218:2011 täydentävän teknisen spesifikaation ISO/TS 15066 eräs keskeisistä ajatuksista on, että kontaktitilanne ei saa aiheuttaa ihmiselle kipua eikä vammoja. Tämän välttämiseksi se tarjoaa kvantitatiivista tietoa suurimmasta voiman ja paineen suuruudesta, jota kehon eri osiin voi kohdistaa tuottamatta kipua tai vaurioita. ISO/TS 15066 auttaa myös määrittelemään suurimmat mahdolliset nopeudet, jolla robottia voi turvallisesti käyttää. Myös eri yhteistyötekniikoiden suunnittelukriteereitä tarkennetaan standardiin ISO 10218:2011 nähden, ja ISO/TS 15066 liitettäneen jossain muodossa osaksi varsinaista standardia vuonna 2017, jolloin se otetaan uudelleen tarkasteluun. (Bélanger-Barrette 2016, 4,6.)

Robottijärjestelmän suunnittelussa olennainen vaihe on riskianalyysin tekeminen. Sen painoarvo kasvaa entisestään yhteistyörobotiikan tullessa kuvaan, ja ISO/TS 15066 on tässä prosessissa tärkeä työkalu.

2.6 Riskianalyysi

Riskianalyysiprosessissa on tarkoitus tunnistaa ja arvioida tilanteeseen liittyvä riskejä, suhteuttaa niitä vertailuarvoihin ja standardiin, sekä määrittää hyväksyttävä riskitaso. Robotti- ja komponenttivalmistajat vakuuttavat CE-merkinnällä noudattavansa EU:n koneturvamääräyksiä ja monet ovat myös sertifioineet tuotteensa puolueettoman laitoksen toimesta. Jokainen turvallisiksi luokitelluista osista rakennettu teollinen sovellus on kui-

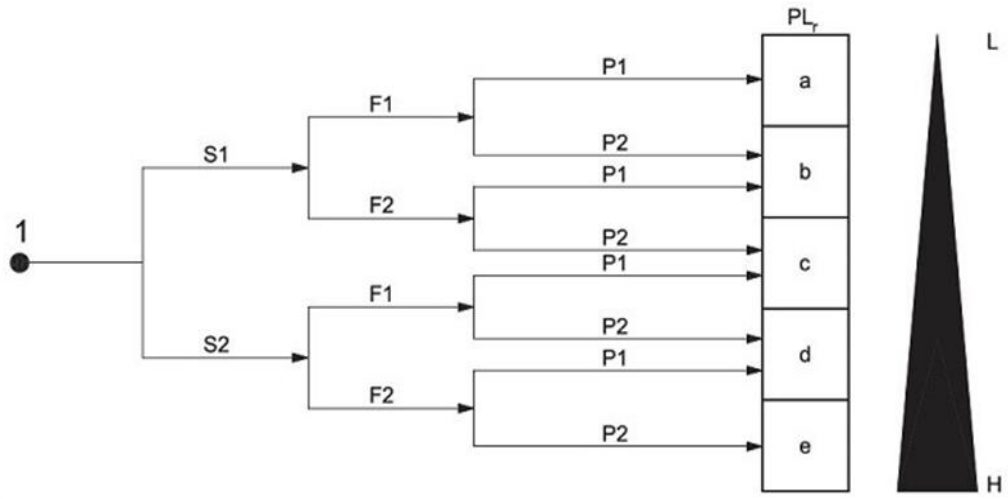
tenkin erillinen tapauksensa ja siitä syystä riskianalyysi on välttämätön (Bélanger-Barrette 2016, 9). Robottijärjestelmää tarkastellaan myös koneturvallisuuden näkökulmasta yhtenä koneena. On siis olennaista, että joku vastaa tästä kokonaisuudesta ja antaa siitä vaatimustenmukaisuusvakuutuksen (Malm 2008, 12).



Kuvio 1. Riskianalyysiprosessi.

Kuviossa 1. kuvattu riskianalyysiprosessi alkaa perinpohjaisella perehtymisellä sovellukseen ja asennusympäristöön. Vain siten on mahdollista tunnistaa kaikki riskit järjestelmän asennusvaiheesta loppukäyttöön.

Riskien vakavuutta, toistuvuutta ja vältettävyyttä arvioitaessa voidaan käyttää standardin SFS-EN ISO 13849-1 liitteessä A olevaa riskin arviointimenetelmää, joka esitetty kaaviossa 2.



Selite

1	Aloituskohta turvatoiminnon osuudenarvioimiseksi riskin pienentämisessä
L	Osuus riskin pienentämisessä pieni
H	Osuus riskin pienentämisessä suuri
PL_r	Vaadittava suoritustaso
Riskimuuttujat:	
S	Vamman vakavuus
S1	Lievä (tavallisesti palautuva vamma)
S2	Vakava (tavallisesti palautumaton vamma tai kuolema)
F	Vaaralle altistumisen taajuus ja/tai kesto
F1	Harvoin...toisinaan ja/tai lyhyt altistumisaika
F2	Toistuvasti...jatkuvasti ja/tai pitkä altistumisaika
P	Mahdollisuus välttää vaaraa tai rajoittaa vahinkoa
P1	Mahdollista tietyissä olosuhteissa
P2	Tuskin mahdollista

Kuvio 2. Riskinarviointimenetelmä selityksineen.

Tässä menetelmässä polkua seuraamalla päädytään oikealla olevaan vaadittavan suoritustason (Performance Level, PL) kohtaan ”a…e”. Kirjaintunnus kertoo turvallisuuteen liittyvältä ohjausjärjestelmältä vaadittavan kyvyn suorittaa turvatoiminto ennakoitavissa olevissa olosuhteissa. A-luokan vaatimustaso suorituskyvylle on matalin ja vastaavasti e-luokan taso korkein. Nämä tasot vastaavat soveltuvin osin IEC 62061 -standardin SIL-

tasoja, mikä helpottaa sovelluksessa käytettävien komponenttien valintaa. (Hietikko ym. 2009.) Kaaviossa 2. oikealla olevan mustan kolmion voidaan ymmärtää myös kuvaavan riskin vakavuutta. Riskin vakavuusaste kasvaa alaspäin mentäessä. Yleisesti ottaen suoritustason a ja b vaatimat riskit ovat riskianalyysiprosessisissa hyväksyttäviä (Bélanger-Barrette 2016, 10).

3 OHJELMOINTI

Käsivarsirobotin ohjelmoinnissa laaditaan halutussa järjestyksessä toteutettava lista tehtävistä, jotka käsivarteen kiinnitetyllä työkalulla suoritetaan sovelluksen vaatimalla tavalla. Tehtävät voivat olla mitä moninaisimpia kappaleen tarkastamisesta pakkaamiseen ja hitsaamisesta maalaamiseen. Ohjelmointiin liittyy olennaisesti myös keskustelu ulkoisten laitteiden, kuten työstökoneiden tai kuljettimien kanssa. Ohjelman tulee sisältää toimintamäärittelyt myös häiriötilanteissa. Teollisuusrobotteja ohjelmoidaan nykyisin pääasiassa opettamalla tai etäohjelmoimalla.

3.1 Opettamalla ohjelmointi

Perinteisesti robotin liikerata määritetään siirtämällä käsiohjainta käyttäen robotin käsivartta ja siihen liitettyä työkalua liikeradan muutoskohdasta toiseen. Nämä pisteet nimetään ja tallennetaan robottiohjaimen muistiin. Se interpoloi, eli laskee työkalun väliasemat pisteiden perusteella (Kuivanen 1999, 79). Jokaisen nivelen paikka-anturin arvo tallennetaan. Tallennettuja paikoituspisteitä on tämän jälkeen mahdollista käyttää ohjelmassa haluamassaan järjestyksessä ja niiden sisältämiä arvoja voi eri komennoin myös muokata. Robotille on tärkeää luoda yksi kotipiste, josta se aloittaa, ja johon se lopettaa työkiertonsa (Pekkanen 2010). Käsiohjaimen tehokas käyttäminen paikoituspisteiden luomisessa vaatii kokemusta. Robotin opettamalla ohjelmointiin kuluva aika on pois tuottavasta työstä, joten liikeradat pitäisi saada luotua mahdollisimman nopeasti (Kuivanen 1999, 79).

Kun robotille on saatu luotua riittävä määrä paikoituspisteitä mielekkään liikeradan aikaansaamiseksi, aletaan sovelluksessa tarvittavia toimintoja ohjelmoida. Eri robottivalmistajat ovat kehittäneet tätä varten omia komentosarjakieliään. Tällaisia ovat esimerkiksi:

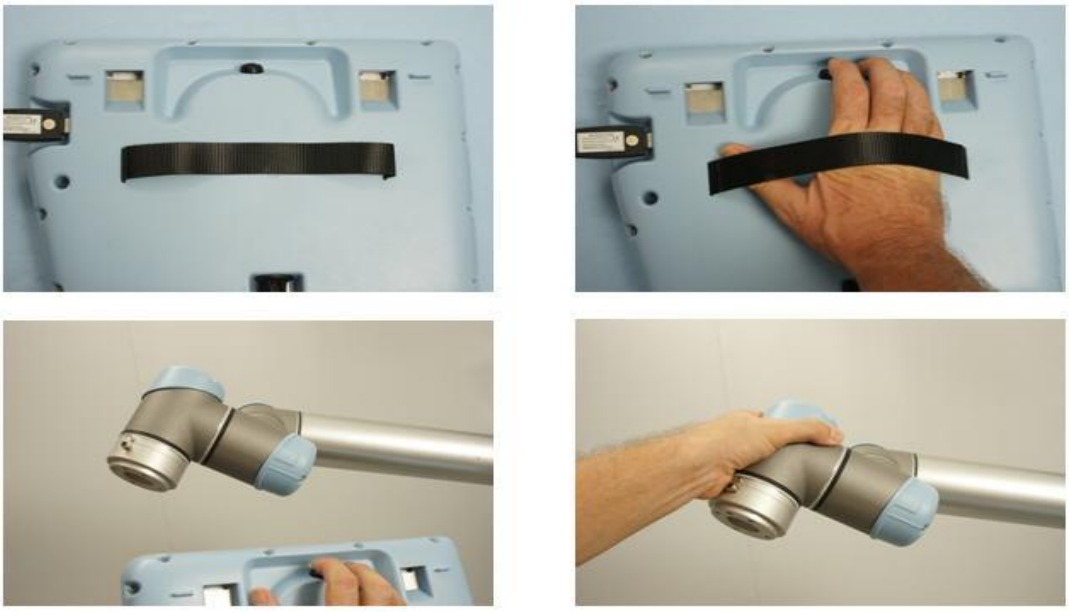
- ABB: Rapid
- Fanuc: TP
- Kuka: KRL
- Yaskawa: Inform
- Universal Robots: URScript

Niiden syntaksi perustuu perinteisiin ohjelmointikieliin (kuten BASIC, Pascal tai C), ja toiset ovat käskykannaltaan laajempia ja monipuolisempia kuin toiset. Käskyjä voi hyvinkin olla jopa 150 (Kuivanen 1999, 80). Näitä käskyjä ja tallennettuja paikoituspisteitä hyödyntämällä rakennetaan haluttu toimintalogiikka kullekin sovellukselle. Opettamalla ohjelmointi on toimiva tapa sovellukselle, jossa robotin ohjelmaa ei ole tarpeen muokata usein. Tällöin opetuksen aiheuttamien tuotannon seisokkien ja varsinaisen tuotannollisen työn suhde pysyy hyvänä.

Liikeratojen luonti käsiohjainta käyttämällä on mahdollista myös yhteistyöroboteilla, mutta niin sanottu taluttamalla ohjelmointi on usein nopeampi tapa, etenkin vähemmän kokeneelle operaattorille.

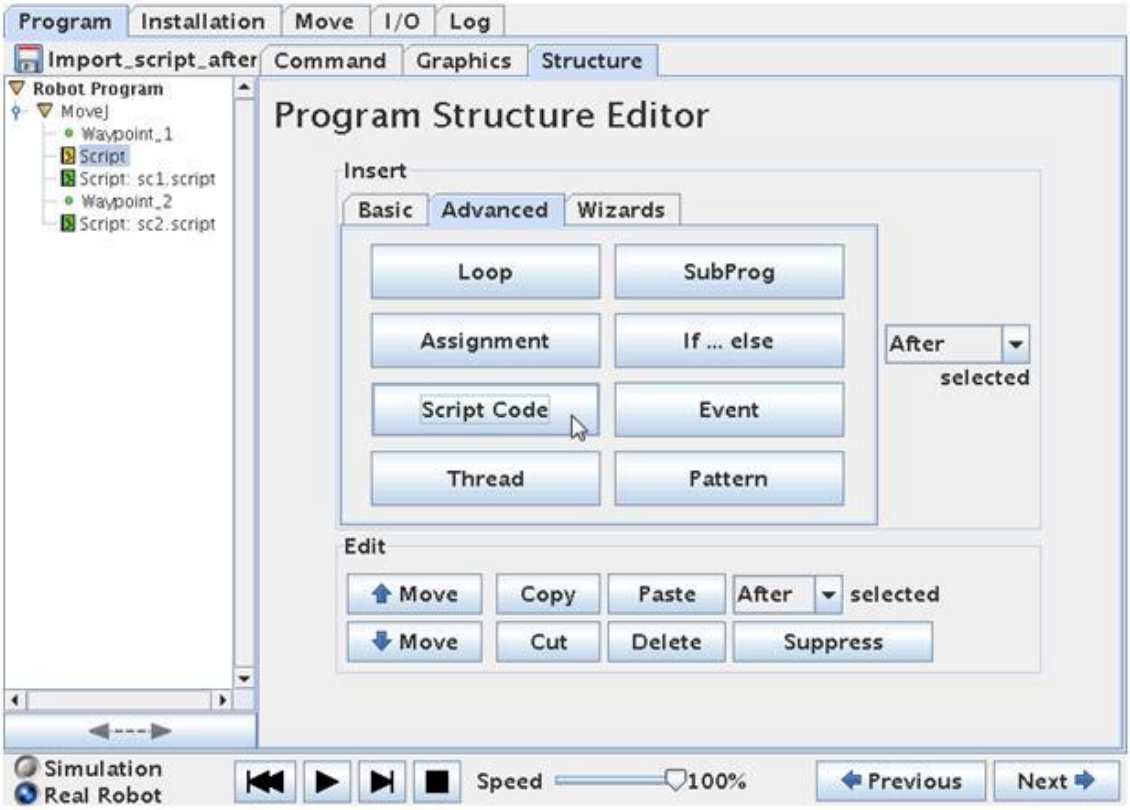
3.2 Taluttamalla ohjelmointi

Voiman tunnistavilla antureilla varustettujen yhteistyörobottien paikoituspisteiden ohjelmointi voidaan toteuttaa myös taluttamalla. Termillä viitataan tapaan, jossa robottikäsivarren servolukot vapautetaan ja käsivarsi siirretään haluttuun asemaan operaattorin voimin. Robotti kannattelee itsensä sekä mahdollisen kuorman, mikäli kuorman massa on määritelty asetuksissa. Liikuttelu on näin ollen varsin kevyttä ja helppoa. Kuvassa 2. näkyy Universal Robotsin käsiohjaimen takana oleva musta painike, joka vapauttaa käsivarren operaattorin liikuteltavaksi.



Kuva 2. Taluttamalla ohjelmointi (Zacobria 2017).

Kun paikoituspiste on luotu, voi ohjelmaan lisätä komentoja haluttujen toimintojen aikaansaamiseksi. Universal Robotsin graafinen käyttöliittymä on nimeltään Polyscope. Kuvassa 3. on Universal robotsin ohjelmaeditori, jota käytetään käsiohjaimen kosketusnäytöltä. (Universal Robots 2015.)



Kuva 3. Universal Robotsin ohjelmaeditori (Zacobria 2017).

Sovelluksen ohjelmointi ja paikoituspisteiden luominen eivät välttämättä vaadi fyysisen robottisolun olemassa oloa, sillä nämä asiat ovat tehtävissä myös simulointimallin avulla tuotannon ulkopuolella etäohjelmointina.

3.3 Offline-ohjelmointi

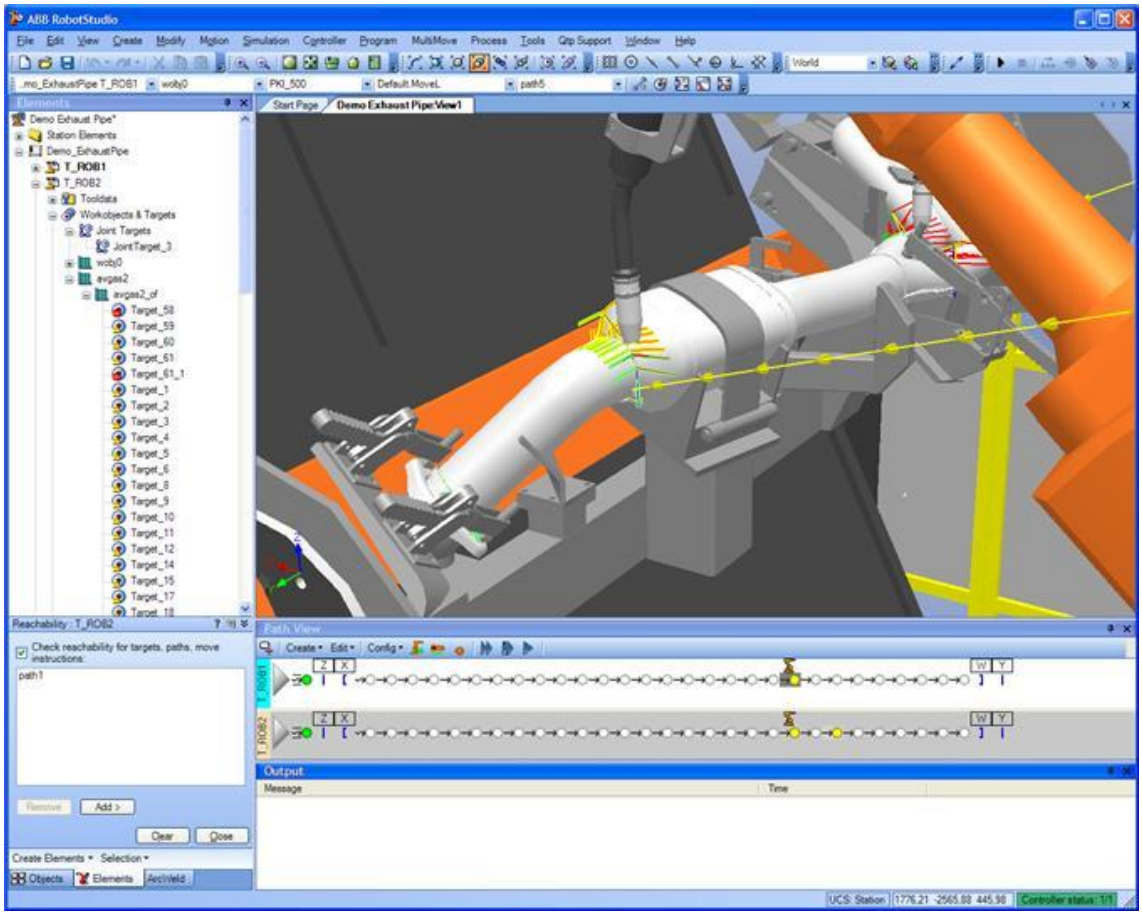
Offline- eli etäohjelmointi (OLP) vapauttaa robotin tuottavaan työhön ohjelmoinnin ajaksi. Näin ollen tuotanto ei merkittävästi kärsi, vaikka tuotantosarjat olisivat pieniä, sillä tuotevaihtoihin kuluva aika lyhenee. Myös uuden robottisolun käyttöönotto nopeutuu, kun ohjelma on valmiiksi tehty. Offline-ohjelmoinnissa käytetään ohjelmistoja, joilla robottiase-

masta rakennetaan kolmiulotteinen simulointimalli ja sitä muokataan graafisessa käyttöliittymässä. Ohjelmista löytyy kirjasto valmiiksi mallinnetuista ja tuotannossa usein käytössä olevia oheislaitteita. Eri robottimalleille on omat kirjastonsa ja robottien simulointimallit vastaavat kinematiikaltaan ja ohjaukseltaan oikeita esikuviaan (Kuivanen 1999, 82). Mikäli kirjastoista ei löydy käytettävien oheislaitteiden 3D-malleja, niitä voi tiedustella laitteen valmistajalta. Toisinaan laite on mallinnettava itse siihen soveltuvalla CAD-ohjelmalla. Robottiasema mallinnetaan todellisiin mittoihinsa ja sen perusteella ohjelman koordinaatistot ja tuotantoon käytettävän robotin koordinaatistot synkronoidaan (Kaarela 2007). Valmis ohjelma on käännettävä käytettävän robotin ymmärtämälle kielelle ja offline-ohjelmistoissa on työkalut tätä toimenpidettä varten.

Sekä etä- että opettamalla ohjelmoinnin perustana on paikoituspisteiden generoiminen liikeratojen luomiseksi. Offline-ohjelmoinnin etuna on mahdollisuus käyttää tuotteen CAD-muototietoja pisteiden määrittämisessä (Pekkanen 2010, 20). Tästä on merkittävää etua sellaisessa valmistusprosessissa, jossa paikoituspisteitä on suuri määrä. Esimerkkeinä tällaisista sovelluksista voidaan mainita hitsaus, maalaus, kiillotus, työstö ja jäysteytys. Mallipohjaisessa ohjelmoinnissa paikoituspisteet liittyvät kappaleeseen ja solun layoutin muuttuessa pisteet seuraavat kappaleen mukana. Näin ollen ohjelmaa ei tarvitse tehdä uudelleen (Kuivanen 1999, 82, 84). Tuotteen ollessa suuri tai vaikeasti lähestyttävä, esimerkiksi hitsattava sauma ahtaassa välissä, on paikoituspisteiden osoittaminen helppoa tietokoneen ruudulla simulointiympäristössä. Offline-ohjelmalla voidaan tutkia robotin ulottuvuuksia, tehdä törmäyksentarkastelua ja simuloimalla määrittää valmiin robottiaseman tahtiaika (Finnrobotics 2017).

Yksi paikoituspiste on usein saavutettavissa monella eri robotin nivelarvon yhdistelmällä. Näistä eri konfiguraatioista ohjelmoijan on valittava tilanteeseen parhaiten sopiva. Varsinaisen ohjelman kirjoittaminen tapahtuu käsky kerrallaan aivan kuten opettamalla ohjelmoinnissakin. Käskyvalikoima ja syntaksi tulee sovellukseen valitun robottimerkin perusteella ja tarjolla on yleisesti ottaen ehto-, rekisterien käsittely-, I/O-, liikkeiden koordinointi- ja prosessikohtaisia käskyjä. Myös yleiskielinen ohjelmointi voi olla mahdollista kääntäjien avulla (Kuivanen 1999, 85).

Ohjelman simulointivaihe on tärkeä virheiden paikallistamiseksi. Parhaassa tapauksessa simuloinnilla saadaan luotua täysin valmis ohjelma ja robotin tuotantotehokkuus optimoitua. Kuvassa 4. on näkymä ABB:n RobotStudion käyttöliittymästä. Kyseisellä simulointi ja offline-ohjelmointiohjelmistolla voi suorittaa koulutus-, ohjelmointi- ja optimointitehtäviä ilman tuotannollista katkoa.



Kuva 4. ABB:n RobotStudio (ABB 2017)

Ensimmäiseksi offline-ohjelmoinnissa on mallinnettava sovelluksen robottisolu työkappaleineen käytössä olevalla ohjelmistolla. Mikäli valmista CAD-mallia ei ole saatavilla, joissain tapauksissa 3D-skanneri saattaa soveltua kappaleen geometrian siirtämiseen digitaaliseksi. Kun tämä on tehty, itse offline-ohjelmointiprosessissa voidaan edetä seuraavin vaihein (Pan 2012, 7.):

1. Sovelluksen vaatimien paikoituspisteiden luominen käsin tai kappaleen piirteitä hyväksikäyttäen. Huomioitavaa on myös työkalun orientaation asettaminen prosessin vaatimalla tavalla.
2. Liikeratojen suunnittelu, ulottuvuuden ja törmäyksen tarkastelu sekä käsivarren konfiguraatioiden valinta on tehtävä ohjelmoijan toimesta, sillä ohjelmiston automaattisesti tarjoamat ratkaisut eivät välttämättä ole optimaalisia.
3. Kolmannen vaiheen tehtäviä ovat I/O-pisteiden lisääminen, liikeratojen ja toimintojen hienosäätö ja ohjelman mahdollisesti vaatima kääntäminen robottiohjaimen ymmärtämään muotoon.

4. Seuraavana on sovelluksen simulointi, joka on offline-ohjelmoinnin suureksi eduksi edelleen tehtävissä ilman fyysistä robottia.
5. Viimeisenä kalibroidaan ja asetetaan solun offline-ohjelman pisteet vastaamaan todellisen maailman kiintopisteitä, ja tarkastetaan, ettei poikkeavuuksia geometrioiden välille ole jäänyt

Offline-ohjelmoinnilla on mahdollista välttää monia muilla ohjelmointitavoilla syntyviä virheitä. Viimeistään simulointivaiheessa paljastuvia virheitä voivat olla esimerkiksi seuraavat (Malm 2008, 99.):

- robotin väärät parametrit (esim. liikemuoto tai koordinaatisto)
- robotin käsivarren singulariteettipisteen lähellä tapahtuvat yllättävät liikkeet
- virheet, jotka syntyvät robotin oikaistessa reittiä
- robotille ohjelmoitu mahdoton reitti
- ohjelmamuutoksista johtuvat virheet
- ohjelmoijan suunnitelmasta poikkeava toiminta

Eri ohjelmointitapojen erot saattavat jatkossa hämärtyä uusien metodien syntyessä ja erilaisia komponentteja yhdisteltäessä. Lisätyn ja virtuaalitodellisuuden tuomat mahdollisuudet voivat jatkossa mullistaa monen muun alan ohella myös robottiohjelmointia. Yritykset kehittävät jatkuvasti tehokkaampia ja nopeampia tapoja luoda monimutkaisenkin sovelluksen vaatimaa ohjelmakoodia. (Pan 2012, 8,9.)

4 MUOTOTIEDON KERÄÄMINEN CAD-MALLISTA

Yhteistyörobottien yleistyminen voi osaltaan mahdollistaa robotiikan käyttöönoton myös pienten ja keskisuurten eli Pk-yritysten tuotannossa. CAD-ohjelmistojen käyttäminen suunnittelussa taas on arkipäivää jo monen pienemmän toimijankin osalta. Tuotteiden suunnittelussa on siirrytty piirustuspöydiltä näyttöruuduille ja tuotteesta on yhä useammin saatavilla CAD-malli. Halu hyödyntää tuon mallin sisältämää muototietoa yhteistyörobottia ohjelmoitaessa on varmasti lisääntymässä. OLP-ohjelmistojen mallipohjaisella ohjelmoinnilla tuo tieto on otettavissa käyttöön, mutta kyseisten ohjelmistojen yleistymistä on haitannut ainakin teollisuusrobottien ohjelmointikielien kirjavuus. CNC-koneiden ohjelmoinnissa käytettävien CAM-ohjelmistojen kehittämistä on helpottanut koneiden ohjauksessa yleisesti käytössä oleva G-koodi (Foit 2017). Teollisuusroboteilla tällaista yhteistä kieltä ei ole. OLP-ohjelmistojen käyttöönottoa vaikeuttavat myös esimerkiksi seuraavat asiat (Neto 2013.):

- Ohjelmiston ostaminen ja tehokkaaseen käyttämiseen vaadittava koulutus voivat olla Pk-yritykselle merkittävä investointi.
- Robottisolusta on luotava tarkka 3D-malli.
- Kalibrointi vaatii ammattitaitoa.

Yksinkertaisen prosessin tuotevaihdossa voi riittää pelkkä robottiohjelman päivittäminen tuotantoon tulevan kappaleen työstössä tarvittavilla paikoituspisteillä. Esimerkkinä voi mainita vaikkapa reikien poraamisen tasomaisen kappaleeseen. Kalliin OLP-ohjelmiston hankkiminen kyseiseen tarkoitukseen voi olla kannattamatonta, mikäli tarvittava tieto on helposti saatavissa suoraan jo olemassa olevasta tuotteen CAD-mallista. Tarkoituksena on siis kerätä reittipisteitä CAD-mallista ja sijoittaa niitä yhteistyörobotin ohjelmaan, jotta vältyttäisiin aikaa vievältä paikoituspisteiden ohjelmoinnilta taluttamalla. Ajansäästö voi olla merkittävä, jos tuotevariantteja ja paikoituspisteitä kuten porattavia reikiä on paljon.

4.1 CAD-malli tiedonlähteenä

3D CAD-mallinnusohjelmistoja on tarjolla paljon, ja yleisesti käytössä olevia ovat esimerkiksi SolidWorks, Catia, Autodesk Inventor, Pro/Engineer, Solid Edge ja Vertex. Tässä opinnäytetyössä perehdytään tarkemmin SolidWorksiin, mutta samat perusperiaatteet pätevät moneen muuhunkin ohjelmistoon.

Useimmin käytettyjä tapoja tiedon keräämiseen CAD-mallista on kolme (Foit 2017.):

1. CAD-ohjelmiston sisällä toimiva makro tai liitännäissovellus kerää halutta tietoa ja tallentaa sen määrättyyn kohteeseen.
2. CAD-ohjelmiston ulkopuolella toimiva itsenäinen sovellus hakee halutut tiedot mallista.
3. CAD-malli muutetaan yleisesti luettavaan tiedostomuotoon, ja siirretään erilliseen ohjelmaan käsiteltäväksi.

Eri tapoja voi myös yhdistellä halutun lopputuloksen saavuttamiseksi. Tämän opinnäytetyön esimerkissä käytetään ensimmäistä tapaa. Jotta CAD-mallin sisältämää tietoa saataisiin kerättyä suunnitteluohjelmistosta, tarvitaan avuksi ohjelmointirajapintaa.

4.2 Ohjelmointirajapinta (API)

Ohjelmointirajapinta, eli API (Application programming interface) voidaan ymmärtää välikädeksi kahden ohjelman keskustellessa keskenään. Kaupallisten ohjelmistojen selkäranka on lähdekoodi, ja se halutaan pitää salaisena. Ohjelmistojen kehittäjät tarjoavat kuitenkin usein mahdollisuuden hyödyntää tai jakaa omien ohjelmiensa toiminnallisuutta jonkun toisen kehittäjän ohjelman kanssa. Tähän tarkoitukseen tarvitaan välittäjä ja sitä kutsutaan nimellä API. Esimerkkinä voidaan käyttää verkkokaupan ohjelmaa, joka keskustelee API:n välityksellä verkkopankin ohjelmiston kanssa. (Roos 2017.)

Monet suunnitteluohjelmistot tarjoavat rajapinnan suunnittelijan käyttöön, jotta työtä voisi automatisoida. Usein toistuvien tehtävien suorittaminen manuaalisesti kuluttaa turhaan aikaa ja altistaa suunnittelutyön inhimillisille virheille. On siis järkevää luoda ja ottaa käyttöön tällaiset tehtävät suorittava sovellus.

4.3 SolidWorks 3D CAD

SolidWorks 3D CAD on suunnittelussa käytettävä parametrinen 3D-mallinnusohjelma. Parametrisella viitataan mallinnustyyliin, jossa ensin piirrettävä kaksiulotteinen luonnos eli sketsi saatetaan erilaisia toimintoja käyttämällä kolmiulotteiseksi. Toiminnoista syntyvät kappaleen piirteet, jotka tallentuvat piirrepuuhun (Tikka 2017). Tämä useiden mallinnusohjelmien piirrepohjaisuus on suuri etu, sillä piirteet säilyvät mallin rakenteena, vaikka arvoja myöhemmin muokattaisiinkin. Näin ollen geometria päivittyy kuvaan, eikä

synny tarvetta aloittaa mallinnusta alusta. Tavallisten 3D-kappaleiden lisäksi SolidWorks-ohjelmistolla voi mallintaa esimerkiksi kokoonpanoja, ohutlevymalleja, profiilirakenteita ja muotteja.

4.4 SolidWorks API

SolidWorksin API-rajapinta on todella monipuolinen tarjoten paljon työkaluja toimintojen automatisointiin sekä varsin laajan ohjeistuksen aiheesta esimerkkeineen. Ohjelmiston API-sovellukset voidaan jakaa kahteen ryhmään:

1. Prosessin sisäiset sovellukset, joita ovat:
    - VBA-makrot (VBA)
    - VSTA-makrot (C#, VB.NET)
    - Add-ins-, eli liitännäissovellukset (C++, C#, VB.NET)
2. Prosessin ulkoinen sovellus, joka on:
    - Stand-alone-, eli itsenäinen sovellus (C#, VB6, VB.NET, C++)

Sulkumerkkien sisälle sovelluksen perässä on sen kirjoittamiseen käytetyt ohjelmointikielet. Kaksi ryhmää eroaa toisistaan siten, että prosessin ulkoinen sovellus on nimensä mukaisesti itsenäinen, eikä tarvitse SolidWorks-ohjelman käynnissä oloa toimiakseen. Ensimmäisen ryhmän sovellukset taas toimivat SolidWorks-prosessin sisällä. (Rice 2014). Itsenäiset sovellukset ovat .exe-tiedostoja, jotka käynnistetään kuten mikä tahansa Windows-sovellus. Liitännäissovellukset taas ovat .dll-tiedostoja, ja niiden avaaminen onnistuu SolidWorksin Tools/Add-ins-valikon kautta. (Dassault Systèmes 2017.)

Liitännäissovellusten luomista varten on SolidWorks-ohjelmiston mukana tulevassa API SDK-paketissa Add-In Wizard-työkalu. Add-In Wizardin mallipohja auttaa esimerkiksi käyttöliittymän rakentamisessa ja käyttöjärjestelmään liittymisessä (Leiniö 2013).

Yleisesti ottaen voidaan todeta, että mitä laajemmasta ja monimutkaisemmasta sovelluksesta on kyse, sitä alemmaksi edellä esitetyssä sovellusjaottelussa kannattaa mennä. Jaottelussa ensimmäisenä ovat VBA-makrot, ja niitä pidetään yksinkertaisimpana ja helpoimmin omaksuttavana tapana tutustua SolidWorksin API-ympäristöön. (Rice 2014.)

4.5 VBA-makrot

VBA tulee sanoista Visual Basic for Applications, ja on Microsoftin sovellusohjelmissa käytetty makrojen kirjoittamiseen tarkoitettu kieli. Se on kehittynyt Microsoftin toimisto-ohjelmissa käytetyistä makrokielistä ja on korvannut ne. VBA-kieltä pidetään vanhentuneena, mutta se on edelleen varsin yleisesti käytössä. Yksi merkittävä syy tähän on VBA:lla kirjoitettujen ohjelmointiesimerkkien valtava määrä. Ongelmatilanteessa aihetta sivuava esimerkkiratkaisu voi olla etenkin kokemattomalle suunnittelijalle ratkaisevan hyödyllinen. Microsoft on kuitenkin kehittänyt jo vuonna 2002 julkaistun uuden .NET-kehykseen ja oliopohjaisuuteen perustuvan VB.NET-ohjelmointikielen, joka tulee korvaamaan VBA-kielen. SolidWorksissä VB.NET on käytettävissä paitsi VSTA-makrojen kirjoittamisessa, myös muissa sovellustyypeissä VBA-makroja lukuun ottamatta. Uudet .NET-pohjaiset kielet ovat etenkin monimutkaisemmissa sovelluksissa vanhaa VBA:ta vaivattomampia. (Rice 2014.)

Makrot itsessään ovat ohjelmoituja toimintasarjoja, ja ne helpottavat usein toistuvien, rutiininomaisten tehtävien suorittamista. Makro voi olla yksinkertainen sarja komentoja, jotka suoritetaan nimellä tai näppäinyhdistelmällä kutsuttaessa. Toisaalta se voi kasvaa kokonaiseksi sovellukseksi asti.

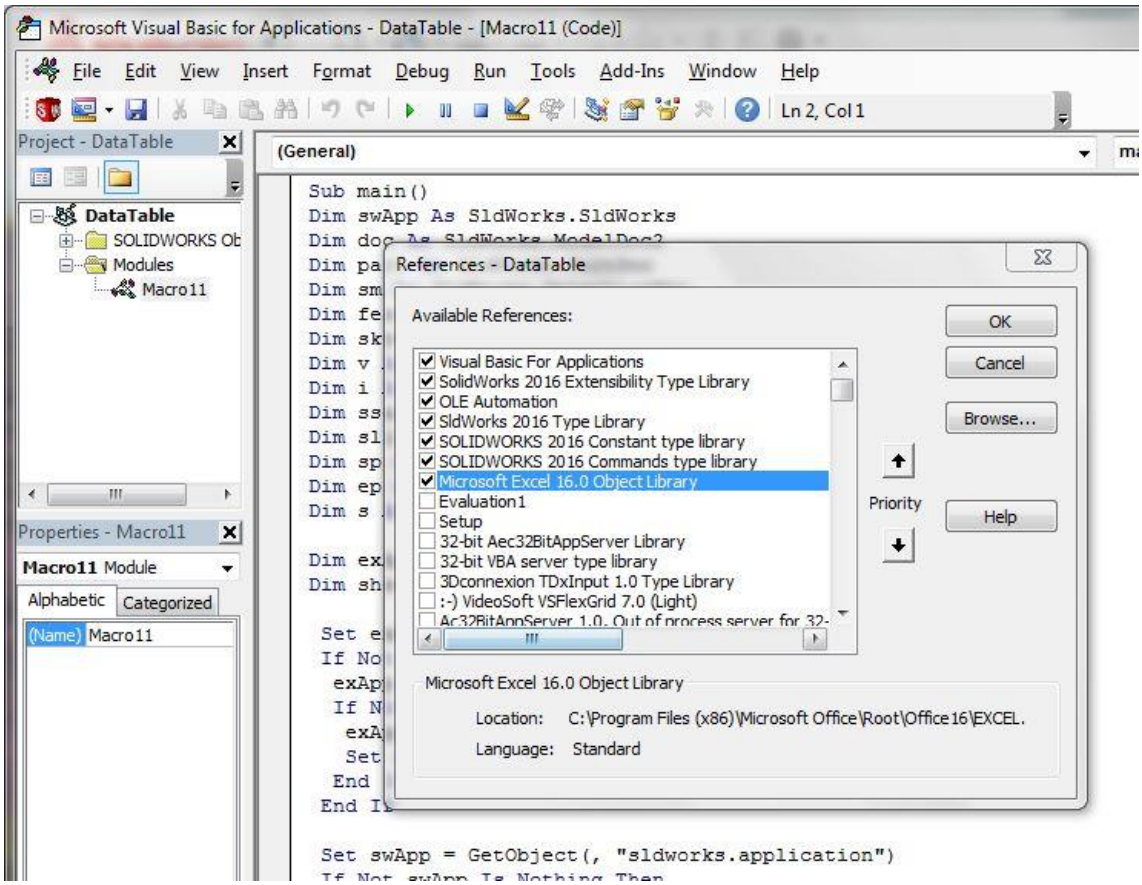
SolidWorksissä on mahdollista luoda makroja yksinkertaisesti nauhoittamalla käyttäjän toimintaa. Toimenpiteet, joista makron haluaa luoda, kannattaa suunnitella huolellisesti etukäteen. Kun makro on nauhoitettu, sitä pääse editoimaan ohjelmointikielisenä, ja poistamaan tai muokkaamaan haluamiaan rivejä. Työkalut eri toimintoihin löytyvät Tools/Macro-valikosta. Työkalupalkkiin voi luoda myös oman pikakuvakkeen makron käynnistämistä varten. VBA-kielellä nauhoitetut makrot tallentuvat .swp-päätteisinä tiedostoina (Dassault Systèmes 2017).

Nauhoittamisessa on kuitenkin omat rajoitteensa, eikä kaikkia toimenpiteitä saa tallennettua makroon nauhoittamalla. Myös esimerkiksi ehdollisten lauserakenteiden nauhoittaminen on luonnollisesti mahdotonta, joten makrojen tehokas hyödyntäminen edellyttää ohjelmakoodiin perehtymistä. Eräs tapa käyttää nauhoittamistoimintoa on ohjelmointikomentojen opettelu. Nauhoittamalla haluamansa toiminnon, ja tarkastelemalla sitä ohjelmointieditorilla näkee ainakin yhden vaihtoehtoisen tavan toteuttaa kyseinen toiminto ohjelmointikielellä. (Rice 2014.)

4.6 VBA ja API

4.6.1 Tyyppikirjastot

Tyyppikirjastot (type library) sisältävät makron ohjelmoinnissa käytettyjen komentojen toiminnallista koodia, metodeja sekä rajapintoja (Leiniö 2013). Kuvassa 5. näkyy VBA-editorin Tools-valikosta löytyvä References-alavalikko ja siitä löytyvät kaikki valittavissa olevat tyyppikirjastot.

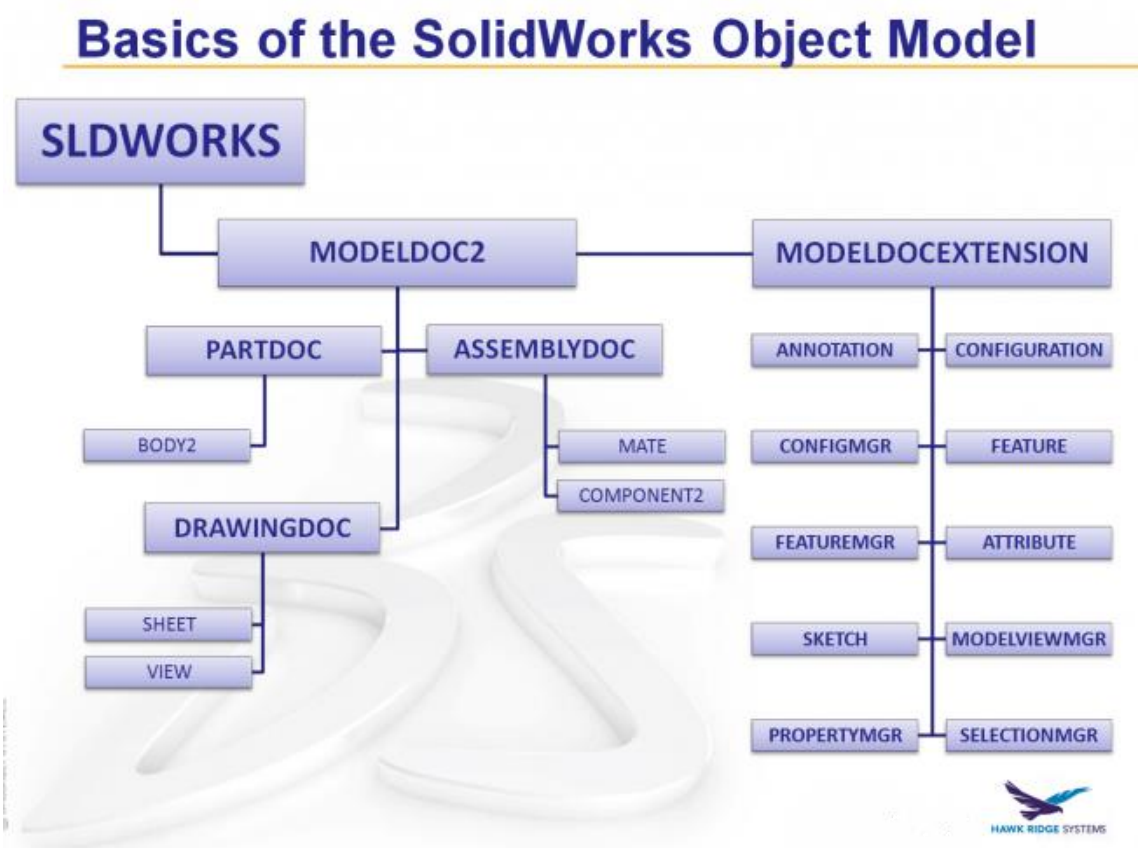


Kuva 5. VBA-editorin References-valikko.

Jotta makrosovelluksella olisi pääsy kirjastoon, siihen on tehtävä viittaus (set reference) valitsemalla se listasta. Luvun 5. esimerkissä hyödynnetään tyypillisten kirjastojen lisäksi Microsoft Excel-tyyppikirjastoa, kuten kuvan 5. valintapalkista huomataan. Näin esimerkin makrosovelluksella on mahdollisuus käyttää rajapintaa Microsoft Excel-toimintoihin, joita kutsutaan objekteiksi.

4.6.2 Objektit ja muuttujat

Eri ohjelmistojen lukuisat toiminnot löytyvät valtavan kokoisista API-objektimalleista. Nämä hierakkiset alaspäin kasvat puurakenteet alkavat ylimpänä olevasta applikaatio-objektista, ja alempana oleviin objekteihin pääsee käsiksi joko suoraan tai epäsuoraan. Kuvassa 6. näkyy SolidWorksin objektimallin erittäin riisuttu perusrakenne. (Dassault Systèmes 2017.)



Kuva 6. SolidWorksin objektimalli (Hawk Ridge Systems 2012).

Objektiin liittyvät termit metodi (method) ja ominaisuudet (properties). Usein käytetty analogia on puhua objektista substantiivina, metodista verbinä ja ominaisuuksista adjektiiveinä. Toisin sanoen metodit ovat niitä toimintoja, joita objektille voi suorittaa ja ominaisuudet ovat määritteleviä attribuutteja (Gomez 2017).

Suuri osa kirjoitetusta ohjelmakoodin riveistä suorittaa yhtä tai useampaa seuraavista toimenpiteistä (MLC CAD Systems 2014.):

- Määrittää muuttujan.

- Asettaa muuttujalle arvon.
- Käyttää muuttujaa.

Objektia käyttääkseen on määriteltävä objektimuuttuja (object variable), ja muuttuja määritetään komennolla *Dim As.* esimerkissä tällaiset objektimuuttujan määrittelyjä Excel-objektimallista ovat rivit:

```
Dim exApp As Excel.Application

Dim sheet As Excel.Worksheet
```

SolidWorksin objektimallissa on satoja rajapintoja sekä toimintoja, mutta yleisesti käytettyjä ovat ainakin seuraavat (Hawk Ridge Systems 2012.):

1. SldWorks
    - Ylimmän tason rajapinta
    - Lähes kaikki toimenpiteet vaativat ensin tämän rajapinnan määrittämisen
2. Modeldoc2
    - SdlWorks-rajapinnan alla
    - Antaa makrolle pääsyn ominaisuuksien muokkaamiseen ja alemman tason toimintoihin.
3. FeatureManager
    - Piirrepuun muokkaamiseen
    - ModelDoc2-objektin alla
4. SketchManager
    - Mahdollistaa sketsin muokkaamisen makron komennoilla

Kielioppisäännöiltään VBA on hyvin lähellä Visual Basic-ohjelmointikieltä, josta löytyy paljon laadukasta oppimateriaalia etenkin englannin kielellä.

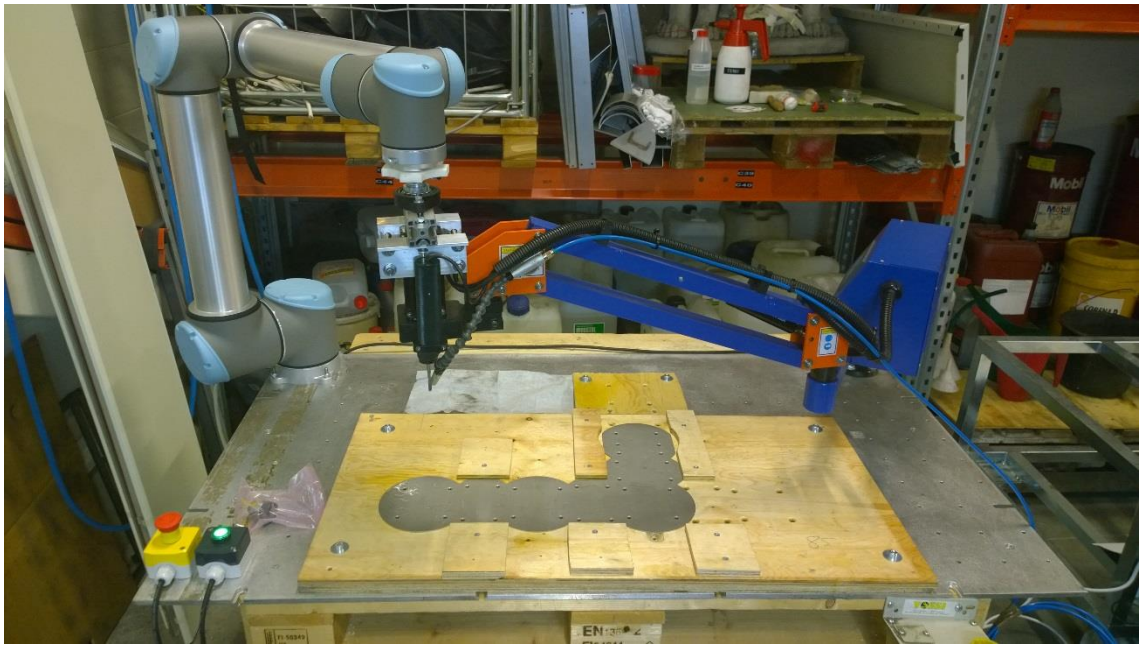
5 ESIMERKKI

Tässä luvussa tarkastellaan esimerkin avulla CAD-mallin muototiedon hyödyntämistä yhteistyörobotin ohjelmoinnissa.

5.1 Lähtökohta

Robottisolu muodostuu Universal Robot UR10 yhteistyörobotista ja kierteistyskoneesta, joka oli laakeroidulla adapterilla liitetty robotin työkalulaippaan. Solu rakennettiin koekäyttöä varten kierteittämään ohutlevykappaleisiin levytyökeskuksessa lävistettyjä reikiä. Kuvassa 7. näkyy kyseinen robottisolu, sekä vanerista rakennettuun sovitteeseen asetettu L-kirjaimen muotoinen ohutlevyaihio. Kierteittäminen levytyökeskuksella pidentää selvästi sen tahtiaikaa, joten kierteitystyö on nähty järkeväksi toteuttaa muulla tavalla. Aikaisemmin kierteityskonetta on käytetty manuaalisesti työntekijän toimesta.

Solua varten oli suunniteltu käyttöliittymä Indusoft Web Studiolla, ja toimeksiantaja halusi saada kappalevarianttien CAD-malleista kierteitettävien reikien XY-koordinaatit, ja siirtää ne käyttöliittymään mahdollisimman vaivattomasti.



Kuva 7. Robottisolu.

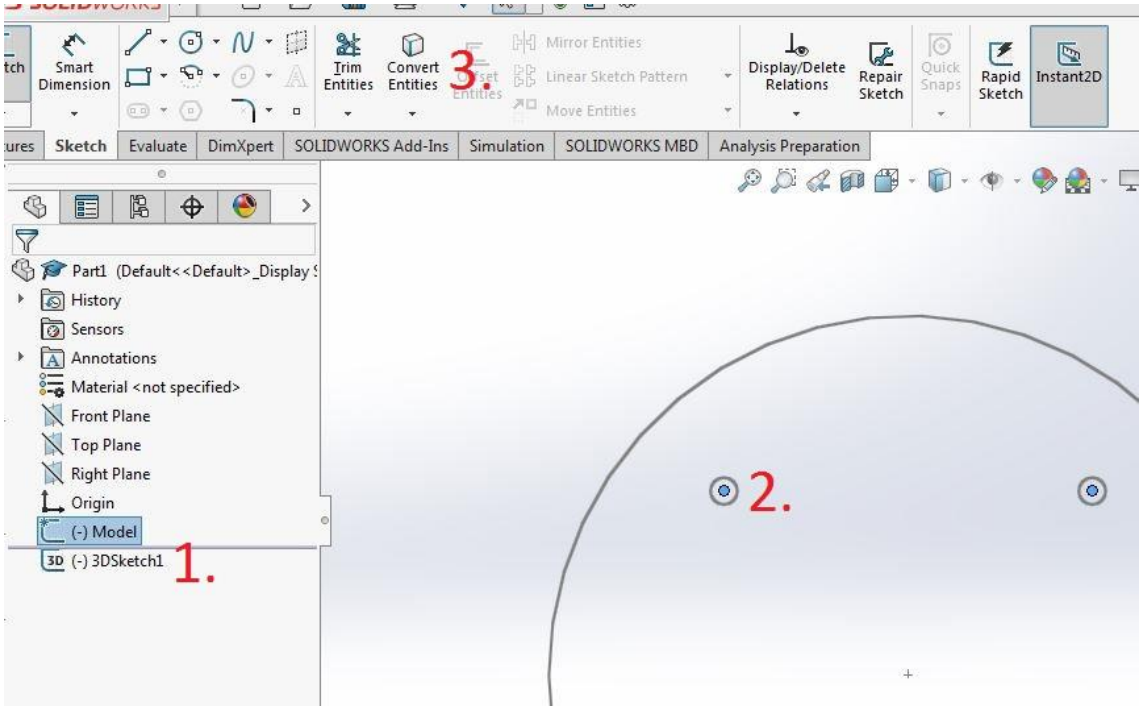
5.2 Indusoft Web Studio

Indusoft Web Studio on HMI/SCADA ohjelmisto PC-tietokoneisiin, teollisuustietokoneisiin, sulautettuihin järjestelmiin ja mobiililaitteisiin. HMI tulee sanoista *human-machine interface* ja se tarkoittaa rajapintaa ihmisen ja koneen välillä. Sen avulla operaattori koordinoi ja kontrolloi teollisia ja tuotannollisia prosesseja tuotantolaitoksessa tai koneessa. Se on ikään kuin kojelauta, joka oikein toteutettuna muuttaa monimutkaisten prosessimuuttujien tarjoaman tiedon helposti ymmärrettävään ja hallittavaan muotoon. SCADA taas tulee sanoista *Supervisory Control And Data Acquisition*, ja suomenkielellä puhutaan yleisesti valvomo-ohjelmistoista. Tämä automaatiojärjestelmän käyttöliittymä on nimensä mukaisesti valvova ylimmän tason järjestelmä, joka voi pitää allansa useamman pienemmän koneen tai laitteen HMI:n (Schneider 2016).

Indusoft Web Studiolla voi luoda HMI:n, jossa on animoituja symboleja, trendikäyriä, raportteja, hälytyksiä ja muita käyttöliittymään kuuluvia elementtejä. Ohjelmointikielenä käytetään joko ohjelman natiivia kieltä tai VBScriptiä (Visual Basic Script). Se on yksi Windowsin ohjelmointikielistä ja nimensä mukaisesti sukua Visual Basicille. Indusoft on laitteistoriippumaton, eli se toimii monien eri valmistajien laitteilla. Nykyisin Indusoft on osa monikansallista Schneider Electric-yhtiötä, jonka toimialat ulottuvat automaatiosta energiahuoltoon, ohjelmistoihin ja laitevalmistukseen.

5.3 Koordinaatit tekstitiedostoon

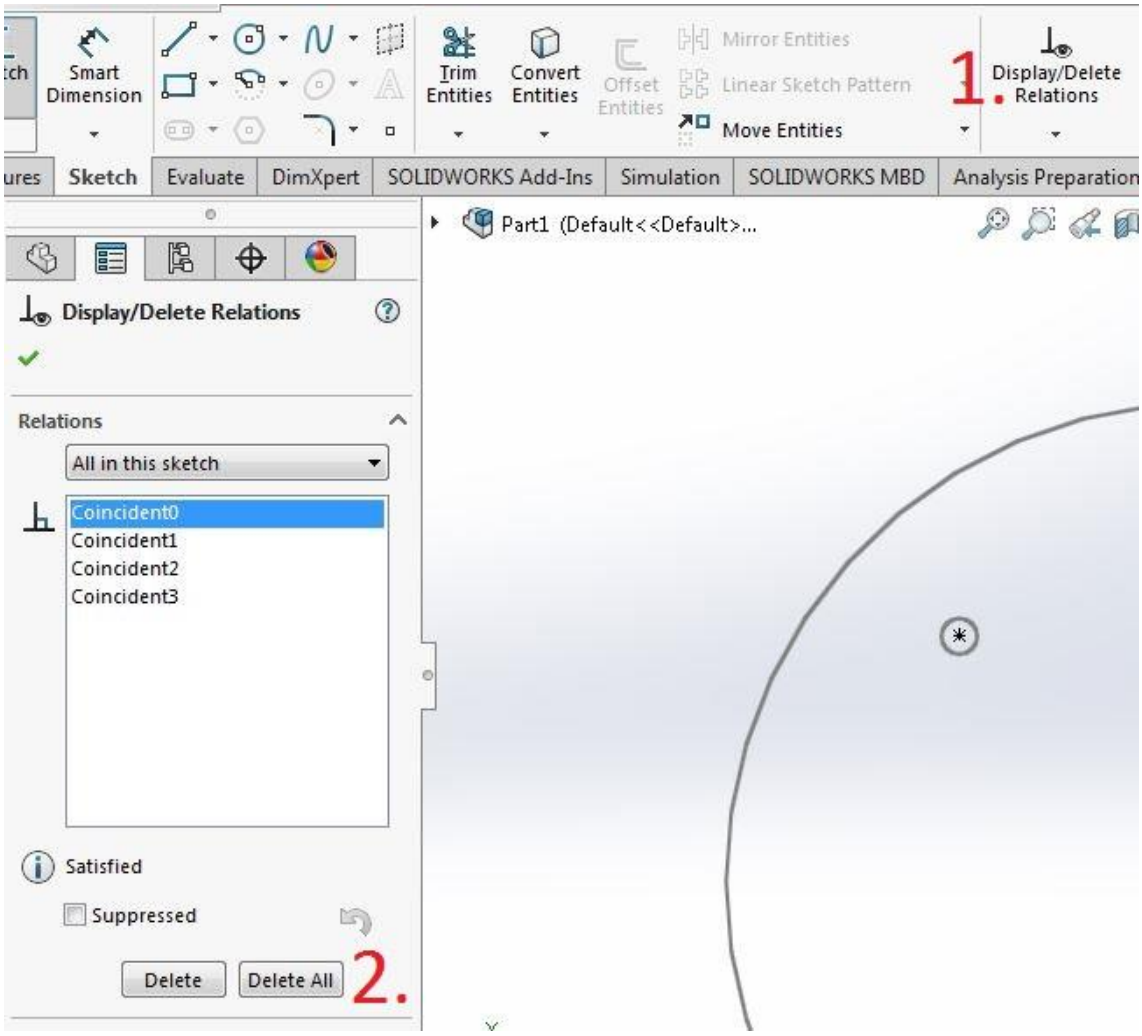
Kierteitettävien reikien koordinaattien vieminen käyttöliittymään aloitetaan avaamalla DXF-muotoinen CAD-tiedosto SolidWorksissä 2D-sketsinä. Seuraavaksi luodaan uusi 3D-sketsi ja valitaan kierteitettävät reiät. Valintatyö on järkevää toteuttaa suunnittelijan toimesta, koska sillä saadaan valittua mallinnetuista rei'istä ne, joihin kierteitystoimenpide halutaan kohdistaa. Reikien valintajärjestys määrää myös taulukointijärjestyksen, ja myöhemmin käyttöliittymään siirrettynä robotin suorittamisjärjestyksen. Näin ollen suunnittelijan kannattaa valita reiät järjestyksessä, jossa ei synny turhia siirtymiä robotille. Valitut reiät sisällytetään 3D-sketsiin komennolla *Convert Entities*, kuten on esitetty kuvassa 8.



Kuva 8. Reikien valinta.

Kaikki 3D-sketsiin liittyvät relaatiot on poistettava ja se tapahtuu komennolla *Display/Delete Relations,* jonka avaamasta ikkunasta valitaan *Delete All* kuvan 9. mukaisesti. Relaatiot pitää poistaa, jotta seuraavassa vaiheessa suoritettava makro ei kirjaisi myös niiden koordinaatteja luomaansa taulukkoon.

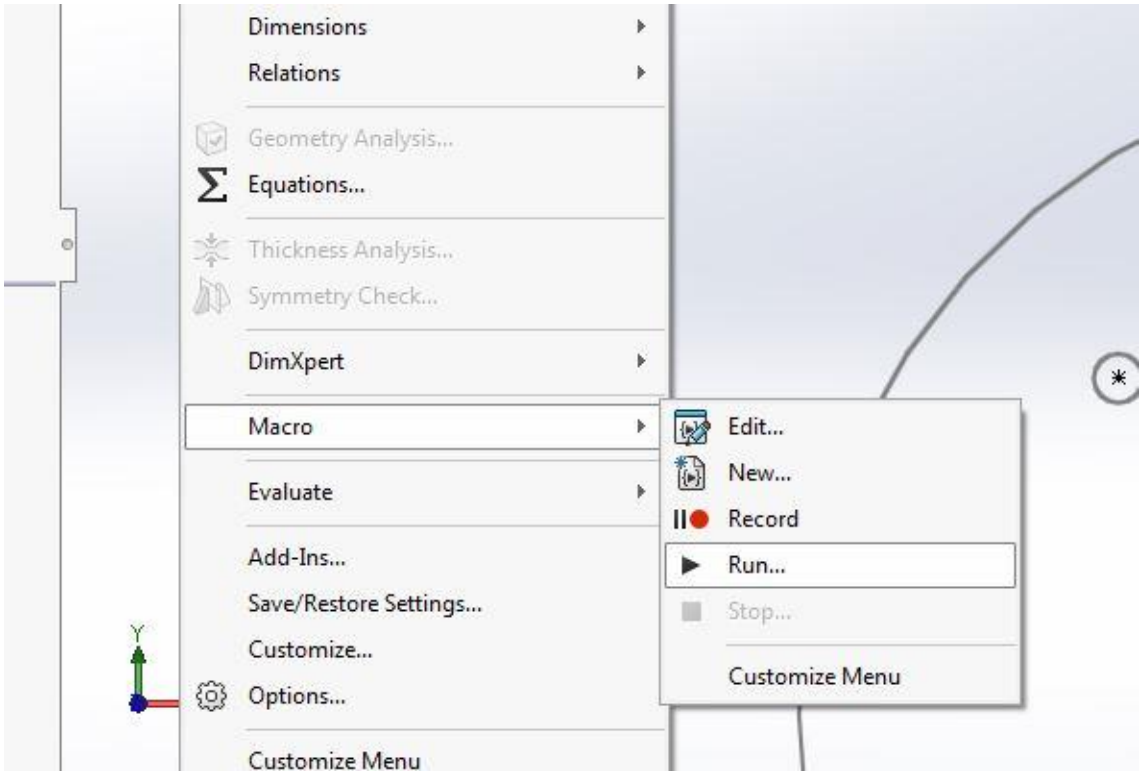
Taulukoitujen arvojen koordinaatiston origo on siinä, mihin se kappaletta SolidWorksillä piirrettäessä on asetettu. Mikäli halutaan origoksi määrittää esimerkiksi ensimmäinen esimerkin piirustuksen ensimmäinen työstettävä reikä, se on mahdollista piirustusta siirtämällä. SolidWorksissä se tapahtuu valitsemalla piirustus kokonaisuudessaan ja painamalla shift-näppäin pohjaan. Näin piirustus on raahattavissa haluttuun paikkaan origon suhteen.



Kuva 9. Kaikki relaatiot poistetaan.

Lopuksi suoritetaan *Tools/Macro*-valikosta löytyvällä komennolla *Run* (kuva 10.) makro, joka kerää valittujen pisteiden koordinaatit exel-tiedostoon. 3D-sketsin pitää olla valittuna makroa ajettaessa.

Exel-tiedosto tallennetaan sarkainerotettuna tekstitiedostona (.txt) haluttuun kohteeseen, esimerkiksi muistitikulle, ja on näin siirrettävissä edelleen käyttöliittymään. Hyvä vaihtoehto olisi tallentaa tiedosto verkkokansioon, jonne myös käyttöjärjestelmää suorittavalla tietokoneella olisi yhteys. Näin ollen eri kappalevarianttien tiedot olisivat helposti ja pysyvästi kierteityssolua käyttävän operaattorin saatavilla.



Kuva 10. Makron suorittaminen.

Suoritettavan makron VBA-koodi on kuvassa 11. ja siinä on käytössä seuraavia komentoja ja rakenteita (Microsoft 2017.):

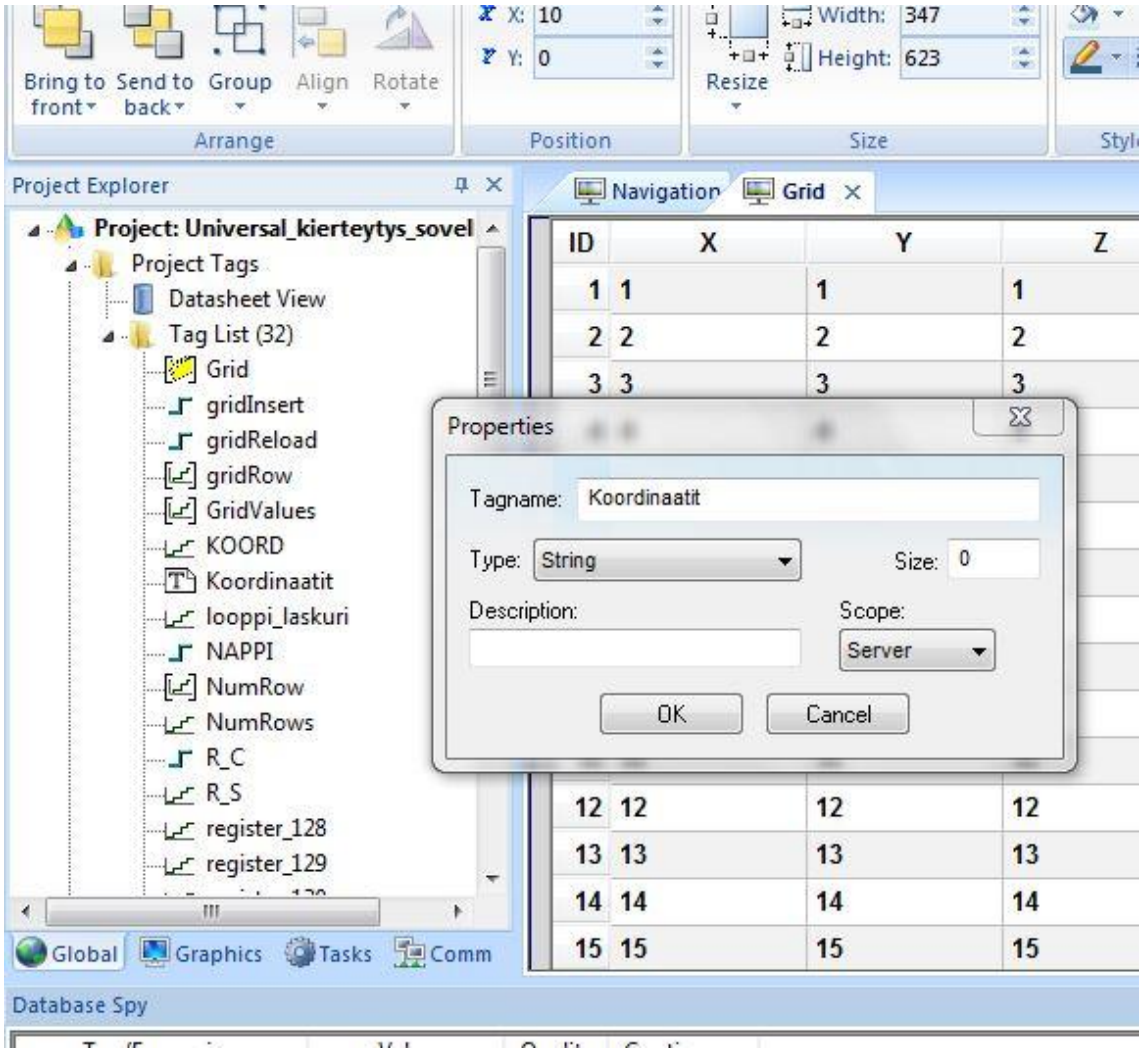
- Ohjelmakoodi kirjoitetaan komentojen *Sub* ja *End Sub* väliin
- Komento *Dim* määrittelee muuttujan ja allokoi sille tarvittavan tilan muistista.
- *Set* komennolla luodaan uusi objekti.
- *If* ehtorakennetta on käytetty viemään ohjelmaa eteenpäin määriteltyjen vaatimusten täyttyessä.
- *Nothing* arvo on objektin tila, jossa sitä ei ole alustettu, eikä se viittaa mihinkään.
- *For* silmukalla listataan koordinaattipisteet alimmasta arvosta (LBound) ylimpään (UBound).

```
Sub main()
Dim swApp As SldWorks.SldWorks
Dim doc As SldWorks.ModelDoc2
Dim part As SldWorks.PartDoc
Dim sm As SldWorks.SelectionMgr
Dim feat As SldWorks.Feature
Dim sketch As SldWorks.sketch
Dim v As Variant
Dim i As Long
Dim sseg As SldWorks.SketchSegment
Dim sline As SldWorks.SketchLine
Dim sp As SldWorks.SketchPoint
Dim ep As SldWorks.SketchPoint
Dim s As String

Dim exApp As Excel.Application
Dim sheet As Excel.Worksheet

 Set exApp = New Excel.Application
 If Not exApp Is Nothing Then
  exApp.Visible = True
  If Not exApp Is Nothing Then
   exApp.Workbooks.Add
   Set sheet = exApp.ActiveSheet
  End If
 End If
 Set swApp = GetObject(, "sldworks.application")
 If Not swApp Is Nothing Then
  Set doc = swApp.ActiveDoc
  If Not doc Is Nothing Then
   If doc.GetType = swDocPART Then
    Set part = doc
    Set sm = doc.SelectionManager
    If Not part Is Nothing And Not sm Is Nothing Then
     If sm.GetSelectedObjectType2(1) = swSelSKETCHES Then
      Set feat = sm.GetSelectedObject4(1)
      Set sketch = feat.GetSpecificFeature
      If Not sketch Is Nothing Then
       v = sketch.GetSketchPoints
       For i = LBound(v) To UBound(v)
        Set sp = v(i)
        If Not sp Is Nothing And Not sheet Is Nothing And Not exApp Is Nothing Then
         sheet.Cells(1 + i, 1).Value = Round(sp.X * 1000 / 1, 3)
         sheet.Cells(1 + i, 2).Value = Round(sp.Y * 1000 / 1, 3)
         exApp.Columns.AutoFit
        End If
       Next i
      End If
     End If
    End If
   End If
  End If
 End If
End Sub
```

Kuva 11. SolidWorks-ohjelmassa suoritettava makro.

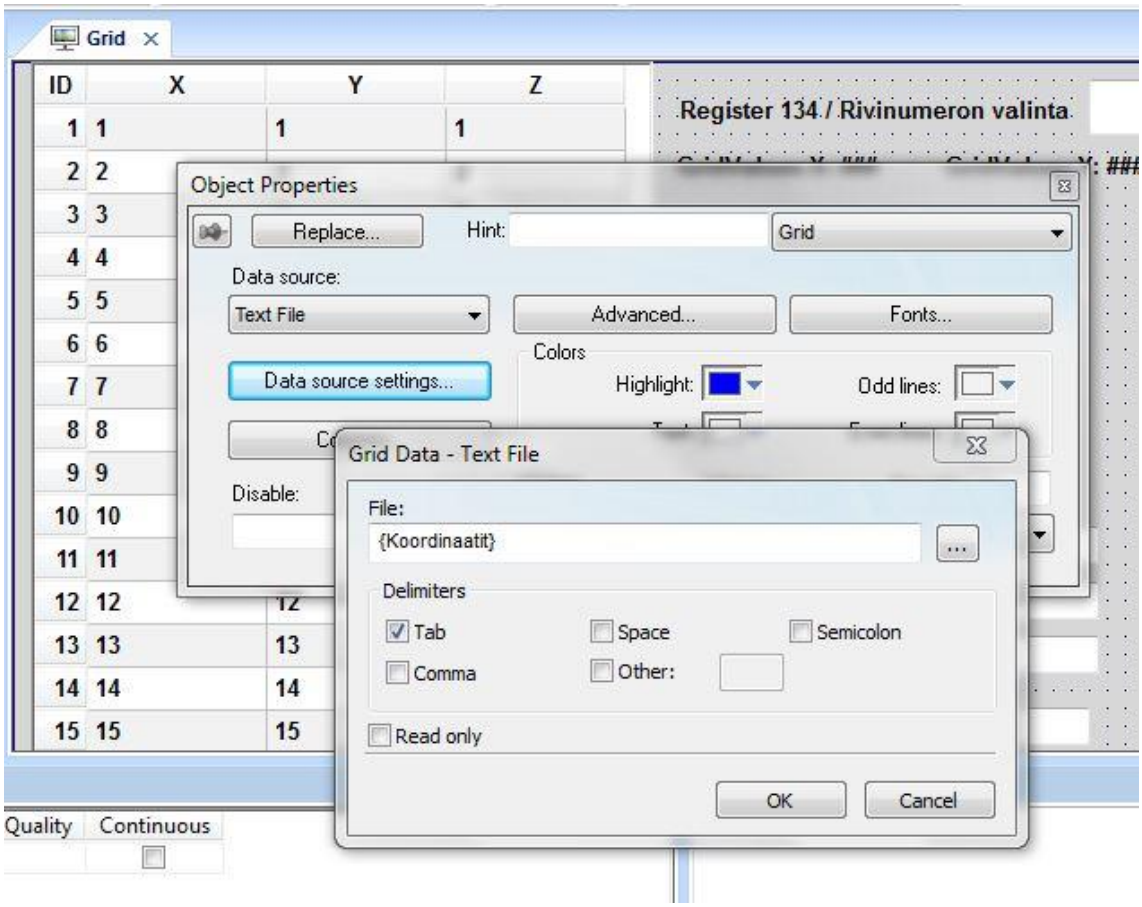
5.4 Tekstitiedosto käyttöliittymään

Seuraavaksi tekstitiedostoon tallennetut koordinaatit tuotiin InduSoft Web Studiolla luotuun käyttöliittymään. Aluksi luotiin uusi *String*-tyyppinen tagi nimeltään Koordinaatit. Tageja luodaan valitsemalla kuvassa 12. näkyvän *Project*-valikkopuun kohdan *Tag List* hiiren oikealla näppäimellä.



Kuva 12. Tagin luominen.

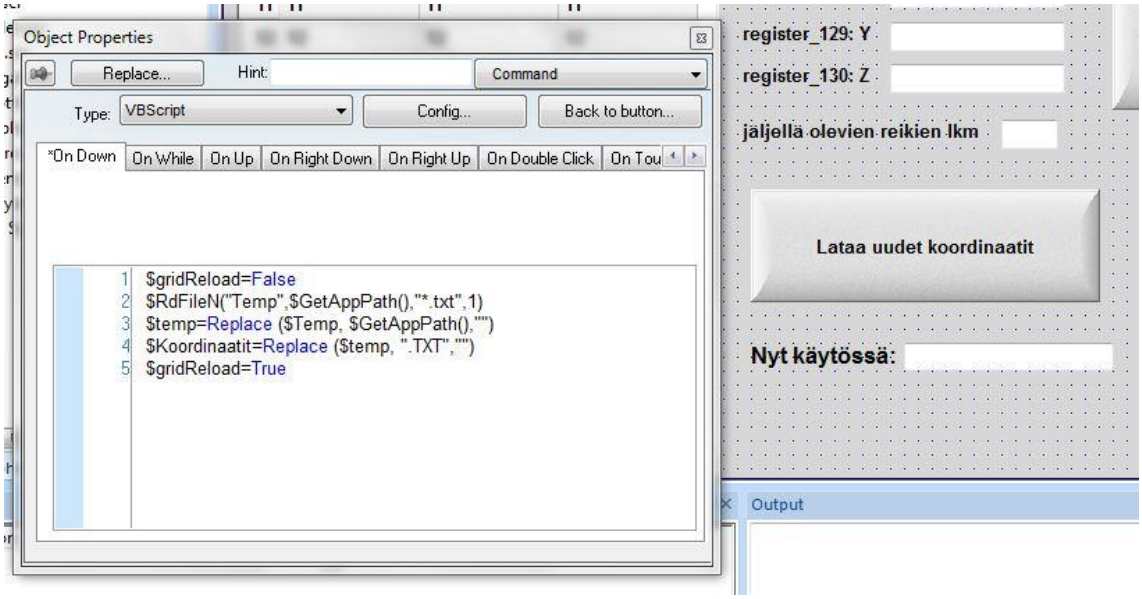
Aiemmin käyttöliittymän paikkatiedot oli syötetty taulukkoon käsin näppäimistöltä. Nyt taulukon, eli Grid-objektin datan lähteeksi määriteltiin juuri luotu tagi *Koordinaatit*, kuten kuvassa 13. Datan lähteenä pitää olla *Text File* ja erottimena sarkain eli tabulaattori.



Kuva 13. Grid-objektin asetukset.

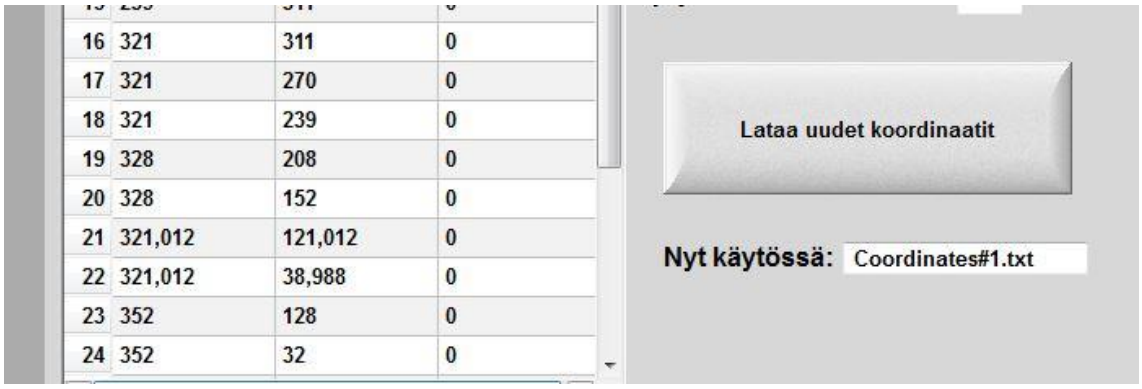
Käyttöliittymään luotiin painike, jolla uudet koordinaatit voidaan ladata hakemistosta grid-objektin käyttöön. Painikkeen ominaisuudeksi määritettiin alas painettaessa käynnistyvä lyhyt VBScriptillä kirjoitettu ohjelma, ja se on esitetty kuvassa 14. Komennolla *RdFileN* annetaan operaattorille mahdollisuus valita haluamansa tiedosto mistä tahansa kansiosta. Lisämääritteillä voidaan asettaa tiedostopolku, suodattaa haluttu tiedostotyyppi ja estää pääsy valitun kansion ulkopuolelle. Taulukko saadaan päivittymään tagin *grid-Reload* tilaa muuttamalla (Indusoft 2012).

Käyttöliittymään lisättiin *Tex Box*-objekti, jonka datan lähteeksi määritettiin myös tagi *Koordinaatit*. Näin saatiin käytössä olevan tiedoston nimi näkymään käyttöliittymään.



Kuva 14. Painikkeen ohjelma.

Taulukkoon ladatut koordinaatit ovat nyt vietävissä eteenpäin. UR10-robotin käyttöjärjestelmän tukeman TCP/IP-protokollan avulla paikoituspisteen tiedon voi siirtää robottiohjaimen muistipaikkaan, ja käyttää sieltä ohjelmassa tarvittavalla tavalla. Lopputulos näkyy kuvassa 15.



Kuva 15. Ladatut koordinaatit.

6 LOPUKSI

Tässä opinnäytetyössä pyrittiin luomaan katsaus yhteistyörobotiikkaan ja muototiedon keräämiseen CAD-mallista. Lopuksi tarkoituksena oli ratkaista toimeksiantajan ongelma kerätyn tietoperustan pohjalta ja esitellä toteutus esimerkinomaisesti.

Yhteistyörobotiikalla on vielä hyvin lyhyt historia takanaan, ja se on helppo havaita lähdekirjallisuutta etsittäessä. Painettuja teoksia on vähän ja ajantasaista lähdeaineistoa tähän työhön on etsitty paljon verkosta. Moni käytetyistä lähteistä on alalla toimiva yrityksen verkkosivusto, ja lähdekriittisyyttä pyrittiin noudattamaan tietoa vertailemalla ja intressejä pohtimalla. Asiaa helpotti hieman koulutuksen aikana kertynyt vähäinen omakohtainen kokemus yhteistyörobotiikasta.

SolidWorksin API:n laajuus yllätti, ja mahdollisuuksia sen hyödyntämiseen on varmasti valtavasti. Se vaatii kuitenkin merkittävää perehtymistä aiheeseen, ja esimerkiksi VBA- tai VB.NET-kielen edes jonkinasteista hallintaa. Tämä työ, vaikkakin vain pintaraapaisuna aiheeseen, herätti kuitenkin mielenkiinnon oppia asiasta lisää.

On täysin sovelluskohtaista, kannattako tietoa kerätä suoraan CAD-mallista, vaiko siirtyä käyttämään OLP-ohjelmistoa. Geneeristen, eli useamman eri robottimerkin kanssa yhteensopivien OPL-ohjelmistojen hintataso tulee varmasti tulevaisuudessa laskemaan. Esimerkkinä voi mainita jo nyt suhteellisen edullisen RoboDK-ohjelmiston, jonka API mahdollistaa lukuisten eri valmistajien robottien ohjelmoinnin ja simuloinnin python-ohjelmointikielellä. Myös STEP- ja IGES-formaattia olevien CAD-mallien muototiedon hyödyntäminen on mahdollista esimerkiksi jyrsintää varten luotavan radan paikoituspisteitä tallennettaessa.

Tämän opinnäytetyön esimerkkinä toteutettu tiedonkeräystapa on varsin yksinkertainen ja soveltuu vain hyvin saman kaltaisiin sovelluksiin. Sen avulla vältytään kuitenkin OLP-ohjelmiston hankinnalta ja opettelulta, ja jonkinlainen käyttöliittymä olisi mahdollista rakentaa myös Windows Excel-ohjelman avulla. Näin säästyttäisiin myös Indusoft Web Studion hankinnalta. Monimutkaisemmat sovellukset vaativat kuitenkin selvästi parempia ohjelmointitaitoja, joten jokaisen projektin mielekkyys ja lähestymistapa on valittava käytettävien resurssien perusteella.

LÄHTEET

ABB 2017. Viitattu 30.6.2017 http://abbcloud.blob.core.windows.net/public/images/aae08d95-db54-4d3f-8f68-204f903910b5/presentation.jpg

Bélanger-Barrette, M. 2016 Collaborative Robot Risk Assessment, An Introduction. Viitattu 15.6.2017 http://blog.robotiq.com/safety-collaborative-robots-risk-assessment

Dassault Systèmes 2017. Viitattu 9.8.2017 http://help.solidworks.com/2014/English/api/sldworksapiprogguide/GettingStarted/Types_of_SolidWorks_API_Applications_Overview.htm?id=7f12d04436e248cfbf6eaf32ea206fc6#Pg0&ProductType=&ProductName=

Finnrobotics Oy 2017. Offline ohjelmointi. Viitattu 30.6.2017 http://finnrobotics.fi/suunnittelu/offline-ohjelmointi/

Foit, K.; Ćwikła, G. 2017. The CAD drawing as a source of data for robot programming purposes. Viitattu 8.8.2017 https://www.matec-conferences.org/articles/matecconf/pdf/2017/08/matecconf_cosme2017_05002.pdf

Gomez, J. 2017. VBA Methods: The Complete Guide To Working With Methods In Excel. Viitattu 14.8.2017 https://powerspreadsheets.com/object-methods-in-vba/

Hawk Ridge Systems 2012. SolidWorks API Building Blocks – Part 3. Viitattu 18.8.2017 https://www.hawkridgesys.com/blog/solidworks-api-building-blocks-part-3/

Hietikko, M.; Malm, T.; Alanen, J. 2009. Koneiden ohjausjärjestelmien toiminnallinen turvallisuus. Helsinki: VTT

Indusoft, Inc. 2012. Viitattu 13.7.2017 http://www.indusoft.com/blog/2013/05/23/using-the-rdfilen-function-in-indusoft-web-studio-scada-software/

Kaarela, J. 2007. Robotin etäohjelmoinnin käyttöönoton vaiheet robotisoidussa ohutlevyn särmäyksessä. Opinnäytetyö. Teknologiaosaamisen johtaminen – koulutusohjelma. Keski-pohjanmaan ammattikorkeakoulu.

Koukkari, T. 2016. Collaborative robotics: Human-robot collaboration in heavy manufacturing tasks. SeAMK. Viitattu 8.6.2017 http://roboyhd.fi/wp-content/uploads/2016/12/Timo_Koukkari_HRC_SeAMK.pdf

Kuivanen, R. (toim.) 1999 Robotiikka. Helsinki: Talentum Oyj/MetalliTekniikka

Leiniö, V. 2013. Ruiskuvalumuotin suunnittelu. Opinnäytetyö. Muovitekniikka. Lahden ammattikorkeakoulu

Malm, T. (toim.) 2008. Vuorovaikutteisen robotiikan turvallisuus. Helsinki: Suomen Robotiikkayhdistys ry.

Microsoft 2017. Visual Basic Reference. Viitattu 21.7.2017 https://msdn.microsoft.com/en-us/library/sh9ywfdk(v=vs.90).aspx

MLC CAD Systems 2014. Writing code in the VB editor. Viitattu 11.8.2017. https://vimeo.com/99799025

Neto, P.; Mendes, N. 2013. Direct off-line robot programming via a common CAD package. Viitattu 8.8.2017 https://estudogeral.sib.uc.pt/bitstream/10316/27383/1/Direct%20off-line%20robot%20programming%20via%20a%20common%20CAD%20package.pdf

Pan, Z.; Polden, J.; Larkin, N.; Van Duin, S.; Norrish, J. 2012. Recent progress on programming methods for industrial robots. Viitattu 19.7.2017 http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1034&context=eispapers

Pekkanen, E. 2010. Uuden robottisolun ohjelmointitavan mahdollisuudet pienille ja keskisuurille yrityksille. Diplomityö. Konetekniikan koulutusohjelma. Lappeenranta: Lappeenrannan teknillinen yliopisto.

Pittman, K. 2016. A History of Collaborative Robots: From Intelligent Lift Assists to Cobots. Viitattu 7.6.2017 http://www.engineering.com/AdvancedManufacturing/ArticleID/13540/A-History-of-Collaborative-Robots-From-Intelligent-Lift-Assists-to-Cobots.aspx

Rice, K. 2014. Macros vs Add-ins vs Stand-Alones. Viitattu 9.8.2017 http://www.cadsharp.com/blog/macro-vs-add-in-vs-stand-alone/

Roberts, A. 2006. The History of Science Fiction. New York, NY: PALGRAVE MACMILLAN.

Robotiq 2016. ISO/TS 15066 Explained. Viitattu 12.6.2017 http://robotiq.com/wp-content/uploads/2016/05/ebook-ISOTS15066-Explained.pdf

Roos, D. 2017. What is an API? Viitattu 9.8.2017 http://money.howstuffworks.com/business-communications/how-to-leverage-an-api-for-conferencing1.htm

Schneider Electric Software, LLC. 2016. Viitattu 7.7.2017 https://www.wonderware.com/hmi-scada/what-is-hmi/

SFS-EN ISO 10218-1. 2011. Robotit ja robotiikkalaitteet. Turvallisuusvaatimukset. Osa 1: Teollisuusrobotit. Helsinki: Suomen standardoimisliitto SFS.

SFS-EN ISO 10218-2. 2011. Robots and robotic devices. Safety requirements for industrial robots. Part 2: Robot systems and integration. Helsinki: Suomen standardoimisliitto SFS

Suomen Standardoimisliitto ry. 2017. Mitä standardisointi on? Viitattu 9.6.2017 https://www.sfs.fi/standardien_laadinta/mita_standardisointi_on

Tikka, A. Parametrinen mallinnus. Viitattu 17.7.2017. http://tekninen3dmallinnus.blogspot.fi/p/parametrinen-mallinnus-on-perinteinen.html

TM Robotics 2017. Cobots and Industrial Robots: Choose the Right Robot for the Job. Viitattu 9.6.2017 http://www.tmrobotics.co.uk/wp-content/uploads/2017/04/Collaborative-vs.-Industrial-robot-wp_FINAL_3-16-17.pdf

Universal Robots 2015. 17.7.2017 https://www.universal-robots.com/media/8764/ur10_user_manual_en_global.pdf

Universal Robots 2017. Viitattu 8.6.2017 ja 17.7.2017 https://www.universal-robots.com/

Zacobria Universal-Robots community 2017. Viitattu 29.6.2017. http://www.zacobria.com/universal-robots-zacobria-forum-hints-tips-how-to/gui-programming-universal-robots/