KONFIGUROINNIN INTEGRAATIO D365 PILVIYMPÄRISTÖSSÄ

LAB-AMMATTIKORKEAKOULU
Insinööri (ylempi AMK)
Digitaaliset teknologiat
Syksy 2020
Janne Loikkanen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Loikkanen, Janne	Opinnäytetyö, YAMK	Syksy 2020
	Sivumäärä 65	

Työn nimi

Konfiguroinnin integraatio D365 pilviympäristössä

Tutkinto

Insinööri (ylempi AMK), Digitaaliset teknologiat

Tiivistelmä

Digitalisaatio on edennyt automatisoiduista paikallisista järjestelmistä pilvipalveluihin, jolloin nykyisten paikallisten järjestelmien ja eri pilvestä tulevien järjestelmien välille pitää saada toimiva integraatio, jotta järjestelmät pystyvät kommunikoimaan ERP -prosessin eri vaiheissa.

Opinnäytetyössä tutkitaan yritys-x:n nykyisen offline -pohjaisen HOT -tarjouskonfiguraattorin ja Dynamics AX2009 -version integroinnin siirtämistä D365 pilvipalveluna tulevaan ERP-ratkaisuun, johtuen Dynamics AX2009 -version tuen päättymisestä. Lisähaasteena oli AX2009 ja D365 versioiden konfiguroinnin tuotemallien ja valintaehtojen tekemisen erilaisuus.

D365 on Microsoft:in tuote, jonka vuoksi Azure oli pilvialustana helpoin ratkaisu. Perus HOT -konfiguraattorin server -ohjelmisto siirrettiin Azure pilveen, johon HOT -offline clientit ottava yhteyttä saadakseen ajantasaisen perusdatan ja voidakseen lähettää uudet tarjouksen tekohetkellä konfiguroidut tarjoukset D365 ERP -järjestelmää tarjouksiksi. Lisäksi D365:n puolelle siirretyt tarjoukset täytyy saada konfiguroitua automaattisesti valmiiksi ilman henkilöstön käsittelyä, koska konfigurointi erätyönä ulkoisilla parametreillä ei ole standardin D365:n ominaisuus.

Pilvessä olevat D365 ja HOT -server saadaan helposti keskustelemaan keskenään OData –rajapinnan kautta, joten se ei ole ongelma, vaan haasteeksi nousee koko prosessi ja sen suorituskyky.

Asiasanat

ERP, D365, HOT Client, HMPCE, JSON, Konfigurointi, Integraatio

Abstract

Author(s)	Type of publication	Published
Loikkanen, Janne	Master's thesis	Autumn 2020
	Number of pages	
	65	

Title of publication

Configurations integratio D365 in cloud environment

Name of Degree

Master of Engineering, Digital Technology

Abstract

Digitization has progressed from automated local systems to cloud services, which now operate with local systems and systems coming from different clouds, in order to act as integrations to enable systems to communicate at different stages of the ERP process.

The thesis investigates company-x's current offline-based HOT quotation configuration tool and Dynamics AX2009 version to move integration to the cloud service due to the inference of Dynamics AX2009 version support for the new D365 cloud service future ERP solution. An additional challenge was the difference in product modeling and selection criteria for configuring the AX2009 and D365 versions.

The D365 is a product of Microsoft, which is why Azure was the easiest solution as a cloud platform. The basic HOT configurator server software is transferred to the Azure cloud, where the connection made by the HOT offline client is sent to the current basic data and the new provider to be won over with the D365 ERP systems configured at the time of creation as offers. In addition, bids transferred to D365 side must be automatically configured without personnel processing, as batch configuration with external parameters is not a feature of the standard D365.

The D365 in the cloud and the HOT server can be made to interact with each other easily, but the challenge lies in the whole process and its performance ability.

Keywords

ERP, D365, HOT Client, HMPCE, JSON, Configuration, Integration

SISÄLLYS

1 JOHDANTO 1
1.1 Opinnäytetyön tavoitteet 1
1.2 Tutkimuskysymykset ja tutkimusmenetelmät 2
1.3 Työn rakenne 6
2 PERUSTEET 7
2.1 Mitä ERP järjestelmä tarkoittaa? 7
2.2 Mitä tuotteen konfigurointi ja modulaarisuus merkitsee? 9
2.3 WebService 13
2.4 JSON 13
2.5 Azure 13
2.6 D365 Business Event 14
2.7 XML 14
3 INTEGROITAVIEN JÄRJESTELMIEN TEKNISET YMPÄRISTÖT 15
3.1 Microsoft Dynamics ERP 15
3.1.1 Dynamics versiot 16
3.2 Dynamics AX2009 18
3.3 Dynamics D365 18
3.4 HOT -Client, HMPCE ja HOTApi 18
4 AX2009 JA D365 VERSIOIDEN KONFIGUROINNIN EROAVAISUUDET 21
4.1 AX 2009 konfigurointi 22
4.1.1 Konfiguroinnin valintaikkuna 22
4.1.2 Konfiguroinnin perusasiat ja muuttujat 23
4.1.3 Ehdot tehdään koodisolmuilla 23
4.2 D365 konfigurointi 26
4.2.1 Konfiguroinnin valintaikkuna 26
4.2.2 Konfiguroinnin mallin perusasiat ja muuttujat 27
4.2.3 Ehdot tehdään Regular Expression pohjaisesti 28
4.2.4 Jälkikonfigurointi ohjelmoimalla PCAdaptor –luokkia 31
5 LOPPUTULOS JA MITEN SIIHEN PÄÄSTIIN 32
5.1 PCAdaptor –luokka 33
5.2 Universal PCAdaptor –luokka 33
5.2.1 Universal PCAdaptor luokan käynnistys ja alustus 34
5.2.2 Etsitään lähetettävät parametrit eli ns. Triggerit 35

5.2.3 Lähetetään parametrit HMPCE –laskenta moottorille....36
5.2.4 Odotetaan ja luetaan vastaussanoma....38
5.2.5 Sanoman tallennus tietokantaan....39
5.2.6 Tuotemallin päivitys....40
5.3 Tuotemallin testaus....48
5.3.1 BOM JSON....52
5.3.2 Esitallennettu konfiguroinnin XML....54
5.4 Automaattinen konfigurointi....56
5.4.1 Automaattisen konfiguroinnin toiminta....57
5.4.2 Automaattisen konfigurointi –prosessin kiihdytys....59
5.4.3 Common Business Event perusta tilanteen jakamiseen....60
5.4.4 Tarjouksen konfiguroinnin tilanteen jakaminen....60
6 YHTEENVETO JA JATKOKEHITYS....62
LÄHTEET....64

1 JOHDANTO

Tämän opinnäytetyön tarkoitus on tutkia digitaalisen toiminnanohjausjärjestelmän version uudistamisesta aiheutuvia muutoksia, sekä tuotemallien konfiguroinnissa, että sen integrointia olemassa olevaan liiketoimintasovellukseen pilviympäristössä. Lähtökohta työlle on integraatio HOT –työkaluun, joka on myyntihenkilöiden tuotteiden konfigurointi ja tarjoushinnoittelutyökalu, sekä Microsoft Dynamics AX 2009 toiminnanohjausjärjestelmien nykyinen integraatio. Versiovaihdon tärkeimpänä syynä on sovellustuen päättyminen MS Dynamics AX 2009 -järjestelmälle. Uutena kohdejärjestelmänä toimii Microsoft Dynamics 365 FO.

Työskentelen IT-alan konsulttiyrityksessä, joka tarjoaa muun muassa asiakkailleen IT-infrastruktuuriratkaisuja, sekä tietojärjestelmien integraatiopalveluja. Yritys-x puolestaan toimii teollisuudessa, jossa asiakkaille toimitetaan konfiguroituja kokonaisuuksia. Yritys-x käyttää konsulttitalon, jossa työskentelen, toimittamaa toiminnanohjausjärjestelmää. Koska yritys-x on nykyinen konsulttiyrityksen asiakas, opinnäytetyön tutkimustulokset palvelevat sekä yritystä, jossa työskentelen että yritys-x:ää, jonka järjestelmiin versiovaihto tehdään.

Opinnäytetyöstäni tuli Design Science tyyppinen suunnittelutieteellinen, tekniikoita testaava ja ratkaisukeskeinen asiakkaani tarpeita toteuttava tutkielma.

1.1 Opinnäytetyön tavoitteet

Opinnäytetyön tavoite oli varmistaa, että yritys-x:n nykyisten tuotteiden tuotemallit ovat mallinnettavissa uudessa Microsoft Dynamics D365 standardi (tästä eteenpäin D365) version tarjoamilla menetelmillä, jottei ulkopuolista tuotetta tähän tarvita. Ulkopuoliset konfigurointijärjestelmät ovat usein huomattavan kalliita ja kaikissa ratkaisuissa on omanlaisensa mallinnusmenetelmät ja mahdolliset kielet, joilla mallinnusta tehdään. Näistä prosessin muutoksista, sekä kielen että tavasta tehdä muutoksia, aiheutuisi yritys-x:lle oman väen koulutustarve ja oppimisen etenemisen varmistaminen, jotka toisivat lisähaasteita tuotannon käyttöönotossa. Nykyinen mallinnustapa tunnetaan perusteellisesti, jolloin samankaltaisella tavalla käsiteltävällä mallilla saavutetaan pienempi oppimiskynnys. Lisäksi haluttiin saada tuotemallien laskentojen datat ylläpidettäväksi yhteen järjestelmään, eikä taas uuteen kolmanteen järjestelmään. Nykyjärjestelmässä laskennat ovat, sekä AX2009 että HOT:in puolella, joissa tarjousten hintaeroissa tapahtuu heittoja aika ajoin. Tämä johtuu siitä, että tuotemallin versiota ja hinnoittelua ei ole aina ehditty päivittämään samalaiseksi molempiin järjestelmiin.

Kolmantena tavoitteena on varmistaa toteutettavien tarjousten sisään tuonti ja jatkaa siitä automaattiseen konfigurointiin HOTCoden pohjalta. HOTCode on merkkijono, jossa on kaikki konfiguroinnin parametrit ja arvot. Automaattinen konfigurointi ei ole D365 standardissa mahdollista ilman käyttäjän tekemään työtä konfigurointivalintojen valinta -näytössä. Halutaan siis, että tarjoukset ovat valmiina, koska ne on jo tehty HOTin puolella valmiiksi konfiguroituna.

Neljäs tavoite oli, kuinka kaikki mallit saadaan käyttämään ulkoista HMPCE laskentamoottoria jokaisen mallin kohdalla. Ongelmana tässä oli se, että oli tiedossa että, D365:n PCA-daptor –luokkia oli mahdollista ohjelmoida, mutta jotta kaikille noin 300 mallille ei tarvitsisi tehdä omaa uudella nimellä ja osasta lievästi eroavia ohjelmallisia kokonaisuuksia tai vain vähän eroavia muutoksia. Ratkaisuksi haluttiin riittävän universaali ratkaisu.

Viides tutkittava ongelma olisi konfiguroinnin saaminen riittävän nopeaksi. Tämä ongelma olisi vasta tutkittavissa, kunnes koko konfigurointi saataisiin toteutettua. Tähdättiin kuitenkin automaattiseen konfigurointiin, jossa ei tarvita käyttäjän käsityötä integraatiossa. Pyrittäisiin tekemään erilaisia ohjelmallisia kiihdytys -toimintoja, jos tarpeen.

Kuudes tavoite on tuoterakenteen osien siirtäminen helposti HMPCE –Java laskentamoottorille. HMPCE on yritys-x:n omatuotantoa oleva tuotemallien konfiguroinnin laskentamoottori, jota käytetään HOT –tarjoussovelluksessa, joka on myöskin yritys-x:n omatuotantoa oleva sovellus. Tuoterakenteen osien siirtäminen HMPCE moottorille on tärkeää. Kateprosentin laskennassa on muuttujia, jolloin kaikkia tuotteita ja sen osista saatavaa katetta ei voida laskea peruskaavalla, jotta yksittäisen kalliin hankittavan toimilaitteen kustannus ei pilaisi koko tarjousta. Tämä voisi johtaa tilanteeseen, jolloin koko tarjous ei olisikaan enää kilpailukyinen. Tuoterakenne paljastaa siis kalliit yksittäiset osat ja niille voidaan antaa pienempi yksittäinen kateprosentti, jolloin laskentamoottori saadaan pienentämään tarjouksen kokonaishintaa yritys-x:n asiakkaille ja tarjous on kilpailukykyisempi.

1.2 Tutkimuskysymykset ja tutkimusmenetelmät

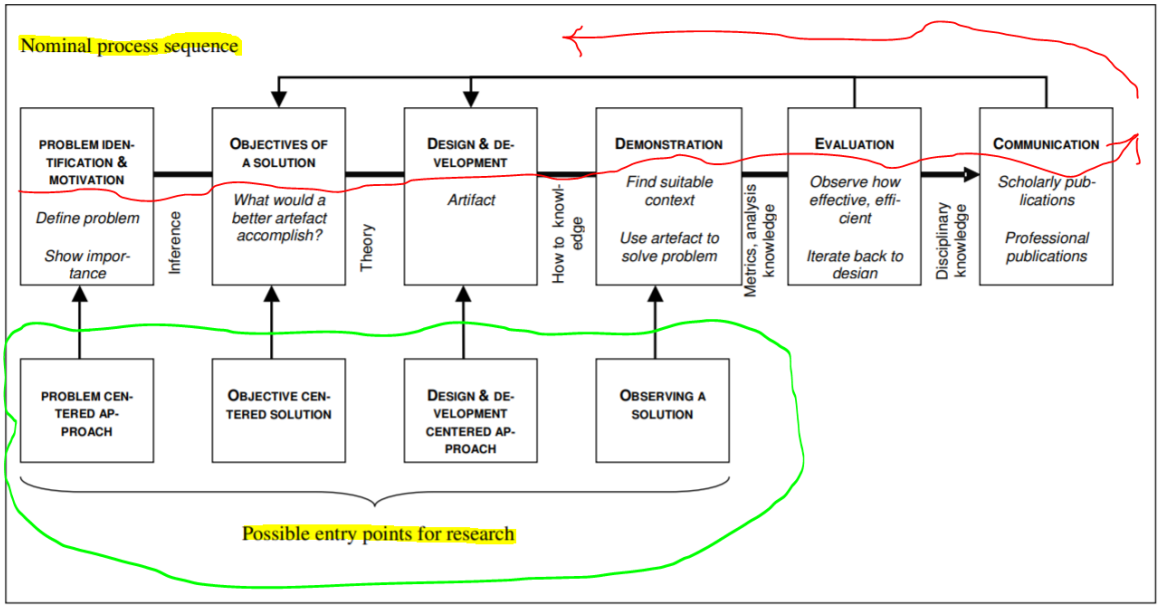
Opinnäytetyöhön liittyvät tutkimuskysymykset ovat:

1. Kuinka yritys-x:n nykyinen tuotemallinnus on toteutettavissa standardilla D365:llä?
2. Kuinka tuotemallien reittien ja tuoterakenteiden ehdollinen muuttuva data saadaan siirrettyä ylläpidettäväksi HMPCE laskentamoottorin puolelle?
3. Kuinka kaikki tuotemallit saadaan käyttämään ulkoista HMPCE laskentamoottoria laskennoissaan?

4. Kuinka tuotemallin konfigurointi saadaan toteutettua riittävän nopeaksi manuaali käytössä ja automaattisen konfiguroinnin erätyön kautta?
5. Miten automaattinen konfigurointi toteutetaan ilman käyttäjän manuaalista työtä?
6. Miten tuotemallin tuoterakenne saadaan siirrettyä D365:sta HMPCE laskentamoottorin puolelle kateprosentin selvittämiseksi?

Opinnäytetyön tutkimusmenetelmäksi tuli Design Science tyyppinen suunnittelutieteellinen lähestyminen (Peffers ym. 2006). Testattiin myös ”Konstruktiivinen tutkimusote”, jossa tehdään todella uutta eli mikä toimii tai ei toimi (Lukka 2001). Myös tämä tutkimusmenetelmä sopisi hyvin ohjelmistokehitystyöhön, koska varsinaisesti ratkaisua ei ole olemassa kuin ulkoiset 3. osapuolen järjestelmät, jotka perustuvat eri ratkaisuun eli ODATA//REST -rajapintaan, eikä suoraan ohjelmalliseen ratkaisuun vaan suoraan datan tuontiin valmiilla rakenteella ja reitityksellä. Ongelmana menetelmässä oli seuraava lause määritelmässä: ”Pelkkää aiemmin kehitettyjen konstruktioiden soveltamista uuteen ympäristöön ei tule pitää konstruktiivisen tutkimusotteen sovelluksena.” Tässä opinnäytetyössä ei kehitetty uutta ratkaisua, vaan käsiteltiin olemassa olevan AX2009 version päivittämistä D365-ympäristöön (Dynamics 365 2020c).

Design science sisältää kuusivaihetta (kuvio 1).



KUVIO 1. Design science -menetelmän vaiheet (muokattu Peffers ym. 2006)

Kuviossa 1 punaisella on esitetty prosessin suunta ja vihreällä on prosessin vaiheisiin vaikuttavat seikat. Huomataan, että prosessi palaa aiempiin vaiheisiin, eli kyseessä iterointi kierrokset, joihin päädytään, kun on tarvetta parantaa ratkaisua (artefaktia).

Prosessin kuvaus (Peffers ym. 2006):

1. Ongelman tunnistus ja motivointi

   Kuvion vihreältä puolelta tulee lähestyä ongelmakeskeisellä tavalla eli tunnistetaan tarpeet ja ongelmat. Saavutetaan motivaatio ratkaisun hakemiseen ja tuloksiin.

2. Ratkaisun ominaisuudet

   Kuvion vihreältä puolelta tulee lähestyä ratkaisukeskeisellä tavalla. Ongelman kuvaksesta pyritään ratkaisemaan saavutettavissa olevat tavoitteet. Lisäksi täytyy löytää asiat, jotka tarvitaan ratkaisun saavuttamiseksi.

3. Suunnittelu & toteutus (artefakti)

   Kuvion vihreältä puolelta tulee lähestyä suunnittelu ja kehitystyyppisellä lähestymistavalla. Etsitään ja kuvataan tarvittava toiminnallisuus eli ratkaisu, siis artefakti.

4. Testailu & protoilu (kokeellinen testailu)

   Kuvion vihreältä puolelta tulee lähestyä testaillen ja kokeilemalla ratkaisua, jossa on tarkoitus havainnollistaa artefakti eli ongelman ratkaisu ja hyödyllisyys. Lisäksi tulee ymmärtää mitä osaa demonstroidaan ratkaisusta, koska kaikkea ei voida testata yhdellä iterointi kierroksella, kuin vasta aivan lopuksi.

5. Toteutus

   Toteutuksen jälkeen seuraava iterointi kierros johtaisi takaisin suunniteluun iterointikierroksen lähes täyden kierroksen myötä. Jos lopputulos ei miellytä niin joudutaan palaamaan takaisin iterointi kierrokselle, niin on avattava ja palattava aiempiin päätöksiin, jotka johtivat kokeiluun. Seurauksena on siis iterointi kierros niin monesti kuin tarvitaan, eli paluu aiempiin kuvio 1:n osittamalla tavalla jalostamalla tarvetta ja ratkaisua.

6. Viestintä

   Viimeinen vaihe olisi julkaisu, voidaan keskustella alkuperäisestä ongelmasta ja ratkaisusta, jota tarvittiin ja sen tärkeydestä. Artefaktin eli ratkaisun yhteydestä tulee keskustella, kuinka se tarjoaisi uusia mahdollisuuksia ja ratkaisun tärkeydestä ongelman ratkaisun käytöstä. Kaikkien artefaktin määrittelyyn ja ongelman ku-

vaukseen osallisien tulisi osallistua ratkaisusta keskusteluun tutkijoiden ohella. Voidaan edelleen joutua iterointi kierrokselle keskusteluiden jälkeen, jos ratkaisu on halutun kaltainen niin ratkaisu eli artefakti on valmis.

Opinnäytetyön monimutkaiseen ja erilaisia ongelmia sisällään pitäviin tilanteisiin suunnittelutieteellinen tutkimus sopi lopulta parhaiten. Ensimmäisessä vaiheessa tunnistettiin ongelmat kaikista tavoitteista kysymysten pohjalta. Varsinaista motivointia ei tarvittu, koska kaikki esitellyt kysymykset olivat pohjiltaan lähes ns. ”Show Stoppereita” eli projekti ei olisi edennyt, ellei asioita olisi voitu ratkaista. Toisessa vaiheessa kuvattiin, mitä tulisi pystyä tekemään, jotta tutkinnan jatkaminen on järkevää. Taustalla oli työnantajan pohjalta kaupallinen projekti, jonka vaiheiden eteneminen tuli olla edistyvä, jotta jatkomahdollisuudet olisivat olemassa. Kolmannessa vaiheessa toteutettiin suunnitelmallinen ratkaisu luomalla helpoimmat ongelmat avatuiksi ratkaisuiksi ja niistä integraation prosessikaavio, jonka jälkeen suunniteltiin, miten ratkaisu eri integraation vaiheissa voidaan toteuttaa. Keskusteluissa oli osallisena asiakkaan tarpeita kuvaavat henkilöt, järjestelmän toimittavan konsulttiyrityksen integrointiin erikoistunut tiimi, D365 puolelta opinnäytetyön tekijä ja tarvittavat osa-alue konsultit. Koska päivitysympäristön integraatio oli monimutkainen, oli tarve tehdä ensimmäisiä toteutuksia, jotta eriasteisia riippuvuuksien vaikutuksia oli mahdollista havainnoida. Neljännessä vaiheessa tehtiin integraation toteutukselle hiekkalaatikko eli ns. ”sandbox” tyyppisessä testiympäristössä. Viidennessä vaiheessa oli todennettavissa, että alkuperäiset tarpeet tutkimuskysymysten pohjalta olisi saavutettu. Varsinaista kuudetta vaihetta ei ole vielä täysin olemassa tätä opinnäytetyötä päätettäessä, vaikka kolme vuotta se onkin kestänyt, koska suuret kaupalliset ERP-hankkeet ovat pitkäkestoisia projekteja ja taloudellisesti suuria ostajalle ja myyjälle.

Kuudennessa vaiheessa voitaisiin keskustella ratkaisujen merkityksestä ja kuinka asetetut artefaktit olivat onnistuneet niin toteutuksen, kohdistuksien ja tunnistuksen osalta. Keskustelut tulisi käydä asettajien, käyttäjien ja toteuttajien kesken. Tämä voisi johtaa dialogiin, josta saadaan selville, jouduttiinko tekemään kompromissiratkaisuja vai toteutuiko integraation päivitys alun vaatimusten mukaisesti. Lisähuomiona mainittakoon, että käytännön toteutuksissa raha lopulta ratkaisee, eikä kaikkea ole tarkoitettukaan ratkaistavavaksi kuten aiemmissa versiossa. Edeltävä toteutus ei ole aina se toteutus, jota lopulta haluttiin, vaikka se aluksi kuvattaisiin vaatimuksina ratkaisulle.

Kokonaisuutena tätä suunnittelutieteellistä lähestymistapaa pyrittiin käyttämään kaiken aikaa. Välillä tarpeet korjattiin nopeasti eri vaiheiden välillä, kun ymmärrys laajeni tarpeiden pohjalta. Tarvittavia muutoksia oli lopulta melko paljon eli tarpeet kasvoivat toteutuksen myötä. Jatko muutokset olivat lähinnä käyttöä ja testausta helpottavia ominaisuuksia.

1.3 Työn rakenne

Työ on jaettu kuuteen osa alueeseen, joista kappaleet yksi ja kaksi käsittelevät työn ja järjestelmien perusteita. Kappale kolme käsittelee integroitavien ympäristöjen perustavanlaatuisia eroja. Kappale 4 käsittelee syvemmin opinnäytetyössä käsiteltävien järjestelmien AX2009 ja D365 versioiden konfiguroinnin eroavaisuuksia. Kappaleessa 5 käsitellään lopullista toteutusta opinnäytetyössä tehdyistä muutoksista.

Kappaleiden sisällä käydään alaotsikoittain per alaotsikko, kaikki merkittävät tehdyt uudistukset ja muutokset lävitse melko tarkasti tarkentavien esimerkein ja koodi leikkauksin. Lisäksi on kerrottu miksi päätökseen päädyttiin muutamilta kohdin, koska ratkaisu oli loppuun asti avoin, miten se kannattaa lopulta toteuttaa. Kappaleessa 6 on yhteenveto sekä jatkokehityssuunnitelmat opinnäytetyössä, että sen kohteen projektista ja mitä jatkokehitys tarkoittaa projektissa. Lisäksi on arvio kokonaisuudesta, joka toteutettiin opinnäytetyön puitteissa.

2 PERUSTEET

2.1 Mitä ERP järjestelmä tarkoittaa?

ERP (Enterprise Resource Planning) tarkoittaa liiketoimintaprosessien hallintaohjelmistoa, jota yleisemmin kutsutaan toiminnanohjausjärjestelmäksi.

> *Toiminnanohjausjärjestelmä eli ERP (Enterprise Resource Planning) -järjestelmä on liiketoimintaprosessien hallintaohjelmisto, jonka avulla voidaan hallita yrityksen taloushallintoa, toimitusketjua, toimintoja, raportointia, valmistusta ja henkilöstöhallinnon toimintoja ja integroida ne. Useimmilla yrityksillä on käytössä jonkinlainen järjestelmä taloushallintoa ja toimintoja varten. Suurin osa ohjelmistoista auttaa kuitenkin vain päivittäisissä liiketoimintaprosesseissa eikä mahdollista liiketoiminnan kasvattamista tulevaisuudessa. (Dynamics 365. 2020e.)*

Yrityksen toiminnanohjausjärjestelmällä (ERP) hallitaan yrityksen prosesseja. Toisin sanoen ERP järjestelmällä voidaan pyörittää koko yritystä. Voidaan siis ylläpitää koko yrityksen prosesseja, eli käyttää ja ylläpitää henkilöstön tietojen hallintaan, palkkoihin, osto ja myynti tapahtumiin, kirjanpitoon, raportointiin, materiaalivirtojen, varastoinnin, varastosiirtojen ja lähetysten hallintaan, sekä kaikkiin tuotannon vaatimiin tehtäviin. Lisäksi voidaan hallita tuotteen rakennetta, työvaiheita tai vaikka työn ja materiaalin hintoja. Kaikista tapahtumista syntyy seuraaviin vaiheisiin täydentäviä tapahtumia. Lopulta kaikki tapahtumat aiheuttavat tuloja ja menoja, jotka tiliöidään tilikartan mukaan, ja tästä saadaan tilinpäätösdataa kirjanpitoon. Sen pohjalta yritys voi toimia kannattavasti ja reagoida nopeasti markkinoiden muutoksiin. Voidaan lisäksi ylläpitää asiakasrekisteriä, projektien vaiheita ja ennustaa tulevaa menekkiä projektin vaiheiden ja kausivaihteluiden trendien pohjalta. Esimerkiksi materiaalien ja työkustannusten hintojen noustessa muuttuneet realiteetit on saatava nopeasti mukaan myytävien tuotteiden ja projektien hintarakenteeseen, jottei kannattavuus romahda. Lisäksi projektimallisessa tai kausimuotoisessa myynnissä trendien löytäminen auttaa liiketoiminnankasvattamisessa (hintojen lasku ja nousu materiaaleista, kausista tai henkilöstömenoista johtuen). ERP:llä saavutetaan lähes reaaliaikainen kannattavuus, ennustettavuus ja liikevoiton seuranta, jolloin vältytään tilanteesta, että yritys on tehnyt tappiolla asiakkaille tuotteita useita kuukausia tai suotta kerännyt varastoon tuotteita, joille ei olekaan kysyntää (asiakasrekisterin merkinnät). Tämänkaltaista laajempaa näkyvyyttä liiketalouden prosessista, eikä vain päivittäistä yrityksen laskujen maksamista ja myyntisaatavien vahtimista (reskontra) voidaan kutsua todelliseksi ERP:ksi, jolla

pystytään seuraamaan reaaliajassa talouskehitystä ja ennustamaan tulevaa, joka mahdollistaa liiketoiminnan kasvun. (Dynamics 365. 2020e.)

Esimerkki AX2009 ja D365 myyntiketjun ERP toiminnasta:

1. Myyjät jättävät tarjouksen asiakkaan tarvitsemista tuotteista asiakkaan tarpeet huomioiden eli tuotteet mahdollisesti räätälöidään tarkemmin konfiguroimalla asiakkaan tarpeiden mukaan.
2. Asiakas hyväksyy tarjouksen, jolloin se vahvistetaan myyntitilaukseksi.
3. Pääsuunnittelu (MasterPlanning) prosessi käsittelee myyntitilauksen sekä selvittää, mistä osista konfiguroidut tuotteet muodostuvat ja millaisia työvaiheita tuotteiden tekemiseen tarvitaan.
4. Pääsuunnittelun (MasterPlanning) tuloksena syntyy toimituspäivämääriin pohjautuen raaka-aineiden ja osien ostoehdotukset ostajille haluttuihin toimituspäivämääriin pohjautuen.
5. Ostajat kilpailuttavat vahvistettavat ostot ja vahvistavat ne, jolloin eri osien toimittajien ERP järjestelmissä heidän tarjouksensa vahvistetaan myyntitilaukseksi. Lopulta toimittajat alkavat tuottamaan tilattuja osia, jotka lopulta toimittavat.
6. Lopulta osat saapuvat tähän tekevään yhtiöön ja osat kirjataan varastosaldoille, jolloin osat varautuvat myytyyn tilauksen osiksi.
7. Pääsuunnittelu (MasterPlanning) prosessi käsittelee taas materiaalien saapumiset ja mahdolliset ehdotetut tuotannot vahvistetaan tuotantoon taaskin toimituspäivään pohjautuen, kun osat ovat saapuneet.
8. Uudet syntyneet tuotannot ajoitetaan taaskin toimituspäivämäärään pohjautuen, jolloin työpisteisiin tulee konevaraukset ja työt löytyvät oikeassa järjestyksessä työntekijöiden ryhmien jonoista.
9. Työntekijät aloittavat omalla työpisteellään niitä töitä, joita ajoituksen jälkeen sinne on listoille ilmestynyt. Työntekijän ilmoituksen mukaan tuotannolle kertyy dataa, josta esimerkiksi työntekijän ilmoittama työaika, tehdyt määrät, kustannukset ja materiaalien otot ovat osa.
10. Työvaiheet etenevät sitä mukaa kun edelliset vaiheet valmistuvat. Seuraavat työvaiheet, tai näistä riippuvat tuotannot, edistyvät puolestaan, kun osia ja edellisiä työvaiheita valmistuu.

11. Lopulta asiakkaan ostamat tuotteet valmistuvat varastoon ja niitä lähetetään asiakkaan haluamissa erissä ja aikataulussa.

12. Lähetyksiä vastaan asiakkaalle lähtee myös lasku, joko etukäteen tai täysin rahtikirjojen pohjalta.

13. Koko tuotannon valmistuttua ajetaan tuotannosta loppuraportti, jolloin kaikki kustannukset lasketaan yhteen ja lopulta sitä verrataan toteutuneeseen myyntihintaan. Tuotantojen hinnat heittävät esim. materiaalivikojen ja työntekijöiden työajan kulun mukaan. Epäonnistuneita työvaiheitakin voi tulla, jolloin materiaalia ja työaikaa kuluu enemmän kuin alun perin oli suunniteltu tai materiaali joudutaan hankkimaan kalliimmalla hinnalla. Nämäkin kustannusten heitot tulisi mahtua suunnitellun voittoprosentin sisään.

14. Reskontra hoitaa asiakkaan laskutuksen valvonnan.

    Valvotaan, että laskut tulevat maksetuksi ja lähetetään muistutukset tai mahdolliset korkolaskut. Lopulta voivat jopa merkitä asiakkaan mustalle listalle, jos luotto myynti on täynnä.

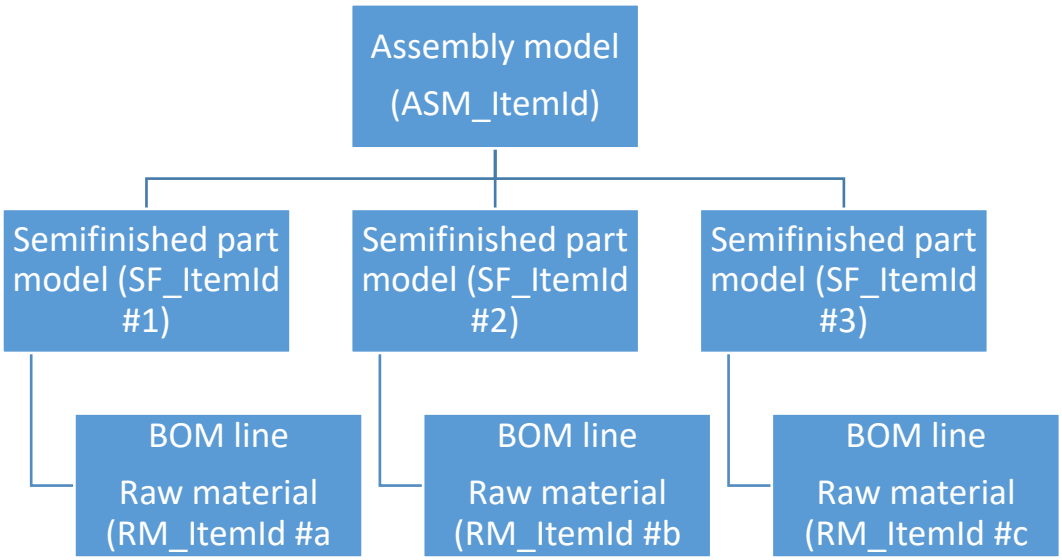
15. Lopuksi johtoryhmä tarkkailee tuotantojen voittoprosenttia, myynnin kannattavuutta ja yleistä taloustilannetta.

2.2 Mitä tuotteen konfigurointi ja modulaarisuus merkitsee?

Tehdas voi tehdä tuotteensa todella uniikisti, jolloin valmiissa tuotteissa kaikki asiat on suunniteltu ja toteutettu juuri täydellisesti tuotteen tarkoitusta vastaavaksi. Tämä tuo usein etua tuotteen kilomääräistä painoa ajatellen, mutta jotta tuotteiden tekeminen olisi halvempaa ja taloudellisesti kannattavampaa, niin tällaisia uniikkeja tuoterakenteita ja työvaiheita tulisi välttää.

> *Moduuli on itsenäinen osa, jollaisista voidaan koota erilaisia kokonaisuuksia. Moduuleista koostuva kokonaisuus on modulaarinen. (Moduuli 2015.)*

Modulaarisuudella tarkoitetaan tilannetta, jossa edellä kuvattu usean uniikin tuotteen yhteiset osat ja työvaiheet puretaan osiin etsien tuoterakenteesta ja työvaiheista yhteiset osat. Näin molemmat lopputuotteet voivat käyttää yhteisiä osia ja työvaiheita suuremmissa erissä ostettuna ja tuotettuna, jolloin saavutetaan massatuotannon etuja. Toisin sanoen tavoitellaan kaiken aikaa mahdollisimman modulaarista toteutusta.



KUVIO 2. Modulaarinen rakenne

Kuviossa 2 kuvataan tuotteen modulaarista rakennetta ja kaikkia alaosia. Myös alituoterakenteilla voi olla alituoterakenteita ja jokaisella rakenteella raakamateriaaleja, joista alimallit koostuvat eli raakamateriaaleja ja alituoterakenteita voi olla rinnakkain. Samat alituoterakenteet voivat olla osina muissakin päätuotteissa, jolloin puolivalmisteita voidaan tehdä moneen tuotantoon massana.

Konfiguroinnilla taas tarkoitetaan tilannetta, jossa valitaan osia ja ominaisuuksia tuotteelle.

> *Konfigurointi - asentaminen, asetusten säätö, asettaminen, konfiguraatio, virittäminen, asentaa. (Konfigurointi 2020.)*

Asiakaskohtaiset konfiguroinnit tekevät tuotteet erilaisiksi toisistaan, lähes uniikeiksi. Jotta voitaisiin saavuttaa massatuotannonetuja niin modulaarisuus auttaa asiassa, eli lähes uniikit konfiguroinnit saadaan sopivalla pilkkomisella modulaarisiksi. Konfiguroinnissa valitaan vain ominaisuuksia ja valinnat vaikuttavat muihin valintavaihtoehtoihin lisäten tai sulkien toisia pois.

> *Massatuotanto on samanlaisten tai hyvin vähän vaihtelevien tuotteiden valmistamista suurissa määrissä. Teollisen tuotannon etujen saavuttamiseksi halutaan valmistussarjojen koot mahdollisimman suuriksi, koska silloin tuotantotekniikkaa pystytään parhaiten tehostamaan, mikä nykyään tehdään pääasiassa automaation avulla. (Massatuotanto 2017.)*

Konfiguroinnista ja modulaarisuudesta löytyy monia konkreettisia esimerkkejä, joissa saman tuotteen erihintaiset versiot käyttävätkin samoja osia ja eroavat vain hyvin pieneltä

osin. Tuotteistuksen tuloksena toinen on halvempi ns. budget tuote ja toinen hintavampi premium tuote. Yleinen esimerkki erihintaisista tuotteista löytyy autoteollisuudesta:

1. Määritellään kaksi autoa erilaisilla ominaisuuksilla. Autonvalmistajat tarjoavat Internetissä myyntikonfiguraattoreita. Näissä konfiguraattoreissa valitaan ominaisuuksia kuten esimerkiksi: moottorin teho, vaihteisto, 2- tai 4-veto, väri, sisätilan penkkien ja sisustuksen värit sekä ominaisuudet elekronikassa. Ensimmäisen auton valinnat on kuvattu konfiguroinnin ajatuksen pohjalta ja toiseen autoon valitaan automaattisesti kaikki halvimmat vaihtoehdot ilman selitystä. Lopuksi tarkastetaan loppuasetelma.
2. Nyt voidaan jo huomata, että jotain samaa autossa voisi olla, kun asiakkaan valinta mahdollisuudet rajoittuvat vain ominaisuuksiin, joita halutaan käyttää eli ei tarvetta vaikuttaa lopultakaan kaikkeen mitkä asiat eivät vaikuta kaikkiin ominaisuuksiin.
3. Valitaan ykkösauton vetotapa 2wd vs. 4wd väliltä. Erimerkki simulaatiossa valitan 4wd.
4. Valitaan moottori konfiguraattorin tarjoamista versioista, joita 4wd –versioon valinnan pohjalta on enää olemassa.
5. Pienin ja taloudellisin moottorivaihtoehto on poistunut valintavaihtoehdoista. Seuraavaksi esimerkkisimulaatiossa valitaan diesel-versio kulutuksen minimoimiseksi isolla kuormalla liian suureksi.
6. Moottorin valinta vaikuttaa suoraan jarrujen kokoon, jota ei voi valita. Valittu dieselversio pakottaa konfiguraattorin tarjoamaan siihen sopivat vaihdelaatikot simulaation seuraavaksi valinnaksi.
7. Esimerkkisimulaatiossa tulee valittavaksi manual vs. automatic -vaihteiston väliltä. Simulaatiossa valitaan automaattivaihteisto.
8. Simulaatiossa autoon on nyt valittu perustekniikkaa, joiden pohjalta on mahdollista löytää modulaarisuuden hyötyjä.
9. Simulaatiossa voidaan pilkkoa auto seuraavaksi alituotteisiin, joista lopulta koko auto muodostuu ja erikoistuu.
10. Auton pohjalevy sisältää kiinnityskorvakkeet usein saman merkin kaikkiin malleihin, ja jopa erimerkkisiin sisarmalleihin, toisin sanoen kaikilla on sama pohjalevy. Kaikki erikokoiset ja tehoa tuottavat moottorit eri vaihdelaatikoineen voidaan kiinnittää samaan pohjalevyyn sekä peruskoriin. Lopulta auton sisarmallien erot sa-

maan pohjalevyyn tehdään vain ulkoisesti erilaisten lisämaskien avulla eri näköiseksi.

11. Autossa olevat elektroniikka, johtosarjat ja sisusta käyttävät usein täysin samoja perusosia. Johtosarja on usein täysin sama eri automalleilla, mutta kaikkia lisätoimintoja, niiden antureita ja niiden hallintanappeja ei kytketä kojelautaan, vaikka kojelaudassa ja sulakerasiassa paikka näille usein on olemassa mutta monesti siellä on vain peitelevy kytketyn lisäominaisuudensijaan.
12. Autosimulaatiossa modulaarisuus tarkoittaa siis sitä, että eri automallit rakennetaan samoista osista niin pitkälle kuin mahdollista. Eri automalleihin lisätään vain tarvittavat työvaiheet ja osat. Luxury sisar merkeissä (vertaa Toyota & Lexus) saattaa olla myös samoja perusmallin sähkökytkimiä.

Tämä kaikki tarkoittaa autojen alituotetoimittajan kannalta sitä, että perusosia voidaan tehdä ja myydä mahdollisesti usealle eri autotehtaalle, joista jokainen kokoaa auton ostajan konfiguroimilla halutuilla ominaisuuksilla. Esimerkiksi Bosch toimittaa moottorin hallinta yksiköitä moniin moottoreihin ja eroavat vain ohjelmallisesti eli monissa saman tyyppisiin moottoreihin menee sama ohjain, koska moottorin ohjauksen kannalta kaikkien polttomoottorien mitattavat ja hallittavat parametrit ovat samoja, riippumatta merkistä tai mallista. Samankin mallin sisällä samalla moottorilla ja samalla ohjaimella tuotettu teho saattaa erota esimerkiksi turbon ahtopaineen, muutetun polttonesteen suihkutuksen ja sytytyksen ajoituksen vuoksi. Näitä säätöjä kutsutaan moottorikartoiksi ja kaikki eroavaisuudet säädöissä on vain ohjelmallisia. Tietenkin auton valmistajat lisäävät omia mausteitaan urheilullisimpiin premium malleihinsa, kuten urheilullisimmat istuimet sivutuella ja joitain pieniä ulkoisia näyttäviä mahdollisesti kromi osia ohjaamon sisätilaan ja ulos korin puolelle joitain lähes turhia tuulen ohjaimia. Tietenkin joillain merkeillä ja malleilla jarrujen levykoot saattavat suurentua urheilullisessa mallissa, kuten mahdollisesti myös vaihdelaatikon ja akseliston vääntömomentin kesto.

> *Nykyaikaiset moottorit on varustettu moottorin hallintajärjestelmällä (EMS) / moottorin ohjausyksiköllä (ECU), jotka voidaan säätää erilaisiin asetuksiin tuottamalla erilaiset suorituskykytasot. Valmistajat tuottavat usein muutamia moottoreita, joita käytetään laajemmissa malleissa ja alustoissa. Tämä antaa valmistajille mahdollisuuden myydä autoja eri markkinoilla erilaisilla säännöillä ilman, että heidän on käytettävä rahaa kehitettäessä ja suunnittelemalla erilaisia moottoreita näiden sääntöjen mukaisiksi. Tämä mahdollistaa myös yhden tuotemerkin käyttämän yhden moottorin, joka on sovitettu ostajan markkinoille. (Moottorin viritys 2020.)*

Tämänkaltaisella toiminnalla saavutetaan massatuotannon edut, joita ovat suuret tuotantomäärät, jolloin yksittäisen osan kustannushinta jää pienemmäksi, koska materiaali määrät osiin saadaan hankittua suuremmassa erässä ja työreitin koneiden asetusajat jäävät yksikköä kohti pienemmäksi.

2.3 WebService

Web service on W3C:n määritelmän mukaan ohjelmistojärjestelmä, joka mahdollistaa keskenään yhteensopivan tietokoneiden välisen vuorovaikutuksen tietoverkon yli eli yleisesti internetin ylitse. Opinnäytetyössä mainittu HMPCE –laskenta moottori on toteutettu WebService –palveluna Azure ympäristössä. (Developer Guide 2020; Web service 2020.)

2.4 JSON

JSON (lyhenne sanoista JavaScript Object Notation) on yksinkertainen avoimen standardin tiedostomuoto tiedonvälitykseen. Tätä sanoma formaattia käytetään opinnäytetyön D365 tuotemallin parametrien välityksessä HMPCE WebService palvelulle ja kaikessa D365:n lähettämissä datoissa OData REST –rajapinnan lävitse HOTApi serverille. Myöskin HOTApiserverin väittämät tarjoukset HOT –clientilta tulevat JSON formaatissa. Tiedostonimestään ja JavaScript-perustastaan huolimatta JSON on JavaScriptistä riippumaton. JSONin Internet media type on application/json ja sen tiedostopääte on .json. (Dynamics 365 2020c; JSON 2020; What is JSON? 2020.)

2.5 Azure

Azure on Microsoft:in pilvijärjestelmä, joka tarjoaa yli 200 tuotetta ja palvelua helpottamaan uusien ratkaisujen tuomista käyttöön. Tarkoitus on helpottaa uusien palveluiden julkaisua pilviympäristössä, johon tarjotaan työkaluja ja hallinta ohjelmistoja ympäristön hallintaan. Opinnäytetyön D365 asennus on pilvi Azure ympäristö, eikä on-premises paikallispilvi, joka se voisi myös olla. On-Premises pilvien valintaan johtaa usein ajatus voiko yrityksen datat olla ulkomailla periaatteessa vieraidenvaltioiden ja oikeuslaitosten päätösten alaisina vai halutaanko data varmasti omistaa itse. On-premises paikallispilvi tarkoittaa käytännössä, että data on yrityksen itsevalitsemassa kotimaisessa palvelinsalissa, eikä suurissa keskitetyissä pilvidatacenterissä ulkomailla. (What is Azure? 2020; Dynamics 365 2020b.)

2.6 D365 Business Event

Mahdollistaa yrityksen liiketoiminnan tapahtumien välittämisen ulkoisiin järjestelmiin tapahtumien jatkumisen eteenpäin. Tätä menetelmää käytettiin opinnäytetyössä tarjouksen konfiguroinnin ja useiden perustietojen muutosten välittämiseen HOTApi –serverille ja sieltä tavallisena WebService kyselynä edelleen HOT –clientille. (Microsoft D365 2020a.)

2.7 XML

XML (Extensible Markup Language) on merkintäkielien standardi, joka määrittää tietojen merkintämuodon loogisella rakenteella (XML 2020). XML-kieliä käytetään, sekä formaattina tiedonvälitykseen järjestelmien välillä, että tiedostomuotona dokumenttien tallentamiseen. Tätä käytettiin luonnostaan esitallennettujen tuotemallikonfigurointi XML-tiedostojen tallentamiseen WMDPreConfiguredValues –tauluun, jotta automaattinen konfigurointi voitiin toteuttaa. D365 käsittelee konfigurointimallia XML-datana.

3 INTEGROITAVIEN JÄRJESTELMIEN TEKNISET YMPÄRISTÖT

Järjestelmäintegrointi tarkoittaa yhden tai useamman järjestelmän yhdistämistä niin että toisen järjestelmän tapahtuma johtaa tapahtumaan seuraavassa järjestelmässä ja tähän interaktioon liittyy usein myös tarvittavan datan tai vähintään sanoman siirtoa. ERP järjestelmässä on sisäisiä intergraatioita ERP-modulien välillä usein automaattisesti, mutta usein myös uusien ominaisuuksien muokkauksien yhteydessä lisätään tarvittaviin vastamoduuleihin toimintoja, jotta tapahtuma jatkaisi eteenpäin. Erillisjärjestelmien integroinnissa tapahtuma johtaa tapahtumaan toisessa ulkoisessa järjestelmässä esimerkiksi tässä opinnäytetyön tapauksessa, ulkoinen myyntikonfigurointijärjestelmä tekee myyntitarjouksia, jotka tuodaan yrityksen D365 ERP järjestelmään. Ulkoinen laskentamoottori laskee tuotujen konfiguroitavien tuotteiden tietoja ja palauttaa tuotannon rakenteisiin ja reitteihin tietoja, jotka vaikuttavat tuotteen osalta moniin ERP-järjestelmän moduuleihin.

3.1 Microsoft Dynamics ERP

Microsoft halusi päästä ERP tuotteita (Microsoft D365 2020b) tarjoavaksi ohjelmistotaloksi ja osti tanskalaisen Navisionin ennen AX4 versiota, jossa ei vielä näkynyt juurikaan Microsoftin muutokset. Suurin lisäys saattoi olla, että .Net –koodaus tuli mukaan perinteisen Axaptan oman X++ koodin lisäksi suoritettavaksi kieleksi kääntäjään. Seuraavan version AX2009 kohdalla jo näkyi Microsoftin käden jälki muutoksissa. GlobalAddressBook, joka tarkoittaa yhtä suurta osoitetietojen taulukokonaisuutta, oli suurin muutos. Tämän muutoksen mukana hävisi jotain alkuperäisestä ERP-tuotteesta, jossa taulujen relaatioiden ID -arvot olivat myös ihmissilmin luettavissa ja täten helpommin tunnistettavissa. Muutoksessa siirryttiin int64-tyyppisiin, 19-numeroisiin kokonaislukuihin. Näitä RefRecID-tyyppisiä pitkiä numerosarjoja käytetään viittamaan esimerkiksi asiakkaan eri osoitetietueisiin. Alun perin Navisionin ERP-ohjelmistoja olivat NAV pienille yrityksille ja Axapta suuremmille konserneille. Näiden ohjelmistojen nimet ovat nykyisin hieman muuttuneet.

Esimerkkinä mainittakoon, että Axaptan eri versiot olivat helposti muokattavia, koska ohjelmistoympäristö oli omaa X++ -kieltä käyttävä ja myös erittäin helposti opittava. Tästä syystä monet Axaptaa käyttäneet yritykset ovat itse koodanneet ympäristöihinsä toimintoja. Esimerkiksi opinnäytetyön kohdeyritys yritys-x on alun perin tehnyt itse toimintoja heidän ERP-ympäristöönsä, ja näitä toimintoja halutaan nyt saada mukaan uuteen D365 pilviympäristöön. Toiminnot ovat tehtävissä, mutta eivät enää yritysten itsensä tekeminä, vaan esimerkiksi konsulttiyrityksen toimittamina. Kaikki toiminnot ovat uudessa D365 pilviympäristössä CIL (common interprete language) -käännettyjä, jossa toimintojen käännös tehdään ja paketit toimitaan kohdejärjestelmään. Ennen konfiguraatioon tehdyt ehto koodit

ajettiin kääntäjällä (compiler), mutta se on poistunut. Eräs syy tähän on se, että pilviversioon ei haluta yrityksen itsensä kirjoittamia koodeja. Näissä koodeissa saattaa olla tehollisesti kuluttavaa, turhaakin koodia ja pilvipalvelun toimittaja on kuitenkin luvannut, että prosessointitehot pilvipalvelussa riittävät valitun käyttäjämäärän suorituskyvyn ylläpitämiseen.

3.1.1 Dynamics versiot

Tanskalainen Navision kehitti tuotteen alun perin aina Axapta 4 –versioon asti, sitten Microsoft osti yhtiön.

1. Navision 1.0 ja 2.0

   Alun perin IBM:n ja tanskalaisen Damgaard Data:n yhteinen projekti nimellä IBM Axapta.

2. Navision 2.5

   Sisälsi paljon jo samoja ominaisuuksia kuten tulevissa versioissa.

3. Navision Axapta AX 3

   Thin & Fat – client versiot & AOSit. Tämäkin versio on vielä käytössä.

4. Microsoft Axapta AX 4

   Microsoft osti Navison tuotteet ja kaikki pysyi aika paljolti samoina. DotNet –käyttö (.NET) suoraan X++ tuli mahdolliseksi.

5. Microsoft Axapta AX 2009

   GlobalAddresBook, surrogatekey relaatiot (RecId eli uniikit tietueen numerot) tulivat uutena ja DotNet –versiota nostettiin uudemmaksi.

6. Microsoft AX 2012

   CIL –käännös (common interprete language), Product Builder vielä mukana konfiguraatiossa ja uusi Regular Expression pohjainen –versio uutena mukana. Surrogatekey monessa paikassa käytössä eli taulun uniikki RecId on avain toiseen tauluun ja yleensä RefRecId –kenttään RefTableId:n kanssa eli taulun rivin uniikki nro. ja ko. taulun uniikki Id. Tässä on pohjalla suurempi suunnitelma, koska periaatteessa kaikki taulut voitaisiin liittää dynaamisesti toisiinsa sopivien välitaulujen kautta eli mahdollistaa dynaamisen raportoinnin luomisen, koska kaikki on kytkettävissä kaikkeen eli mahdollistaa DataMiningin, joka aiemmin on ollut toisten tieto-

jenlouhintaa tarjoavien tuotteiden hallinnassa esim. Cognos. Dynaamisella relaatiolla tarkoitetaan, sitä että eri tauluista ei tarvitse etsiä uniikkia relaatioavainta vaan se on RecId & TableId yhdistelmä suoraan. Nyt myös Reporting service korvaa jo osittain dynaamiset ohjelmoitavat vanhat tulosteet.

7. Microsoft Dynamics D365 FO pilviversio ja myös on-premise (paikallispilvi) versio saatavilla. Käyttöliittymissä työasemalle asennettavat Client .exe –ohjelmat ovat poistuneet kuten myös Enterprise portal, joka oli WEB –versio asennettavalle Clientille. Eli nyt kaikki käyttö perustuu WEB –selaimeen aivan kuten edeltänyt Enterprise Portal –käyttö. Lisäksi tuote konfiguroinnissa vanha koodattava Product builder on poistunut ja vain Regular Expression tapa tehdä ehtoja ja laskentoja on tarjolla.

Tämä D365 on nykypäivää, johon tämä opinnäytetyönikin perustuu ja on yritys-x:lle valittu D365 pilvi –versio, jossa kaikki on toisin kuin aiemmissa versioissa. Kehitysympäristö on Visual Studiossa ja aikaisemmat helpot, asiakkaan muokkaamat koodaukset ja raportit ovat jo historiaa. Toki D365:n näyttöjä voidaan vielä muokata rajoitetusti, mutta kuitenkin voidaan lisätä näyttöihin uusia datakenttiä. Aiempi syy valita Axapta, jossa asiakkaan toimesta tapahtuva muokkaaminen oli helppoa, on mennyttä ja nyt ollaan samassa veneessä monien muiden ERP –tuotteiden kanssa, jossa seurataan kaiken aikaa tulevien muutosten tarjoamia ominaisuuksia suhteessa siihen, pitäisikö ominaisuus koodata ko. järjestelmän kylkeen vai onko jokin ohjelmistopalvelunyritys tarjonnut tähän sopivan ominaisuuden. On siis menetetty muokkauksen helppous, nopeus ja halpuus. Asiakkaat saattavat vain odottaa kuka tekee sopivan ominaisuuden milläkin hintaa. Loppu oletus on, että kaikki yritykset toimivat samalla tavalla mikä onkin ehkä totuus mutta, jotta asiakkaat saadaan ymmärtämään, että heidän ehkä tulisi muuttaa toiminta prosessiansa eikä ERP:sä koodia, koska muutoksen hinta heidän vanhan järjestelmänsä mukaiseksi maksaakin yllättävän paljon. Muutoksen hinta lopulta ajaa yritykset samaan muottiin ja tekemään asioita taas itse Excelillä, josta jo päästiin pois, kun Axapta oli helppo ja halpa muokattava omin päinkin. Tilanne on kyllä paranemassa, koska MS julkaisee kovalla tahdilla uusia ominaisuuksia ja avaa omiin koodeihinsa kohtia, johon saa tehdä laajennuksia. Edelleen D365:n muokkaus on tavallaan kiertotien etsimistä, jos luokat ja taulut ovat private tai protected määrityksellä, jolloin laajennus ei onnistu ollenkaan. Ratkaisuksi täytyy siis löytää sopiva kohta, johon laajennus ja muutos tehdään ennen tai jälkeen suljetun kohdan. Lisäksi on Cag (Confidentiality Advisory Group) jolla pystytään pureutumaan suljettuihin osiin muttei kuitenkaan välttämään olemassa olevan Microsoft koodin ajamista ollenkaa, eli ERP prosessia ei voida estää etenemästä tai poikkeuttaa paljolti ollenkaan. On käytännössä tyydyttävä siihen mitä voidaan tehdä ja on muokattava prosessia sopivaksi olemassa ole-

vaan ERP:n business logikkaan, jotta riittävä muutos saadaan aikaan, joka kattaa asiakkaan tarpeet. (Microsoft Dynamics AX 2020.)

3.2 Dynamics AX2009

Dynamics AX2009 on käytössä opinnäytetyön kohdeyrityksellä yritys-x:ssä. Tähän versioon tuli siis suurena muutoksena kappaleessa 3.1 mainittu GlobalAddressBook. Edelleen kaikki oli ohjelmallisesti helposti muokattavissa ja yritys-x onkin tehnyt paljon omia koodeja ja saaneet konsultaatio tukea omiin tekemiinsä muutoksiin. Myöskin yritys-x:n kovasti käyttämä ja muokkaama tuotekonfigurointi, jossa muka oli yritys-x:n oma HmaxEngine –luokka ei siis mitään yhteyttä Java puolelle; tästä poikkeuksena ne koodit ja laskennat, jotka olivat kahdennettu ja jotka olivat molemmissa versioissa samaa koodausta. Tästä tosin aiheutui myös ongelmia, koska ajoittain versiot erosivat ja laskivat eri hintoja. Aiemmassa integroinnissa tehtiin siihen tarkistuksia, ja laskettiin virheprosentti sille, kuinka paljon ko. erot saivat olla tai sitten kyseessä oli tuotteen konfiguroinnin pysäyttävä virhe.

3.3 Dynamics D365

Dynamics D365 FO on lähtökohtaisesti pilvipalveluna myytävä ratkaisu, mutta siitä on saatavilla myös on-premise -versio, jossa pilvi asennetaan yrityksen paikallispilveksi. Yleensä on-premise -ratkaisuun on syynä se, ettei yritys halua tietojensa ja palveluidensa sijaitsevan ulkomaisilla palvelinkeskuksilla. Pilvipalvelussa on kyse vain palvelimista, joita hallitaan ja käytetään WEB –selaimella suoraan tai WebService –pohjaisesti esimerkiksi integroiduissa palveluissa ja ne voivat olla paikallispilviä eli paikallisia servereitä tai tuntemattomassa konesalissa eli pilvessä. (Dynamics 365 2020b.)

Aiemmissa Dynamics versioissa ennen D365 -versiota oli mukana MS Sharepoint web-portaaliin pohjautuva Enterprise Portal –Web sivusto, jossa oli monia ominaisuuksia, kuten esimerkiksi myyntitilauksen luonti ja tuotteen konfigurointi Extranet –tyyppisesti asiakkaidenkin käyttöön Web –selaimella käytettäväksi. Nykyinen versio on suoraan vain Web selain –pohjainen ja toimii myös tabletti ja mobiili alustoilla, jonka vuoksi työasemakohtaiset aiempien versioiden Client.exe -asennukset ovat historiaa.

3.4 HOT -Client, HMPCE ja HOTApi

HOT –Client aiemmin AX2009 version kanssa käytetty myös nimeä HMAX Client (Yritys-x AX) on yritys-x:n omaa ohjelmistokehitystyön tulosta ja tämä Client –ohjelmisto on toteutettu Javalla, joka toimii offine –tilassa. Tällöin ei tarvita jatkuvaa online-yhteyttä HMaxEn-

ginen serveriin, joka on uudemman version HMPCE edeltäjä, jotta Clientillä voidaan tehdä tarjouksia ja konfiguroida tuotteita tarjoukselle.

Client sisältää Javan Jar paketissa HMPCE (yritys-x Production Calculation Engine) tuotemallien laskennat, jotka siis aiemmin täytyi ohjelmoida AX2009 konfigurointiin omaksi luokakseen konfiguraattorin käytettäväksi. Tämä johti siihen, että tuotemallien hintaeroja esiintyi aika ajoin, koska Client:in mallit ja AX2009:n laskentaluokkien arvot ja kaavat eivät olleet täysin yhdenmukaisia. Lisäksi Client pakettiin kuuluu paikallinen tietokanta (DBLite), joka on myös Jar –paketissa mukana datoineen (HMPCE –laskukoneen arvot). Ratkaisu on käytössä myös jatkossa, jotta offline-ominaisuus saadaan säilytettyä.

HOTAPI –palvelin sisältää oman MYSQL –tietokantansa, johon kaikki datat ja luotujen tarjousten tilat siirtyvät D365:stä perusdatojen osalta. Clientin osalta luodut tarjoukset, jotka HOTAPI välittää taas D365:n puolelle edestakaisin ERP ketjun hallinnoitavaksi ja Clientin puolelle takaisin välitettäväksi, kun ne ovat D365:n puolella käsitelty. HOTApi –server myös hallitsee tilanteen, jossa myyntihenkilö ottaa omalla paikallisella Clientillaan yhteyden palvelimeen. Uuden version olemassaolosta ilmoitetaan ja sitä tarjotaan ladattavaksi. Toisin sanoen HMPCE -laskentamoottori ja HMPCE -tietokanta yleisesti muuttuvat useammin kuin itse Client -ohjelmisto, mutta tuokin muutos tulee samassa ladattavassa paketissa.

Tämä takaisin välityksenansiosta Clientit saavat nopeasti tiedon, kun tarjoukset ovat D365:n puolella menossa eteenpäin. Tämä on ollut aiemmin ja mahdollisesti jatkossakin asia, johon pitää löytää jonkinlainen parempi ratkaisu, koska myyntitiimi on hektinen ja toteutuneet kaupat halutaan heti eteenpäin. Tarjouksen tuonti ja konfigurointi D365:n puolella ottaa aikaa, koska kaikki sisään tuotavat tarjoukset täytyy uudelleen konfiguroida D365:n puolella. Uudelleen konfigurointi tuottaa oikeat tuoterakenteet tuotantoihin, työvaiheiden reitit ja työt tulevat luoduksi. Prosessi valmiista konfiguroidusta tarjouksesta vahvistetuksi myyntitilaukseksi ja siitä suunnitelluksi tuotantotilaukseksi ja vahvistetuksi tuotantotilaukseksi on standardia D365:n toimintoa. Ainoastaan automaattinen erätyönä tehtävä ohjelmallinen konfigurointi ilman käyttäjän käsin tehtäviä valintoja on toteutettava tässä opinnäytetyössä.

Suuri osa integraation lopputoiminnoista ovat siis jo standardi D365 -toimintoja, joten varsinaisesti integraatiossa ei tarvitse näistä asioista huolehtia. Vain uudet kentät, joka eivät ole standardia D365 FO:n kenttiä tulee lisätä integraatioon dataentiteettien –laajennuksena. Tarjousten importoinnissa on vain odotettava lopputulosta ja se ottaa aikaa, koska suoritettavaa koodia on paljon. Tietenkin kokonaisajoaika riippuu tuotteiden tuoterakenteista, mutta suoritusaika konfiguroinnissa riviä kohden on yritys-x:n suurim-

missa tuoterakenteissa noin 40 sekuntia. Tällöin tarjouksella, jossa olisi 200 kappaletta konfiguroituvia tuotteita veisi aikaa 8000 sekuntia, kun niitä siirretään D365 järjestelmään kaikkine prosessien ajoineen. Tässä ajassa on mukana konfigurointi D365:n puolella, jossa käynnistyy Universal PCAdaptor. Opinnäytetyössä toteutettu Automaattinen konfigurointi ja Universal PCAdaptor, joka konfiguroi tarjousrivin tuotemallin lähettämällä parametrit HMPCE –laskukoneelle. Paluu sanomassa palautuu tarvittavat parametrit takaisin, joilla tuotemalli konfiguroidaan. Tarjousrivit on konfiguroitu kertaalleen jo HMAX Clientin puolella, mutta tämä on tehtävä uudelleen automaattisesti, jotta tarjous olisi vahvistettavissa myyntitilaukseksi mahdollisimman nopeasti prosessin loppuvaiheessa ja saadaan tuotantoon. Tässä uudessa versiossa on etuna se, että kaikki konfiguroinnit pohjautuvat samaan HMPCE –tuotemallin laskentaan, jolloin hinnat ovat varmasti täysin samat. UniversalPCAdaptor huolehtii HMPCE integroinnista, jolloin koodien kahdennusta ei tarvita. (Dynamics 365 2020a.)

4 AX2009 JA D365 VERSIOIDEN KONFIGUROINNIN EROAVAISUUDET

Yritys-x:n vanhan ERP:in ja uuden D365 versioiden välillä on eroja, jotka yritys-x tunnisti ERP projektin ns. ”show stopperiksi”. Tämä tarkoittaa sitä että, jollei tuotemallinnusta saada toimimaan standardi D365 versiolla niin päivitysprojektia pitää tutkia lisää, koska nykyisen tuotemallinnuksen idea on ollut riittävän tarkka modulaarisuus. Modulaarisessa tuotemallinnuksessa osat ja niiden ali-tuoterakenteen aina alas asti on mallinnettu omiksi tuoterakenteiksi, jolloin konfiguroinnin täytyy pystyä alirakenteiden alirakenteissa muokkaamaan osia. Lisähaasteen tähän tuo modulaarisuuden ominaisuus, jossa alamalleja pitäisi pystyä käyttämään toisissakin malleissa. Modulaarisissa tuoterakenteissa asiakkaan haluamat ominaisuudet ovat usein alimmilla tasoilla, jolloin kaikkien tarvittavien parametrien on vaikutettava kaikkien rakenteiden lävitse alimmille tasoille asti, jotta ne kelpaavat useisiin ylemmän tason tuotteisiin osiksi ja sitä myöten vastaavat asiakkaan tarvitsemaa tuotetta eli räätälöityä jopa mittatilaus tuotetta. Halutut ominaisuudet saatetaan haluta erilaisiin mittoihin, jotta ne sopivat asennettavaan kohteeseen.

Nykyisin tuotannossa olevassa AX2009 versiossa on käytössä standardi Product Builder moduuli, jossa ehdot kirjoitettiin X++ koodilla, joka on Javan ja C++ hybridi. Ehtoihin, joilla tuotteiden osia ja työvaiheita haluttiin vaihdettavaksi konfiguroinnin valintojen pohjalta, tehtiin vain pieniä X++ if-else -ehtoja parametrien ja muiden taulujen pohjalta; esimerkiksi select X from taulu where ehto == on_haluttu. Täydet koodausoikeudet olivat olemassa, jopa ulkoiset luokat, joissa voitiin tehdä ihan mitä vain, jopa Del *.* tyyppisesti. Tällä komennolla voidaan poistaa kaikkea mahdollista dataa, kun se vain kohdistetaan, kun tuotemallia konfiguroidaan. Tämä oli ensimmäinen esimerkki, miksi D365 on todella rajoitettu toiminnallisuuksiltaan. Toinen suurempi syy on se, että AX2009:ssä koodit ajettiin kääntäjän lävitse koodisolmun sisällön mukaisesti ja saatu vastaus ratkaisi ehdon mukaisen valinnan tai poisti mahdollisesti kaiken, jos niin oltiin koodi kirjoitettu tahallaan. D365:ssä kaikki koodit ovat CIL esikäännettyjä, jonka vuoksi compiler –tyyppinen, kääntäjän lävitse ajettava -merkkijono tyyppinen koodi, on mahdotonta. Tässä on kysymys myös suorituskyvystä, koska D365:ta on tarkoitus tarjota pilvipalveluna, jolloin asiakkaan itse tekemiä koodeja ei haluta suorittaa tai ainakin niiden kuormitusta järjestelmälle halutaan hallita (katso kappale 3.1). Nyt kaikki ajettavat koodit ovat esikäännettyjä ja ne validoidaan, jotta voidaan varmistua siitä, että koodit täyttävät hyvän ohjelmoinnin piirteet. Microsoft on antanut valitulle pilvipalvelulle ja käyttäjämäärälle suorituskykylupauksen, jonka vuoksi kaikki ajettava koodi tulee validoida sekä CIL-esikääntää.

D365:ssä kaikki ehdot rakennetaan Regular Expression eli säännönmukainen lauseke -tyyppisten ehtolauseiden tapaan. Suoraa koodia ei kirjoiteta, vain parametrien ehtoja, se-

kä tuloksia toisiin parametreihin, että tuoterakenteiden ja reittien arvoihin tai attribuutteihin. Attribuuteilla tarkoitetaan muuttujia, joiden arvoja voidaan välittää läpi konfiguroinnin ja ehdollistaa näiden arvojen pohjalta muita valintoja.

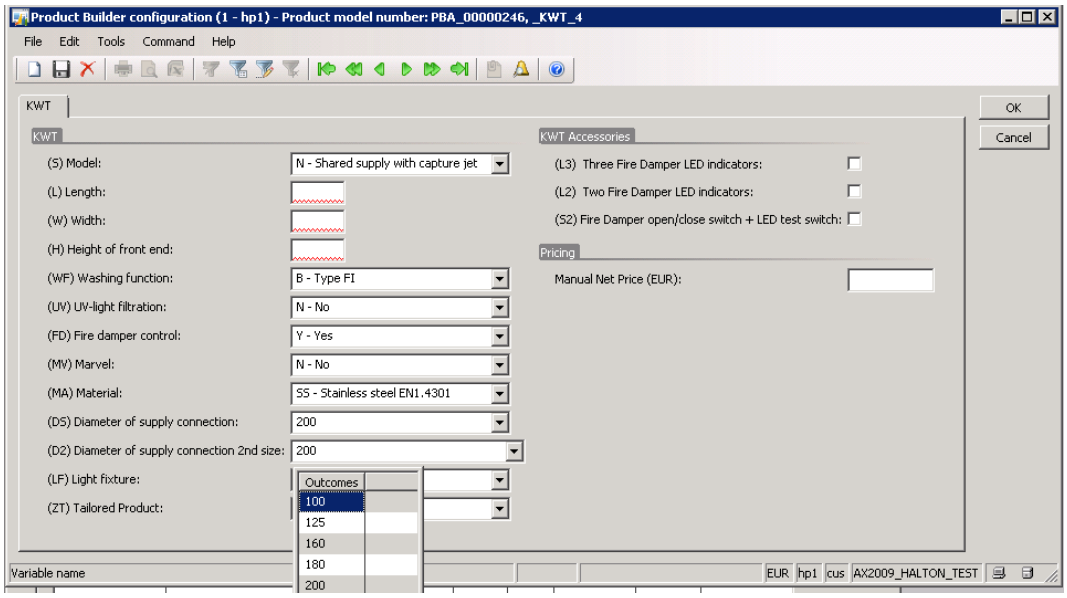
4.1 AX 2009 konfigurointi

Tämä versio on yritys-x:n käytössä olevan ERP:n konfigurointi, joka käyttää aiemmin esiteltyä kääntäjää (compiler), jossa on mahdollista merkkijono pohjaisesti suorittaa merkkijono dataa eli kaikkea koodia ilman esikäännöstä ja validioitia koodin sisällöstä. Esimerkit ovat lähes samoja uudemman version esittelyssä (kappaleessa 4.2), jolloin vertailu helpottuu.

4.1.1 Konfiguroinnin valintaikkuna

Tämä on valmiin tuotemallin konfiguroinnin osien tuottava valintaikkuna, joka käynnistyy aina kun tarjoukselle tai myyntilauksen riville valitaan konfiguroitava tuote. Valintaikkunassa näkyvät parametrit on tuotemallia rakennettaessa määritelty arvoiksi, joita täytyy kysyä.

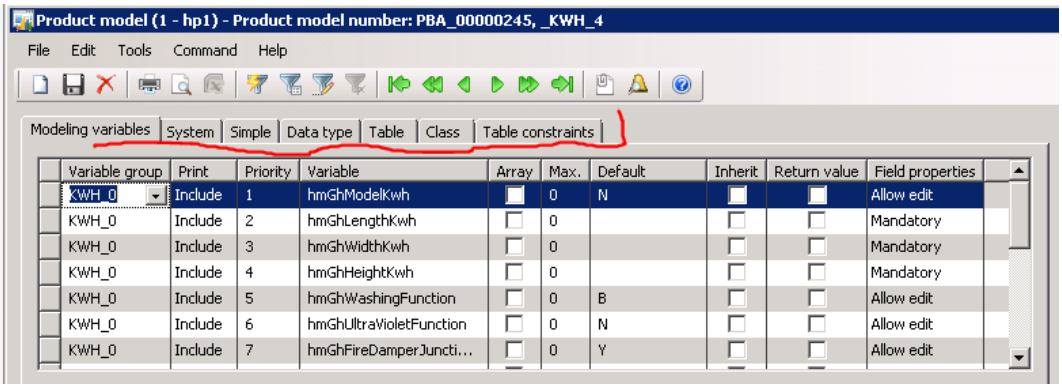
Ikkunaan valitaan arvot ja arvojen valinnat muuttavat valinta ikkunan sisältöä, koska se voi avata uusia valintoja, kun valitaan jokin alimallin osa, joka tarvitsee oman lisä konfiguroinnin. Kuvassa 1 esiintyvät kyselyikkuna on AX2009 Client ohjelmasta eikä Enterprise Portal version WEB sivuja, jotka ovat lähellä nykyistä D365 konfigurointia. Enterprise portal ja D365 ovat molemmat täysin WEB –selaimella käytettäviä.



KUVA 1. AX2009 konfigurointinäyttö

4.1.2 Konfiguroinnin perusasiat ja muuttujat

Parametrit tulee esitellä jokaisen oman tyypin alle, jotta niitä voidaan valita solmuihin ja kutsua koodissa (kuva 2). Lisäksi parametreille ilmoitetaan näkyvätkö ne valintalistoilla.



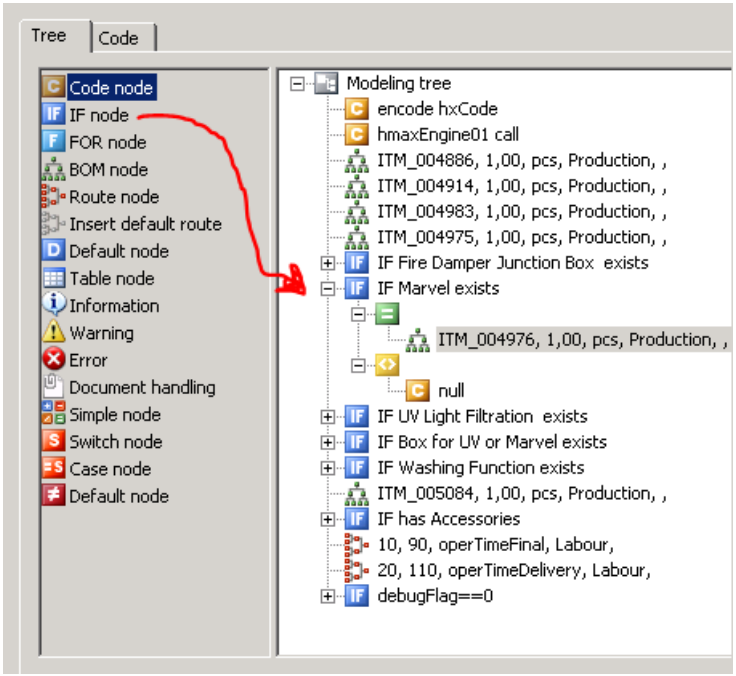
KUVA 2. AX2009 perusmuuttujat

Jokaisen tab -valinnan takana on tyypin ryhmään kuuluvat muuttujat esitelty.

4.1.3 Ehdot tehdään koodisolmuilla

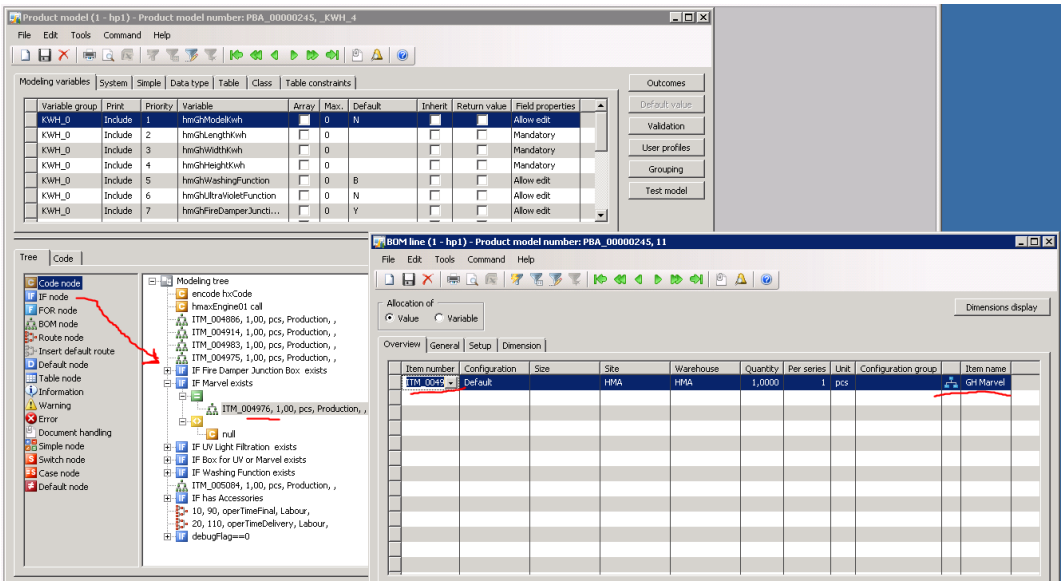
Koodin perusrakenne vedetään Modeling tree solmuiksi ja niitä voi tarvittaessa kopioida, mikä helpottaa suurempien kokonaisuuksien rakentelua. Tällöin koko rakennetta ei tarvitse rakentaa aina alusta alkaen, vaan vaihdetaan koodisolmuihin vain laskettavia asioita ja se, miten ko. kohta tulisi vaikuttamaan ehtojen kautta lopputulokseen; reitit, ajoajat, lopulliset työt kuormituspaikkoineen ja mallin rakenteet.

Mallia ajetaan ylhäältä alaspäin eli esimerkiksi HmaxEngine –luokka, jossa laskenta tässä vanhassa versiossa suoritetaan täytyy ajaa ensin (vertaa: D365:n kanssa käytetään HMPCE –laskenta moottoria UniversalPcAdaptor –luokan lävitse WebService -kutsuna), jotta ehtojen muuttujille eli attribuuteille saadaan arvoja. Siksi se näkyykin ihan ensimmäisellä rivillä Model Tree:ssä kuvassa 5, jossa se alustetaan HOTCoden pohjalta init-metodin suorituksen vaiheessa, jolloin HOTCode viedään parametrina luokalle sisään.



KUVA 3. AX2009 Model Tree

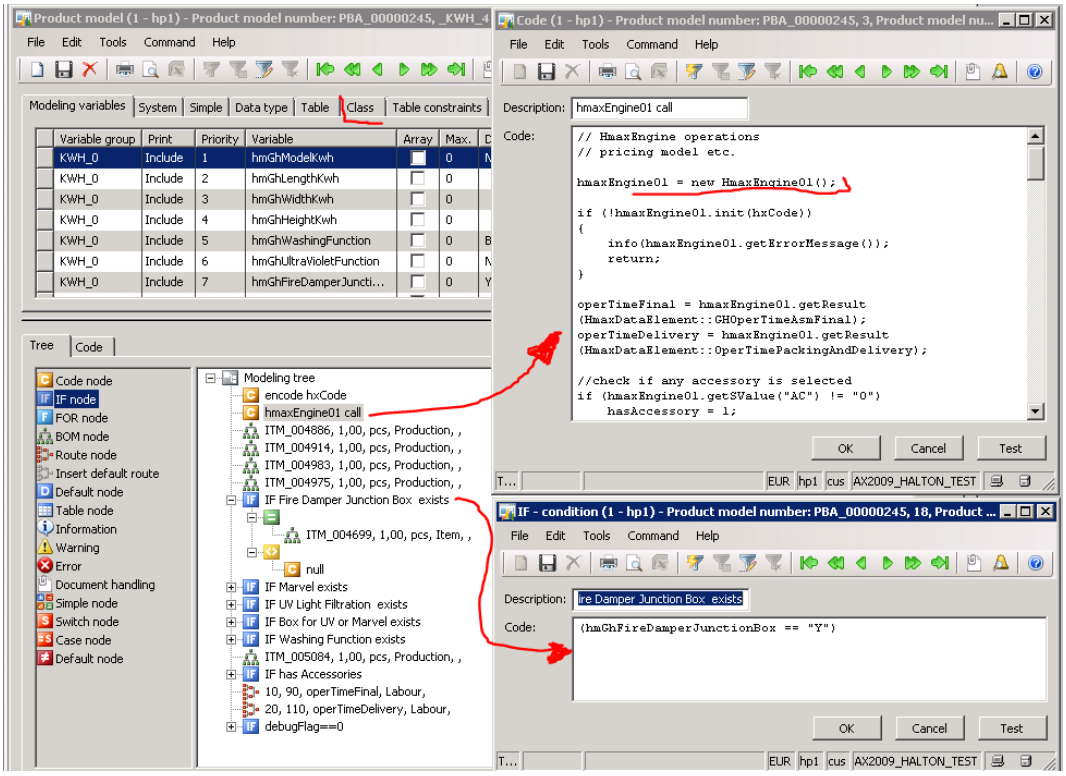
Solmuille annetaan ehtoihin kohdistuvat nimikkeet / alituoterakenteet, joita ehdoilla halutaan hallita. Eli näissä ..if exists lausessa testaillaan Marvel boxin olemassa oloa eli konfiguraattorin valintoja ja sen mukaan lisätään sekä osia että kuvassa 4 näkyviä työreitin työvaiheita. Toisin sanoen, jollei haluta tuotteeseen kaikki ominaisuuksia niin valitsematta jääneitä osia ja työ vaiheita ei lisätä tuoterakenteeseen eikä työreitin vaiheisiin.



KUVA 4. AX2009 Model Tree ja reitin määritys

Kaikki ehdot ja muuttujien asetukset. Koodit kirjoitetaan itse Notepad-tyyppiseen tekstieditoriin, tallennetaan tietokantaan ja suoritetaan lennossa kääntäjän (Compiler) lävitse.

Kuvassa 5 on nähtävissä myös koodisolmun testaus –nappi, jolla voidaan testata juuri tehty koodi.



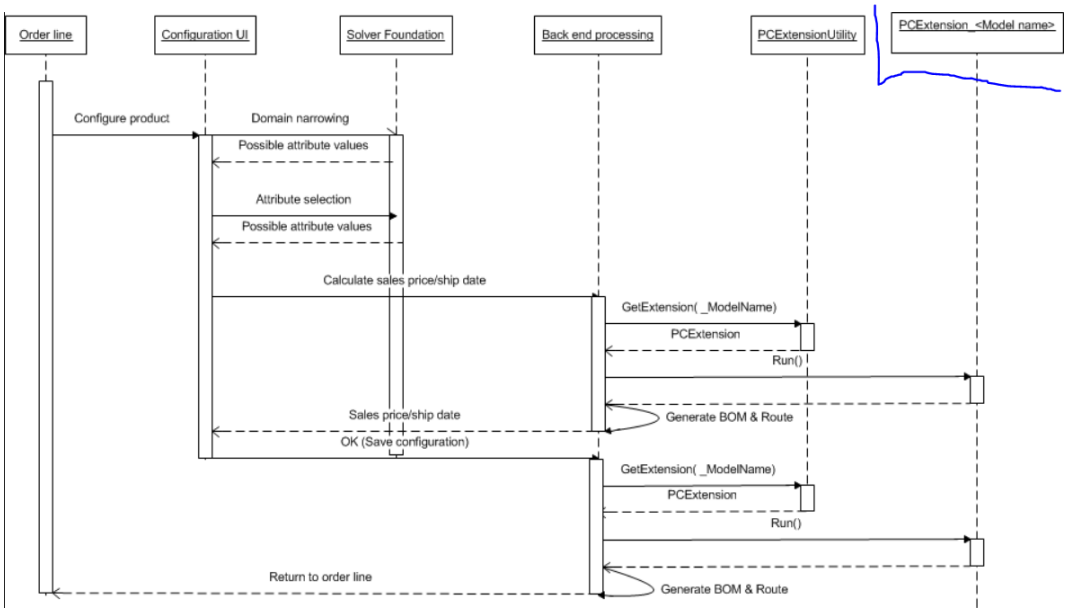
KUVA 5. AX2009 Model Tree ja koodisolmut

Kuten esimerkiksi HmaxEngine luokan kutsu kuvassa 5, luokan laskentojen alustus, jossa viedään valintanäytön valitut parametrit sisään ja laskenta-arvojen tulokset sijoitetaan mallin muuttujiin, joita sitten ajon edetessä testataan ehdoissa, joilla rakennetaan tuoterakenne ja työvaiheet. Tämä lähestymistapa on ollut monesti asiakkaan mieleen, mutta nyt vuosien jälkeen on nähtävissä, että muutokselle on olemassa tilaus, koska Notepad-tyyppisessä vapaata tekstiä sisältävässä tekstikentässä ei ole normaalia AX:n tarjoaman IDE:n syntaksin tarkistusta, eikä se auta koodivirheen korjauksessa mitenkään. Rajoittuneempi D365:n lähestymistapa aiheutti aluksi pettymystä. Uusi lähestymistapa ohjasi niin asiakasta, ettei valinnoissa voinut tehdä virheitä, sillä he vain valitsivat verrattavat asiat ja ehdot listoilta, D365 formatoi syntaxin ja kertoi virheistä. Aluksi rajoittunut tapa toteuttaa asioita oli hankala, mutta myöhemmin tavasta löydettiin paljon hyvää.

AX 2009 versiossa virheen löytymiseksi koodi oli kirjoitettavat Job:ksi, joka oli siis yksittäin ajettava koodisolmu. Sen luontiin oli käytössä kunnolliset IDE:n tarkistukset ja avustajat. Koodin mentyä käännöksestä läpi vain kopioitiin tarvittavat osat takaisin konfiguroinnin koodisolmun Notepad –tyyppiseen kenttään. Näin oli tehtävä, jotta saatiin syntaxin täysi tarkistus IDE:n kautta varmistettua ongelmallisissa koodilohkoissa.

4.2 D365 konfigurointi

D365:ssä tapahtuu yritys-x:n uuden ERP:n konfigurointi, joka käyttää aiemmin esiteltyä CIL-esikäännettyjä ja validoituja koodeja, jonka vuoksi ehdotkin valinnoille ovat Regular Expression pohjaisia (esitelty kappaleessa 3.1). Esimerkit ovat lähes samoja vanhemman version esittelyssä (kappaleessa 4.1), jolloin vertailu helpottuu.



KUVIO 3. D365 konfiguroinnin prosessin vaihejärjestys

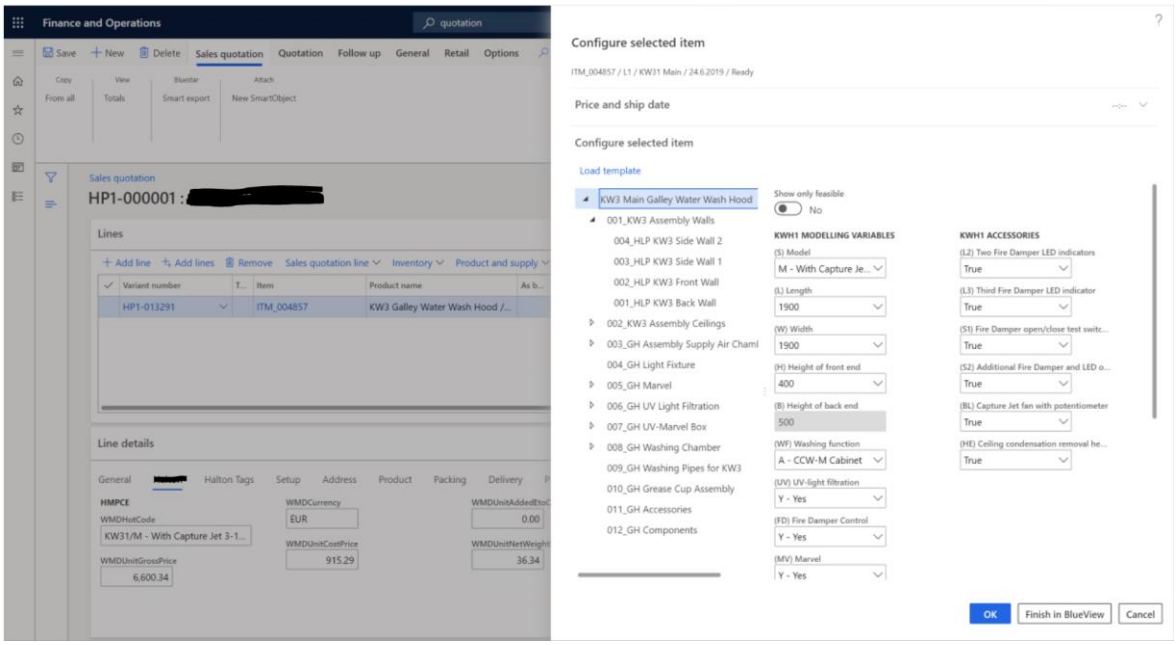
Kuviossa 3 esitetään D365 konfiguroinnin kaikki vaiheet aikajärjestyksessä. Sinisellä merkitty osa on PCAdaptor, jonne voidaan vapaasti tehdä koodia, joka tietenkin validoidaan ja CIL-esikäännetään. Opinnäytetyössä toteutettiin UniversalPcAdaptor, joka ajetaan juuri tuossa vaiheessa kaikille konfigurointia vaativille malleille. D365 käyttöliittymän kautta ei ole enää mahdollista kirjoittaa koodia näihin ajettaviin luokkiin kuten ennen D365 versiota, jolloin konfiguroinnin koodit olivat string-muotoista dataa, jota ajettiin kääntäjän (compiler) lävitse lennossa. (Microsoft Dynamics 365 2020.)

4.2.1 Konfiguroinnin valintaikkuna

Tämä on valmiin tuotemallin konfiguroinnin osien tuottava valintaikkuna, joka käynnistyy aina kun tarjoukselle tai myyntilauksen riville valitaan konfiguroitava tuote. Valintaikkunassa näkyvät parametrit on tuotemallia rakennettaessa määritelty arvoiksi, joiden arvoja täytyy kysyä konfiguroinnin aikana.

Valitaan arvot ja arvojen valinnat muuttavat valintaikkunan sisältöä, koska se voi sulkea eli poistaa valintoja laittamalla tarpeettomat kentät harmaiksi kuten kuvassa 6. Harmaisiin kenttiin ei voi antaa arvoja, jolloin edellinen valinta on ollut poissulkeva. Edellisissä versi-

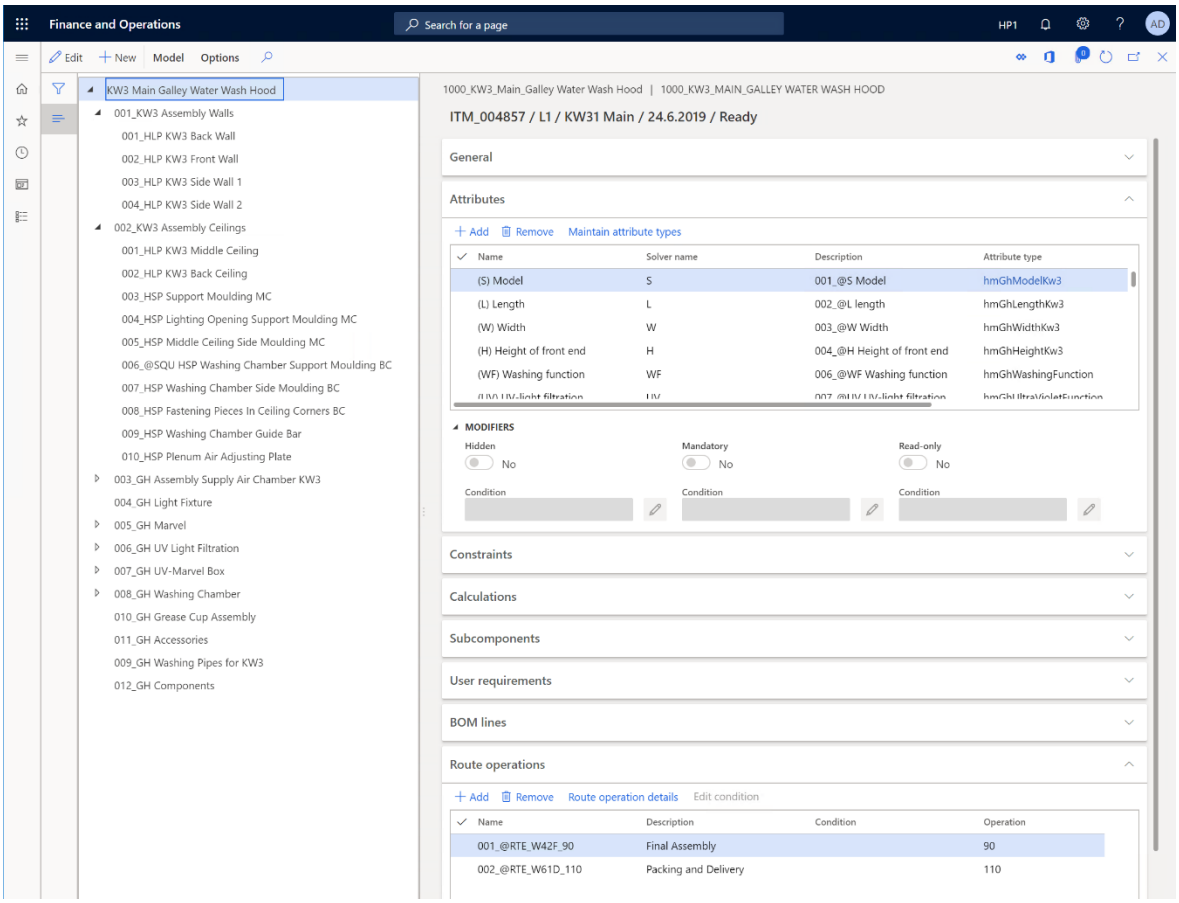
oissa valintojen vaikutus oli dynaamisempaa. Tämä johtui siitä, että uudessa versiossa koko käyttöliittymä on täysin HTML-WEB pohjainen, jolloin tietynlaisia näytön muokkauksia on hankalampaa toteuttaa, koska Web selaimet renderöivät HTML–koodin sisällön eli muodostavat tilan ja lohkot osilleen.



KUVA 6. D365 konfiguroinnin valintanäyttö

4.2.2 Konfiguroinnin mallin perusasiat ja muuttujat

Päätuotemalli näkyy vasemmalla kuvassa 7 ylinnä. Päämallin alamallit ovat omina lohkoinaan ja niidenkin alikokoonpanot ja rakenteet aukeavat myös puumaisesti tähän, jos vain avaisi pienen harmaan kolmion. Valitsemalla jonkin osan vasemmalta aukeaa oikealle tieto kohteesta ja jokainen vetolista aukaisee omat tietonsa (kuva 7) niiden attribuuttien kohdalla auki, jotka on valittu näytettäväksi. Kaikki mallin tiedot rakennetaan näin. Kaikki alimallit ovat yksilöitä ja näkyvät myös itsenäisinä kokonaisuuksinaan, jonka vuoksi niitä voidaan käyttää muissa pää- ja osakokoonpanomalleissa osina. Näin ei tarvitse kirjoittaa jokaiselle päämallille koko rakennetta aina uudelleen vaan hyödynnetään modulaarista tuotemallinnusta.



KUVA 7. Tuotemallin rakenne ja attribuutit

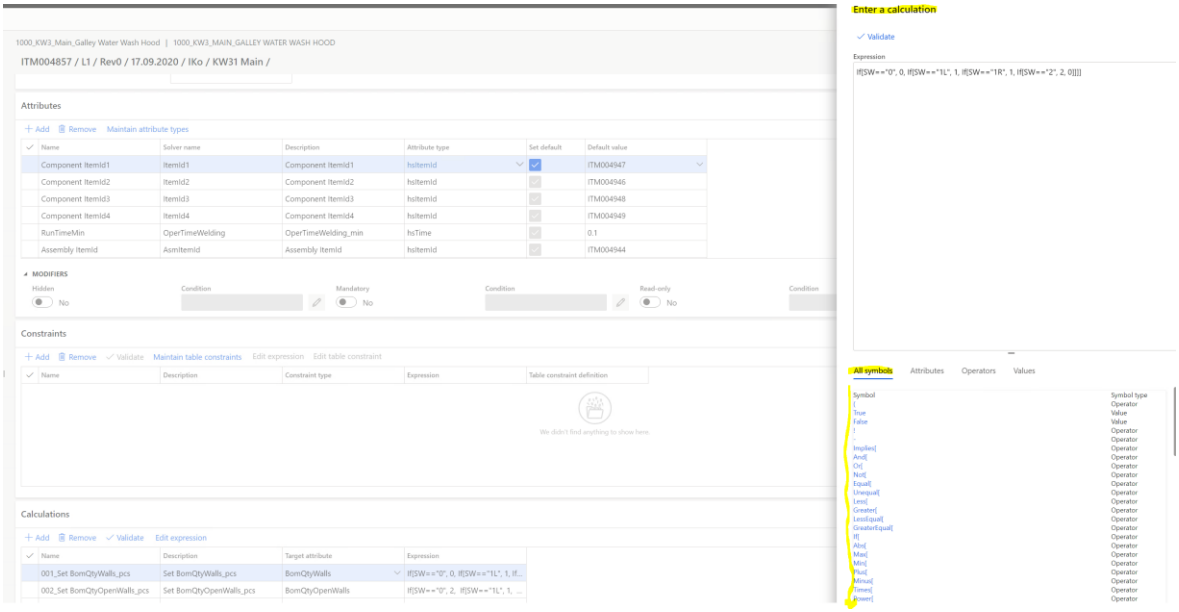
Kuvassa 7 on näkyvissä päämalli avattuna ja oikealla on mallin osat esimerkiksi kaikki attribuutit vetovalikossa, joita voi muokata ja lisätä että poistaa.

4.2.3 Ehdot tehdään Regular Expression pohjaisesti

> *Säännöllinen lauseke (engl.* ***regular expression****, lyhyesti* ***regexp*** *tai* ***regex****) on tietojenkäsittelyteoriassa lauseke, joka määrittelee säännöllisen kielen. Kieli tarkoittaa tässä yhteydessä joukkoa merkkijonoja. ... Säännölliset kielet voidaan tunnistaa äärellisillä automaateilla. (Säännöllinen lauseke 2019.)*

Ehdot tehdään Regular Expression –pohjaisesti, joka tarkoittaa sitä, että normaalia koodia ei kirjoiteta kuten aikaisemmissa versioissa vaan tehdään ehtoja, vertailuita ja laskentoja arvojen aikaan saamiseksi. Voidaan esimerkiksi ehdollistaa valintalistojen arvojen näkymistä riippuen toisista valinnoista tai valita osien tulemista tuotemallille valintoja ehdollistamalla.

Alla olevissa kuvissa 8 ja 9 on avattu laskentojen syöttö editori.

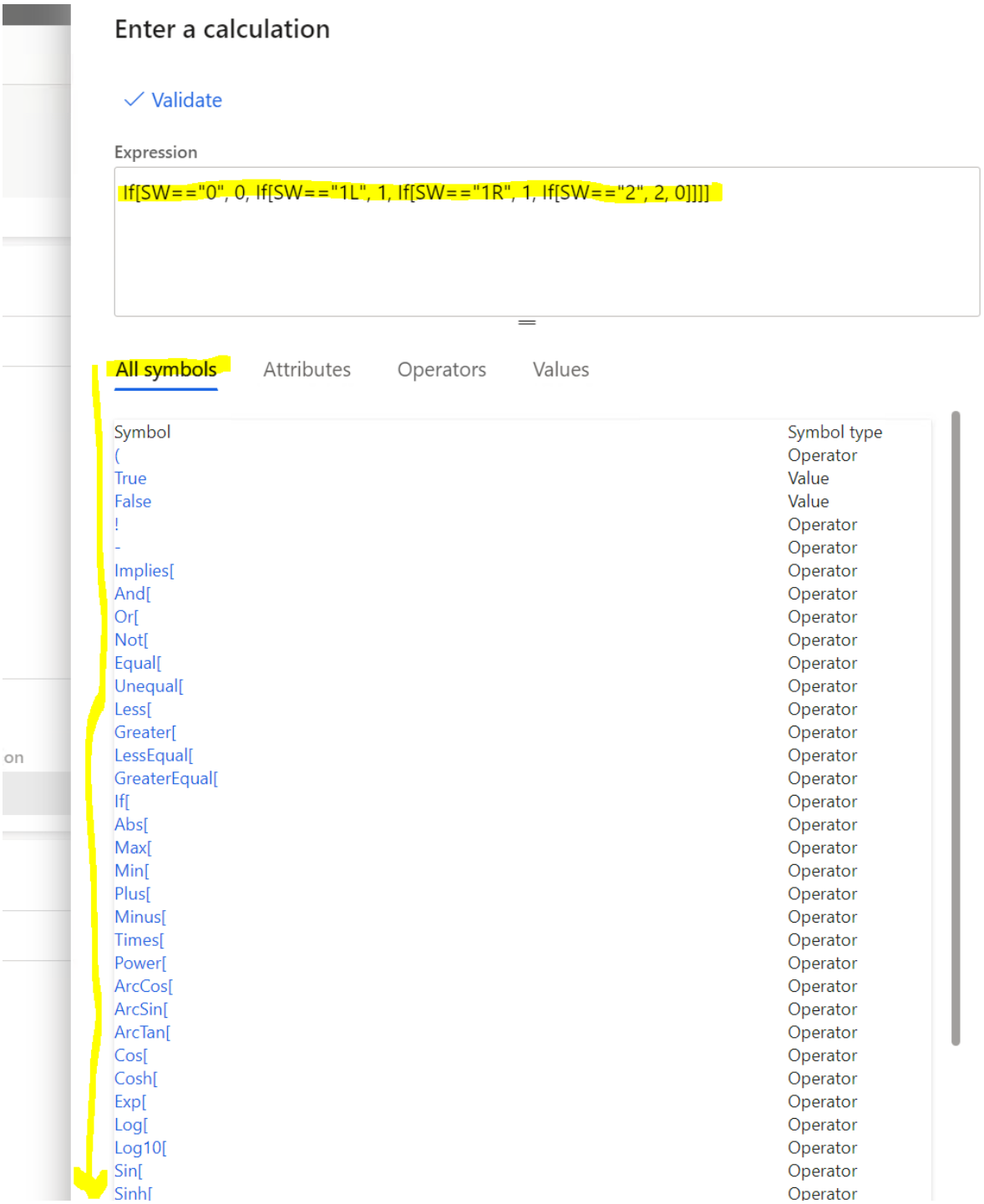


KUVA 8. D365 attribuutit ja laskentakaavat

Kuvassa 8 on yleisnäkymä näytöstä, jossa on valittu kohde, jonka laskentoja oikealla halutaan muokata.

Kuvassa 9 on suurennettuna Regular Expression editor, jossa kaikkia attribuutteja, operaattoreita, operaatioita ja funktioita voidaan valita suoraan listoita. Palikat voidaan kirjoittaa suoraan ja lopuksi validoida, jotta se on kunnossa. Kaikki listoilla olevat arvot on määritelty, joko Microsoftin puolesta tai mallin attribuuteissa ja muissa osissa, joka on mallikohtaisia ja alamalleilla myös eli jos ko. alamalli on käytössä toisaalla niin samat laskennat ovat olemassa valmiiksi. Voidaan esimerkiksi välittää arvoja päämallilta alamalleille, jolloin alamallit osaavat toimia kuten päämalli haluaa.

Tämän kaltaisessa toiminnassa täytyy olla yhteneväinen laskenta ja attribuuttien nimeäminen hyvin hallussa eli on tehty yritykselle tietty tapa tehdä modulaarista konfigurointia.



KUVA 9. D365 Regular Expression editorin lähikuva

Kun kaikki arvot on saatu rakennettua mallille, niin sitä voidaan testata tai kuten opinnäytetyössä tarvitaan vielä seuraava vaihe, jossa avoimet HMPCE –laskenta moottorin puolella pidettävät parametrit käydään hakemassa.

4.2.4 Jälkikonfigurointi ohjelmoimalla PCAdaptor –luokkia

AX2012:ssa esiteltiin uusi jälkikonfigurointitapa, joka tapahtuu ohjelmoimalla PCAdaptor –luokkia. Mukana pidettiin vielä ProductBuilder –tapa, jossa ehdot saatiin kirjoittaa X++ kielellä Notepad-tyyppiseen tekstieditoriin, kuten yritys-x:llä on nykyisessä AX2009 versiossa.

Ajatus tässä uudessa tavassa on, että RegularExpression ehtojen jälkeen mallin konfiguroinnin tuloksia voidaan vielä ohjelmallisesti muokata, mutta ehto-attribuuttien ehtoja ei kuitenkaan ajeta uudelleen, jolloin niiden ohjelmallinen muuttaminen ei vaikuta tehtyihin Regular Expression -loppuehtojen ehtojen tuloksiin. Voidaan vain muokata lopputulosta: reittejä ja tuoterakennetta, joka käytännössä konfiguroinnin jälkeen syntyy tuotannolle tai jos reittien tai tuoterakenteiden arvot on valittu attribuutti tasoisiksi niin sitten nämäkin muutokset menevät paikalleen kuten yritys-x:n tapauksessa tehtiin. Tämän luokan ajettavassa koodissa voidaan luokan käynnistävän myyntitilaukselle tai tarjouksen riveille välittää helposti arvoja standardi-kenttiin tai tarvittavia tietoja omiin lisättyihin kenttiin.

Tämä luokka, kuten kaikki D365 -luokat, ovat myös CIL -käännettyjä, jonka vuoksi sekin validoidaan. Oletuksena tämä on tehtävä jokaiselle mallille erikseen, vaikka sisällöllisesti se kelpaisi kaikille.

Tässä jälki ohjelmointi mahdollisuudessa on sellainen ikävä ominaisuus, että jos ehdot on tehty huolella ja mahdollisen muokkauksen jälkeen vain attribuutteja muokkaamalla päästäisiin oikeaan tulokseen, se ei onnistu, jos joudutaan vielä suorittamaan Solver –vaihe, jossa laskennat suoritetaan. Joudutaan siis muokkaamaan lopullisia BOM ja Route rakenteita, joka on työläämpää kuin muuttaa laskennallisia attribuuttien arvoja ja antaa laskentojen mennä uudelleen. Joudutaan siis ohjelmallisesti rakentamaan useampia loop –rakenteita ja miettimään hieman tietorakenteen kokonaisuutta, jotta muokataan juuri oikeaa kohtaa.

Ratkaisuksi saatiin opinnäytetyössä niin hyvä kokonaisuus, että se ei ollutkaan niin hankalaa ja työlästä, koska tähän löydettiin helpottavia toteutusmenetelmiä, johon palataan lopputoteutuksen esittelyssä.

5 LOPPUTULOS JA MITEN SIIHEN PÄÄSTIIN

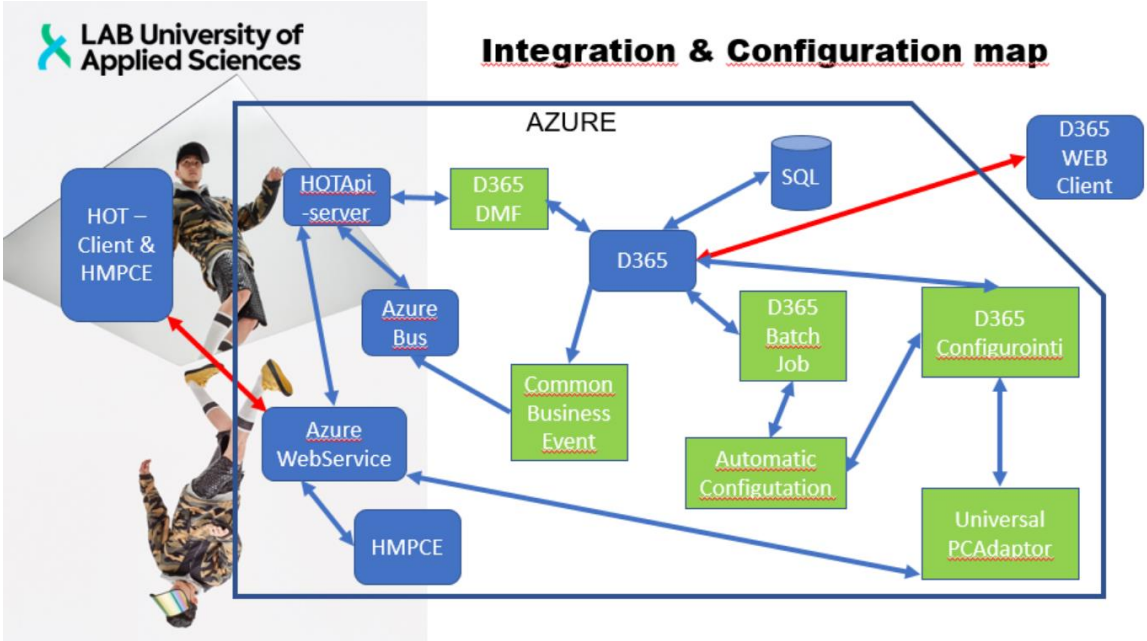
Tämän kehittämistyön pohjalta voidaan huomata, että versiot eroavat ihan syystä toisistaan ja syykin on selvä. Ei ole mielekästä antaa Microsoftin puolelta antaa palvelulupausta toimivasta järjestelmästä, jos asiakas saa itse ohjelmoida koodin sekaan omia epätehokkaita koodejaan. Tämä johtaisi siihen, että palvelinkeskuksissa pitäisi valjastaa runsaasti palvelimia murskaamaan asiakkaan tehotonta koodia, kun lupaus on, että kaikki toimii hyvin valitulla käyttäjämäärällä.

Toisaalla voidaan myös tehdä asioita, jotka johtavat suorituskykyongelmiin vain tekemällä toimistossa turhan raju raportti liian huonoilla tietokannan indeksointiavaimilla, jotka eivät osu. Huonon kyselyn seuraus voisi olla, että tuotantohallin työntekijöiltä tippui rukkaset 15 minuutiksi, kun vanha Axapta jumitti täysin eli juurikin tämän tyyppisiä asioita näillä koodauksen ja raportoinnin rajoituksilla haetaan, koska palvelulupaus on annettu.

Alusta asti suurin opintotyön ongelma oli se, miten monimutkainen tuotemalli saadaan päivitettyä tarvittavilta osin monimutkaisella vastaussanomalla eli kumpaa lähdetään lukemaan ja päivitetään toista vai luetaan toista ja etsitään sopiva kohta sitten toisesta? Ja kuinka tämä voitaisiin edes toteuttaa läheskään kuin aiemmassa versiossa.

Tähän keksittiin opinnäytetyön tutkimuksissa todella tehokas ratkaisu, joka ei hyydy vielä suuremmissakaan tuoterakenteissa.

Kuviossa 4 on kuvattu kokonaisuus palveluiden ja prosessien lähettämien datojen kohteiden välillä. Nuolet kuvaavat datan siirtoja ja vihreällä kuvatut ovat D365:n sisällä olevia prosesseja, joka ovat toteutettu tässä opinnäytetyön osuudessa. Osassa kuvion 4 vihreistä prosesseista oli olemassa osittain tai edes perusteet muokkauksille kuten RunBaceBatch –luokka, joka mahdollistaa erätöiden luomisen kaikista luokista, jotka laitetaan periytymään tästä luokasta. Automaattista konfigurointia, jossa konfigurointi olisi toteutettu ilman käyttäjän manuaalista työtä ei ollut olemassa, joten se täytyi kehittää täysin ja löytää ratkaisu tavalle toimia ja manipuloida standardia prosessia. Kehitetyn automaattisen konfiguroinnin liittäminen erätyöksi ei ollut kuin luokan periyttäminen RunBaseBatch –luokasta ja tarvittavien haluttujen ajoehtoparametrinen Pack ja UnPack –metodien luomista ja välittämistä. (Dynamics 365 2020d.)



KUVIO 4. Integrointitoteutuksen iso kuva

5.1 PCAdaptor –luokka

PCAdaptor –luokkiin liittyvät tämän kehitystyön suurimmat löydökset. Näissä PCAdaptor luokissa tehdään Solverin (D365:n sisäinen regular expression koodit suorittava osa) ehdottamiin attribuutteihin, tuoterakenteeseen ja töiden reitteihin muutoksia. Muutetaan esimerkiksi tuoterakenteen rakennetta, reittien asetus- ja ajoaikojen kestoja.

Tässä koodaus paikassa on mahdollista tehdä koodia PCAdaptor luokkaa extendoimalla (laajentamalla aiempaa ja käyttämällä sitä uudella nimellä) ja se on vielä kytketty nimellä haluttuun tuotemalliin. Ongelmallista tässä olisi rakentaa kaikille lähes 300:lle tuotemallille oma nimetty PCAdaptor –luokka ja toimitta ne Build –putkeen (toimitusputki, jossa versionhallinnan koodimuutokset etenevät kohti PROD ympäristöä) johtuen CIL –käännöksen tarpeesta. Tarvittaisiin paljon koodausta ja asiakkaalle hinta olisi valtava.

5.2 Universal PCAdaptor –luokka

Opinnäytetyön tutkimuksen aikana kasvaneen ymmärryksen ja tehdyn muutoksen myötä tätä luokkaa nyt kutsutaan yritys-x:n ratkaisussa kaikille konfiguroinneille automaattisesti, jolloin muutoksia vaativat tuotemallit kutsuvat määrityksissään triggereiden avulla.

Nyt tehdyn helpottavan muutoksen avulla kaikki tuotemallien nimet johtavat tähän ”Universal” PCAdaptor luokkaan, jolloin se ajetaan kaikkien mallien kanssa ja jollei siellä ajettavalta tuotemallilta löydy näitä triggereitä, niin sitten se ei vaikuta konfiguroinnin tuloksiin mitenkään. Mitään parametrejä ei siis lähetä HMPCE –laskenta moottorille, jollei triggerei-

tä löydy tuotemallilta. Jos tätä muutosta ei oltaisi rakennettu, olisi jokaiselle 300 kpl tuotemalleille jouduttu kopioimaan samantapainen PCAdaptor luokka omalle nimelleen linkitetyksi. Alusta alkaen pyrittiin hyvin yleiseen luokkaan, jossa kaikille tehtäisiin samat prosessiin suunnitellut mahdolliset muutokset. Lisäksi tehdystä lisäyksestä yritys-x:n ei tarvitsisi tilata koodaustyötä jatkuvasti uuden mallin teon yhteydessä, koska siellä on CIL –käännökselle tarve ja vielä olisi tehtävä Build -toimitusputki mukaan lukien validointi Microsoftin puolesta tuotantoympäristöön lisättävästä koodista. Pyrittiin siis todella Universaaliin toteutukseen, jossa kaikki menisi saman kaavan mukaan ja muutokset olisivat yritys-x:n omissa käsissä muuttaessaan D365:n tuotemalleja käyttöliittymästä ja lisäksi omissa käsissään olevassa Azuressa olevassa Java- pohjaisessa HMPCE –laskentamoottorissa ja sen SqlLite –tietokannan datoissa. Kaikki on suunniteltu niin, ettei yritys-x:n tarvitsisi aina tilata koodiuudistuksia, kun päivittävät mallejaan tai paikallista HMPCE -tietokantaa ilman jatkuvia versio päivityksiä ja Build –putkeen uusien mallien lähettämistä aina muutostarpeen noustessa esiin.

5.2.1 Universal PCAdaptor luokan käynnistys ja alustus

Tämä kehitetty yleisluokka kaikille malleille on toisaalta hyvin suoraviivainen mutta useampia alustusmuuttujia tarvitaan monesta syystä. Esimerkiksi, koska D365 on herkkä tilanteissa, jossa kysytään attribuuttien eli yleismuuttujien arvoja ja kysyttävää attribuutti muuttujaa ei olisikaan olemassa niin peruuttamaton Error –virhe lähtee välittömästi ja laajennetussa luokassa se johtaa aina konfigurointi virheeseen tai siis sen perusluokan virheen hallinnan käsittelyyn. Samoin käy myös tuoterakenteen ja reitinvaiheiden kysymisessä eli täytyy hallita mallin sisältöä ja lopultakin tietää minkä muuttujan arvoa kannattaa kysellä. Mallit ovat todella modulaarisia eli sisältävät alimalleja (submodel), jotka ovat osia eli omia tuoterakenteita moniin malleihin. Tämä johtaa siihen, että tässä yleismallisessa tuotemallin konfiguraatiossa useampi attribuutin, tuoterakenteen ja reitin arvojen omistavat avain arvot, jotka omistavat arvonsa on tallennettava alustettaviin pinoihin ja tässä tapauksessa väliaikaisiin containereihin (säiliö), joilta voi tarkistaa uskaltaako kysyä mallin ko. kohdan arvoa, jota oltaisiin muuttamassa, jottei aiheutettaisiin konfiguroinnin peruuttamatonta virhettä. Tässä peruuttamattomassa virheessä on sellainen ongelma, että kun luodaan yleismallista ratkaisua niin järjestelmän täytyy pystyä ilmoittamaan millainen virhe oli ja kun peruuttamaton virhe käynnistyy, niin silloin ylemmän tason ttsAbort –toiminto aktivoituu, jolloin kaikki ko. ttsbegin (tietokannan transaktio) aikana tehdyt muutokset peruuntuvat ja myös virheilmoitukset, jotka tulisivat ns. infoLog –viestistä ulos peruuntuvat myös tämän konfiguroinnin ollessa erätyönä, joka on se tärkein ja viimeisin vaihe sisään tulevissa tarjouksissa eli automaattinen konfigurointi ominaisuus. Opinnäytetyössä täytyi

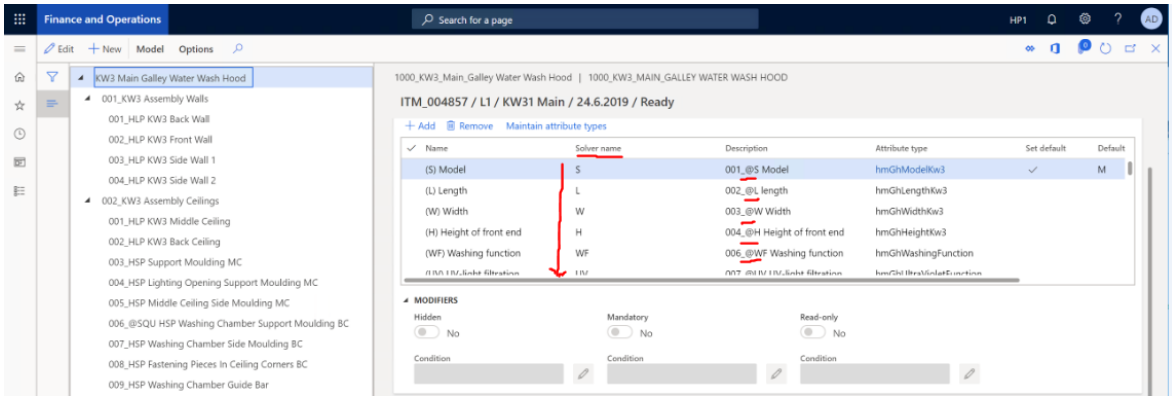
siis keksiä jonkinlainen ratkaisu tallentaa mahdollinen virhelogi talteen vaikka ko. pahin mahdollinen ongelma osuisi kohdalle. Tämänkaltainen ongelma on enemmän kuin odotettu, koska pyritään tekemään koodi yleiseksi ja kaikille sopivaksi eli vain kysyvät data triggereiden avulla lähetetään ja HMPCE vastaus datan tulee osua attribuuttien nimien ja kysyvien isäntä ja ali nimikkeiden kohdalta oikein eli data virheitä on varmasti tulossa. InfoLog –tason lisäominaisuus tehtiin myös helpottamaan testausta ja virheen hallintaa, mutta siitä tarkemmin tulevissa kappaleissa.

Infologin tallennuksessa ratkaisuksi löytyi UserConnect –komento, joka mahdollistaa halutun tietokantaan tehdyn lisäyksen (insert) –komennolla eri tts:än (transactionId:llä), jolloin ttsabort –komento ei peru perustapahtuman TransActionID:n tapahtumia ja logit saadaan talteen. Tätä menetelmää käytin useammassa kohdassa, jossa oli pelko, että pienikin virhe peruu, jopa Infologin talteen kirjoituksen täysin. (Microsoft Dynamics Community 2020a; Microsoft Dynamics Community 2020c.)

Lopulta tätä tarvittiin myös mallin TEST –toiminnossa, koska haluttiin tuotemallin koko rakenne JSON –fomaatissa talteen ja TEST- tilassa kaikki päättyy lopulta ttsAbortiin eli ei kuluteta tietokannan numerosarjoja kuluttamalla niitä vaan ilmoitetaan TEST –tapahtuma virheeksi standardi D365 FO:n toimesta virheeksi. Tämä johti siihen, ettei saatu tarvittua WMDPreConfiguredModelData –xml –tiedostoa talteen, joka oli ratkaisu automaattiseen konfigurointiin ilman käyttäjän käsin valintoja näytöissä vaan käytettiin ATK:ta suorittamaan kaikki nopeasti ennalta annettujen arvojen pohjalta, jotka olivat myyntimiehen tarjousta tehdessä tehtyjä konfiguroinnin valintoja. Tähän palataan myöhemmin.

5.2.2 Etsitään lähetettävät parametrit eli ns. Triggerit

Luetaan käynnistävän mallin tuotemalli kokonaan lävitse ja etsitään niitä triggereitä, jotka tunnistavat arvot joita tuotemalli haluaa lähetettäväksi HMPCE –laskenta moottorille ja odottaa saavansa vastauksen arvoihinsa.



KUVA 10. Lähtevien triggerien esiintyminen parametreillä

Kuvassa 10 on esitetty miten lähtevät attribuutit on määritetty merkittäväksi, jotta ne poimitaan HMPCE:lle lähtevään sanomaan. Attribuutit joiden nimessä esiintyy @ -merkki selitys kentässä niin tämän attribuutin nimi ja konfiguroitu (valittu) arvo lähetetään lähtevässä sanomassa. Esimerkki on kappaleessa 5.2.3.

Action	Tag	Description
Update quotation/sales line	none	Update sales unit price, unit cost estimate, ETO unit cost, unit weight estimate and HOT-code to quotation or sales line[1]
Update sheet metal part data for BOM line	@BSD	Update sheet metal part dimensioning data (quantity, x,y,z,scrap) for a BOM line
Update BOM quantity	@BQU	Update BOM quantity on BOM line
Update subcomponent quantity	@SQU	Update subcomponent quantity
Update route operation data	@RTE	Update route operation's setup and operating times
Update attribute value for boolean switch	@ABS	Update attribute value (TRUE/FALSE) for Boolean switch

TAULUKKO 1. D365 mallin muutosten triggerit

Lisäksi taulukosta 1 puuttuvat poimittavat @RTE_Bending ja @RTE_Cutting –tagit.

Kaikki Tagit käydään lävitse kappaleessa 5.2.6 Päivitetään tuotemallia.

Taulukossa 1 Tag –kentän arvot ovat tapahtumia malleilla, jotka on sovittu oleva triggereinä, jonka pohjalta tuotemallin solmu poimitaan lähtevään sanomaan, jonne halutaan palautusarvo, jonka pohjalta attribuutteja, tuoterakennetta tai reittiä muokataan.

Kutsutaan siis UniversalPcAdaptor luokan CreateRequest –metodia, jossa omien parametrien alustusten jälkeen kutsutaan FindParameters2Be_Sended – metodia, jossa kerätään lähetettävät osat. Laajennettujen asiakkaan tarpeiden myötä tässä toiminnossa kerätään myös tuotemallin koko rakenne alimalleineen samalle lähtevälle sanomalle mukaan BOM JSON (bill of material) lohkoon mukaan. Tästä lisätoiminnosta ja sen tarpeesta on oma kappaleensa myöhemmin.

Lopulta lähettävä metodi saa vastauksen ja palauttaa osat joihin halutaan muuttaa sisältöä tai saada muutettuja arvoja laskentamoottorilta. Siitä omassa kappaleessa 5.3.1 BOM JSON myöhemmin.

5.2.3 Lähetetään parametrit HMPCE –laskenta moottorille

Lähetetään HMPCE –laskukoneelle juuri kerätyt parametrit WebServicen ylitse -JSON muodossa Base64 koodattuna merkkijonona, jotta skandit ja muut erikoismerkit eivät me-

nisi rikki. Nyt voidaan jo huomata, että nuo ovat juuri ne samat parametrit, jotka esiintyivät konfiguroinnin valinta ikkunassa.

Esimerkki sanomasta on kuvassa 11.

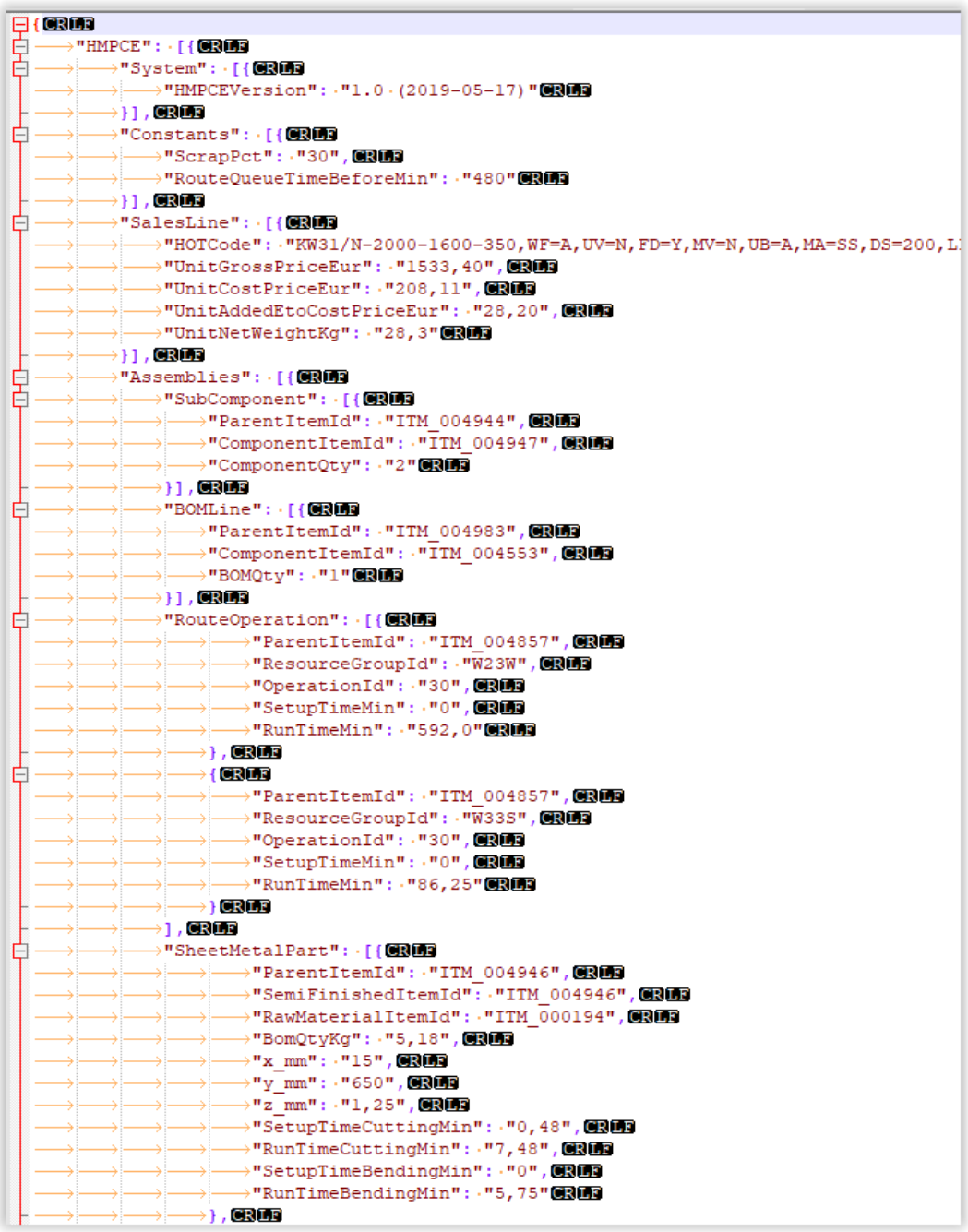

```
JSON output:
{"productname":"KW3 ","attributes":[
{"attribute":"S","order":1,"value":"M - With Capture Jet 3"},
{"attribute":"L","order":2,"value":"1900"},
{"attribute":"W","order":3,"value":"1100"},
{"attribute":"H","order":4,"value":"400"},
{"attribute":"B","order":5,"value":"500"},
{"attribute":"SW","order":6,"value":"2 - Both sides"},
{"attribute":"WF","order":7,"value":"A - CCW-M Cabinet"},
{"attribute":"UV","order":8,"value":"Y - Yes"},
{"attribute":"FD","order":9,"value":"Y - Yes"},
{"attribute":"MV","order":10,"value":"Y - Yes"},
{"attribute":"UB","order":11,"value":"B - UV/Marvel-box Stainless Steel"},
{"attribute":"MA","order":12,"value":"SS - Stainless steel EN1.4301"},
{"attribute":"S1","order":13,"value":"100 mm"},
{"attribute":"Q1","order":14,"value":"1 pc"},
{"attribute":"LF","order":15,"value":"B - Type LED"},
{"attribute":"ETO","order":16,"value":"N - No"},
{"attribute":"AC_L2","order":17,"value":"No"},
{"attribute":"AC_L3","order":18,"value":"No"},
{"attribute":"AC_S1","order":19,"value":"Yes"},
{"attribute":"AC_S2","order":20,"value":"No"},
{"attribute":"AC_BL","order":21,"value":"Yes"},
{"attribute":"AC_HE","order":22,"value":"No"}]
}


Base64:
eyJwcm9kdWN0bmFtZSI6IktXMyAiLCJhdHRyaWJ1dGVzIjpbeyJhdHRyaWJ1dGUiOiJTIiwib3JkZXI
iOjEsInZhbHVlIjoiTSAtIFdpdGggQ2FwdHVyZSBKZXQgMyJ9LHsiYXR0cmlidXRlIjoiTCIsIm9yZG
VyIjoyLCJ2YWx1ZSI6IjE5MDAifSx7ImF0dHJpYnV0ZSI6IlciLCJvcmRlciI6MywidmFsdWUiOiIxM
TAwIn0seyJhdHRyaWJ1dGUiOiJIIiwib3JkZXIiOjQsInZhbHVlIjoiNDAwIn0seyJhdHRyaWJ1dGUi
OiJCIiwib3JkZXIiOjUsInZhbHVlIjoiNTAwIn0seyJhdHRyaWJ1dGUiOiJTVyIsIm9yZGVyIjo2LCJ
2YWx1ZSI6IjIgLSBCb3RoIHNpZGVzIn0seyJhdHRyaWJ1dGUiOiJXRiIsIm9yZGVyIjo3LCJ2YWx1ZS
I6IkEgLSBDQ1ctTSBDYWJpbmV0In0seyJhdHRyaWJ1dGUiOiJVViIsIm9yZGVyIjo4LCJ2YWx1ZSI6I
lkgLSBZZXMifSx7ImF0dHJpYnV0ZSI6IkZEIiwib3JkZXIiOjksInZhbHVlIjoiWSAtIFllcyJ9LHsi
YXR0cmlidXRlIjoiTVYiLCJvcmRlciI6MTAsInZhbHVlIjoiWSAtIFllcyJ9LHsiYXR0cmlidXRlIjo
iVUIiLCJvcmRlciI6MTEsInZhbHVlIjoiQiAtIFVWL01hcnZlbC1ib3ggU3RhaW5sZXNzIFN0ZWVsIn
0seyJhdHRyaWJ1dGUiOiJNQSIsIm9yZGVyIjoxMiwidmFsdWUiOiJTUyAtIFN0YWlubGVzcyBzdGVlb
CBFTjEuNDMwMSJ9LHsiYXR0cmlidXRlIjoiUzEiLCJvcmRlciI6MTMsInZhbHVlIjoiMTAwIG1tIn0s
eyJhdHRyaWJ1dGUiOiJRMSIsIm9yZGVyIjoxNCwidmFsdWUiOiIxIHBjIn0seyJhdHRyaWJ1dGUiOiJ
MRiIsIm9yZGVyIjoxNSwidmFsdWUiOiJCIC0gVHlwZSBMRUQifSx7ImF0dHJpYnV0ZSI6IkVUTyIsIm
9yZGVyIjoxNiwidmFsdWUiOiJOIC0gTm8ifSx7ImF0dHJpYnV0ZSI6IkFDX0wyIiwib3JkZXIiOjE3L
CJ2YWx1ZSI6Ik5vIn0seyJhdHRyaWJ1dGUiOiJBQ19MMyIsIm9yZGVyIjoxOCwidmFsdWUiOiJOby J9
LHsiYXR0cmlidXRlIjoiQUNfUzEiLCJvcmRlciI6MTksInZhbHVlIjoiWWVzIn0seyJhdHRyaWJ1dGU
iOiJBQ19TMiIsIm9yZGVyIjoyMCwidmFsdWUiOiJOby J9LHsiYXR0cmlidXRlIjoiQUNfQkwiLCJvcm
RlciI6MjEsInZhbHVlIjoiWWVzIn0seyJhdHRyaWJ1dGUiOiJBQ19IRSIsIm9yZGVyIjoyMiwidmFsd
WUiOiJOby J9XX0=
```


KUVA 11. D365 lähtevä HMPCE JSON -sanoma

5.2.4 Odotetaan ja luetaan vastaussanoma

Odotetaan vastausta ja palautteena saadaan seuraavanlainen JSON sanoma:



```
{
  "HMPCE": [{
    "System": [{
      "HMPCEVersion": "1.0 (2019-05-17)"
    }],
    "Constants": [{
      "ScrapPct": "30",
      "RouteQueueTimeBeforeMin": "480"
    }],
    "SalesLine": [{
      "HOTCode": "KW31/N-2000-1600-350,WF=A,UV=N,FD=Y,MV=N,UB=A,MA=SS,DS=200,L
      "UnitGrossPriceEur": "1533,40",
      "UnitCostPriceEur": "208,11",
      "UnitAddedEtoCostPriceEur": "28,20",
      "UnitNetWeightKg": "28,3"
    }],
    "Assemblies": [{
      "SubComponent": [{
        "ParentItemId": "ITM_004944",
        "ComponentItemId": "ITM_004947",
        "ComponentQty": "2"
      }],
      "BOMLine": [{
        "ParentItemId": "ITM_004983",
        "ComponentItemId": "ITM_004553",
        "BOMQty": "1"
      }],
      "RouteOperation": [{
        "ParentItemId": "ITM_004857",
        "ResourceGroupId": "W23W",
        "OperationId": "30",
        "SetupTimeMin": "0",
        "RunTimeMin": "592,0"
      },{
        "ParentItemId": "ITM_004857",
        "ResourceGroupId": "W33S",
        "OperationId": "30",
        "SetupTimeMin": "0",
        "RunTimeMin": "86,25"
      }],
      "SheetMetalPart": [{
        "ParentItemId": "ITM_004946",
        "SemiFinishedItemId": "ITM_004946",
        "RawMaterialItemId": "ITM_000194",
        "BomQtyKg": "5,18",
        "x_mm": "15",
        "y_mm": "650",
        "z_mm": "1,25",
        "SetupTimeCuttingMin": "0,48",
        "RunTimeCuttingMin": "7,48",
        "SetupTimeBendingMin": "0",
        "RunTimeBendingMin": "5,75"
      },
```


KUVA 12. D365 HMPCE vastaussanoma

Paluusanoma kuvassa 12 on JSON formaattia, jonka vuoksi, jos olisi yritetty lukea sitä täysin olemassa olevia apuluokkia käyttäen, dataformaatti olisi kovakoodattava, jotta JSON serialisointi ja vasta deSerialisointi olisi toiminut. Aluksi todella yritettiin parsinnassa hyödyntää ensin Swaggler ohjelmistoa, jolla voitiin generoida Swaggler client .dll, joka kytkettiin D365:een resurssina MS Visual Studion yritys-x:n -solution kokonaisuuteen, jota

yritys-x:lle koodataan, jossa on siis mukana kaikki mitä halutaan muokata ei vain Universal PCAdaptor koodi muutokset.

Ongelmaksi muodostui Swaggler .dll:n kutsumisessa se, että D365:n .Net –koodissa kaiken pitää olla kovin staattista ennalta koodattua eli pitää tietää mitä parametria lähdetään kysymään koodista. Tämä onnistui vain testitapauksessa, ja kun ajatellaan laajemmin, niin tarkoituksena oli tehdä koodi, jossa parametrejä ei tiedetä eikä halutakkaan tietää ennalta, jolloin niitä ei voida koodiinkaan kirjoittaa. Tämän pohjalta yritys-x:n itse määrittämät parametrit, joita haluavat tuotemalleilla lähettää, muuttaa ja ottaa vastaan eivät voi olla etukäteen koodattavissa datamalliin, jonka JSON olisi vaatinut. Yritys-x:n tehtävänä olisi vain pitää huoli, että heidän ylläpitämät mallit D365:ssä ja heidän kehittämänsä ns. HMPCE–Java laskukoneohjelmansa ymmärtävät nämä parametrit. Konfiguroinnin on mentävä lävitse ilman, että koodia tarvitsee yhä uudelleen toimittaa toimitusputken lävitse tuotemallien muuttuessa.

Lisäksi Swagger .dll luonnin käyttämisessä ongelmia löytyi, vaikka Swaggler –dll tarjosi hienosti mahdollisuuden lukea koko vastaanotettu sanoma. D365:n puolella pidemmät lauseet, eli tietorakenteet, piti pilkkoa aina omaksi, tyypitetyksi ominaisuuksiksi. Tämä ei käynyt, koska ei haluttu tietää vastaussanomasta ennalta mitään, koska tavoiteltiin ”Universaalia ratkaisu”. Siinä yritys-x saa itse päättää jatkossakin tuotemalliensa rakenteen, jossa vain Trigger -toiminnot ovat kovakoodattuja. Sanoman osat vaihtuvat tunnuksia myöten mallien vaihtuessa kaiken aikaa. Ainoa kokoava asia on, että vastaussanomassa pitää samat kirjain tunnukset kohdentua kuin lähtevässäkin sanomassa eli D365:n puolella ei ole väliä millaisia alitunnuksia käytetään, kunhan vastaussanomassa on samat, jolloin triggerin tyypin mukainen muutos voidaan toteuttaa tietämättä itse sanoman sisältöä. Opinnäytetyössä tehtiin siis täysin ”Universaali” ja tunnoton logiikka, joka toimii kunhan D365:n malli ja HMPCE ovat samalla tasolla. D365:n puolella mallit voidaan rakentaa suoraan käyttöliittymällä ilman, että esimerkiksi ehtoja tai mitään luokkia on olemassa, jolloin ei tarvita koodien validointia toimitusputkessa, eikä muitakaan hidasteita, jotka eivät olisi yritys-x:n käsissä. Lisäksi HMPCE –moottorin Java päivitys on heidän omissa käsissään, hoitavat vain molemmat samaan aikaan ja koko paketti integrointeineen toimii.

5.2.5 Sanoman tallennus tietokantaan

Ehkä yksi suurimmista innovaatioista oli rakentaa riittävän yhdenmukainen tietokannan WMDConfiguredValues -taulu D365:n puolelle, jonne vastaussanoma JSON sanomasta pilkottiin Swagger testin epäonnistumisen jälkeen. Tietokannan käyttö tässä sopi hyvin lopputarkoitukseen, koska JSON oli nopea purkaa yksinkertaisella tavalla ja SQL-tietokanta pystyy helposti saamaan ”select join” -liitoksella kaikki saman JSON ryhmän

datat nippuun, jolloin kyselyn where -lauseen ehdot on usealla SQL-tietokannan rivillä ja parametrit toisilla. Ainoa yhdistävä tekijä on ryhmä eli path –kenttä taulussa, joka on suoraan peräisin JSON vastaussanoman Array –ryhmästä ja parametrit ovat tietenkin sieltä peräisin. Lisäksi kaikki konfiguroinnin parametrit ovat tallessa ja helposti tarkasteltavissa esimerkiksi, kun kehitetään uusia malleja D365:n puolella ja HMPCE:n puolella eli lähetettyjen parametrien tulee tuottaa oikeat vastaus parametrit, jotka vastaavat D365:n mallin tehtyjä komento triggereitä. Voidaan sanoa, että epäonnistuminen Swaggerin kanssa johti parempaan lopputulokseen, koska luodusta taulusta saatiin todella ”Universaali” eli pystyy syömään kaiken ja hakemiskysely (select) lauseet toimivat tehokkaasti sopien indexien avulla vaikka path kysely onkin ns. ”wild card” kysely, joka saattaisi aiheuttaa ”full database scan” kyselyn.

ID	InventTransId	Number	Parameter	Path	TypeIdNbr	Value
HP1-000131	HP1-001051	1	BomQtyKg	Output.SheetMetalParts[1]	0	-0.0202111875
HP1-000131	HP1-001051	1	OperationId	Output.RouteOperations[1]	0	40
HP1-000131	HP1-001051	1	ParentItemId	Output.SheetMetalParts[1]	0	ITM_004944
HP1-000131	HP1-001051	1	RawMaterialItemId	Output.SheetMetalParts[1]	0	ITM_000194
HP1-000131	HP1-001051	1	ResourceGroupId	Output.RouteOperations[1]	0	W23W
HP1-000131	HP1-001051	1	RunpTimeBendingMin	Output.SheetMetalParts[1]	0	11.5
HP1-000131	HP1-001051	1	RunTimeCuttingMin	Output.SheetMetalParts[1]	0	10.5493333333333
HP1-000131	HP1-001051	1	RunTimegMin	Output.RouteOperations[1]	0	632.5
HP1-000131	HP1-001051	1	SemiFinishedItemId	Output.SheetMetalParts[1]	0	ITM_004947
HP1-000131	HP1-001051	1	SetupTimeBendingMin	Output.SheetMetalParts[1]	0	0
HP1-000131	HP1-001051	1	SetupTimeCuttingMin	Output.SheetMetalParts[1]	0	-0.0022493095766129
HP1-000131	HP1-001051	1	SetupTimeMin	Output.RouteOperations[1]	0	0
HP1-000131	HP1-001051	1	x_mm	Output.SheetMetalParts[1]	0	2065
HP1-000131	HP1-001051	1	y_mm	Output.SheetMetalParts[1]	0	-1
HP1-000131	HP1-001051	1	z_mm	Output.SheetMetalParts[1]	0	1.25

KUVA 13. WMDConfiguredValues –taulun esimerkki

JSON-sanoma yksinkertaisesti jaettiin parametri -arvopareihin (kuva 13), ja path-kenttään laitettiin JSON ryhmän polkuun, jolloin kaikki samassa ryhmässä olleet parametrit kuuluivat samaan paikkaan tuotemallilla. Tämä mahdollisti seuraavaksi esitellyt SQL select lauseet ilman, että ”Full Database Scan” toteutuu path parametrin wild card -ominaisuudesta huolimatta.

5.2.6 Tuotemallin päivitys

Malli luetaan uudelleen rekursiivisesti lävitse. Rekursiivinen kutsunta tarkoittaa tilannetta, jossa käynnistynyt luenta metodi kutsuu itseään uudelleen ja uudelleen aina kun uusi alimalli löytyy luettavasta mallista, jolloin päästään helposti kaikkiin alamallin n kpl. alimallin osiin koodaamatta mitään suurta ennalta suunniteltua loop –rakennetta. Ainoa miinus tämänkaltaisessa on se, jos rakenteissa olisi kehäkierto. Kehäkierrossa rakenteen osalla on

aiempi osa aliosanaan eli johtaisi tilanteeseen, jossa alirakenteiden tutkinta ei loppuisi vasta kun kaikki käyttömuisti palvelimesta olisi kulutettu (pinon pullistaja -tilanne).

Päämalli luetaan siis rekursiivisesti lävitse ja etsitään ne triggerit, joihin haluttiin muutosta, jolloin muutoksen triggerin tyypin mukaan valitaan oikea SQL -lause, jolla löydetään WMDConfiguredValues -tallennetusta sanomasta oikeat lohkot. Taulukossa 2 olevat Tagit on sovittu toteuttavaksi.

Action	Tag	Description
Update quotation/sales line	none	Update sales unit price, unit cost estimate, ETO unit cost, unit weight estimate and HOT-code to quotation or sales line[1]
Update sheet metal part data for BOM line	@BSD	Update sheet metal part dimensioning data (quantity, x,y,z,scrap) for a BOM line
Update BOM quantity	@BQU	Update BOM quantity on BOM line
Update subcomponent quantity	@SQU	Update subcomponent quantity
Update route operation data	@RTE	Update route operation's setup and operating times
Update attribute value for boolean switch	@ABS	Update attribute value (TRUE/FALSE) for Boolean switch

TAULUKKO 2. D365 muutettavat triggerit

Taulukossa 2 esiteltyjen triggereiden lisäksi mukaan ovat tulleet:

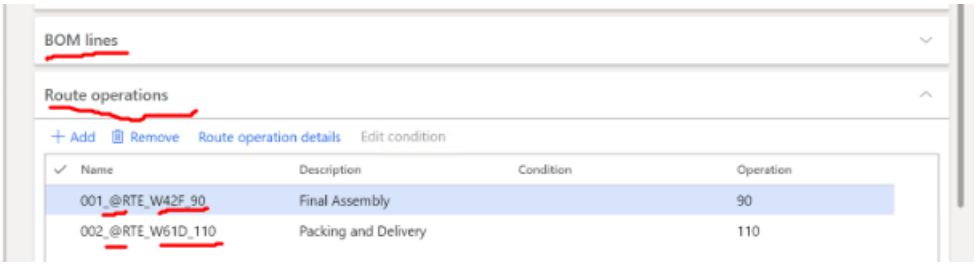
@RTE_Bending

Päivitetään reitintietoja, jotka riippuvat alimallista ja päärakenteesta, jossa tämä taivutus työvaihe on mukana. Täysin omistavan päämallin nimike ja osan nimeke riippuvaisia kyselyitä, joille data haetetaan suuresta määrästä vastaavia SheetMetalParts datoja.

@RTE_Cutting

Päivitetään reitintietoja, jotka riippuvat alimallista ja päärakenteesta, jossa tämä leikkaus työvaihe on mukana. Täysin omistavan päämallin nimeke ja osan nimeke riippuvaisia kyselyitä, joille data haetetaan suuresta määrästä vastaavia SheetMetalParts datoja.

Kuvassa 14 on esimerkki, kuinka Tagit löytyvät D365:n tuotemallin määrityksistä ja nyt on kyse @RTE –tagista, jolla W42F –kuormituspaikan/työpoisteen työvaiheeseen 90 asetetaan HMPCE –laskukoneelta tulleet asetus ja työvaiheen suoritusajat.



KUVA 14. D365 @ RTE –triggerin esimerkki

Rekursiivisessa tuotemallin alimallien ja näiden alimallien läpikäynnissä aina jokainen taso tietää suoraan kuka itse on eli tietää ItemId –koodinsa, koska uudessa kutsussa välitetään edellisen tason kutsuva ItemId parentItemId:n ja itse alamalli tietää suoraan mikä itse on. Kuvasta 15 nähdään alimallin ItemId, joka lähtee rekursiossa mukaan, eli koodi kutsuu itseään uusilla parametreillä kunnes kaikki on käyty lävitse.

KUVA 15. D365 Alimallit

Nyt rekursiossa lähetetään aina alimallia kysyvän ItemId mukaan _parentItemId:n, joka näkyy SQL -kyselyssäkin ja tuo toinen _MeMySelfItemId on tämä alimalli, jossa etsintä on menossa ja sekin taso aina tietää itsensä eli ItemId:n. Näin helposti voitiin löytää suuresta määrästä ryhmiä juuri oikean tuotemallin alimallin parametrit. Kuvassa 13 on värikynällä merkattu samaan ryhmään path arvon omaavat rivit ja SQL –lauseita tehneet näkevät alempana kuvasta 18 alkaen kuinka triggereitä haetaan select –lauseilla ja miten nämä _parentItemId ja _MyMySelfItemId sijoitetaan where –ehtoihin. Vertaamalla ylempää kuvaa 13, jossa data tulee ja se pystytään kohdistamaan oikea JSON –lohkoon, josta vain sitten löydettyä triggeriä palvellaan ja asetetaan halutut arvot tulleesta datasta paikalleen.

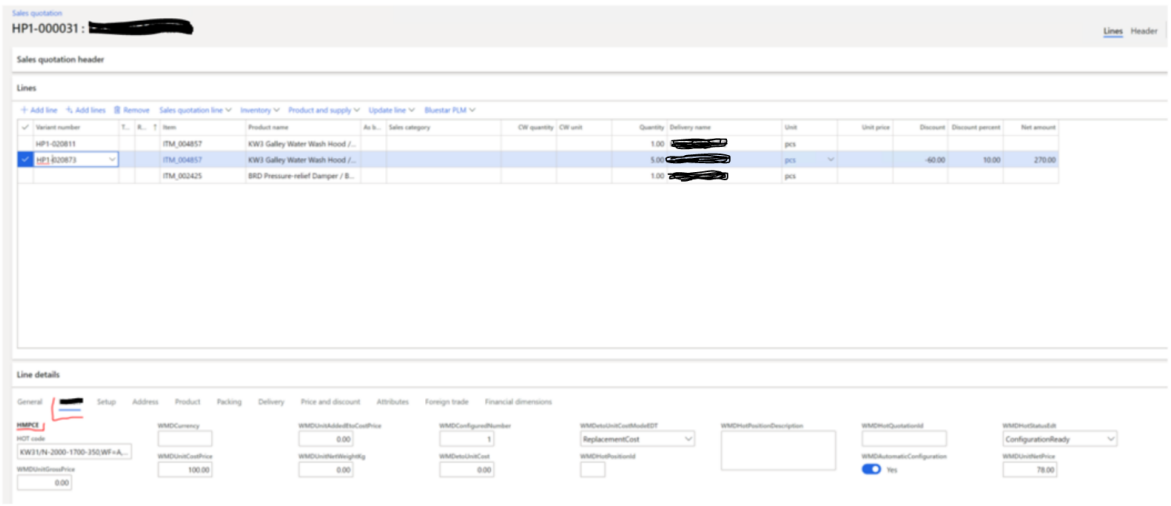
1. Yleinen osa

   Tässä näytetään kaikki parametrit, jotka halutaan siirtää konfiguroinnin aloittaneelle modulille: Myyntitilauksen tai myyntitarjouksen riveille. Nämä arvot löytyvät HMPCE JSON vastaussanomasta ja tallennetaan tauluun JSON SalesLine –lohkosta kuten muutkin parametrit (kuva 16).

```
"Output":{"BooleanSwitches":{
"BooleanValue":"TRUE",
"AttrSolverName":"SW_01",
"ParentItemId":"ITM_004857"},
"SalesLine":{
"HOTCode":"KW31/M - With Capture Jet 3-1900-1100-400-500",
"UnitGrossPrice":12581.38,
"Currency":"EUR",
"UnitCostPrice":1801.72,
"UnitAddedEtoCostPrice":0,
"UnitNetWeightKg":36.91},
"Constants":{
"RouteQueueTimeBeforeMin":8,
```


KUVA 16. D365 HMPCE Myyntitilauksen palautusarvot

Konfiguroinnin UniversalPCAdaptor –osuuden lopuksi ne tallennetaan kutsuvan taulun riveille omiin kenttiinsä (kuva 17).



KUVA 17. D365 Myyntitilauksen ja tarjouksen uudet kentät

2. @BSD –triggeri:

Tuotemallin rakenteen osan dimensio mittatietojen päivitys triggeri. Tämä pohjautuu SheetMetalsParts –osan JSON palautus datasta haettavaan tietoon ParentItemId:n (malli itse) ja SemiFinishedItemId:n parametrin arvolla _MeMySelfItemId:llä (osan ItemId). Arvoilla etsin datasta Path -kentän SheetMetalsParts ryhmää, jossa ehdot kohdistuvat. Löydetystä ryhmästä saadaan osan X,Y ja Z –arvojen mittatiedot paikalleen tulleiden parametrien nimillä samannimisille attribuuteille, jotka on kytketty osan dimensioihin.

```
} else if (strScan(adaptorBOMLine.getName(),"@BSD",1,strLen(adaptorBOMLine.getName()))){
        info(strFmt("BOM: %1 ItemId: %2",adaptorBOMLine.getName(),""));//adaptorBOMLine.parmItemNumber()));
    //or: %6",adaptorBOMLine.includeInGeneration(),adaptorBOMLine.parmCondition(),adaptorBOMLine.parmTurnoverQuantity(),
      adaptorBOMLine.getName(), PCTemplateComponent.Name,adaptorBOMLine.parmColor())); // attribute.getValueAsLocalizedString()));
    select * from _wmdcv where _wmdcv.InventTransId == m_inventTransId
                            && _wmdcv.ID == m_id
                            && _wmdcv.Number == m_number
                            && _wmdcv.Path like "*SheetMetalParts*"
                            && _wmdcv.Parameter == "ParentItemId"
                            && _wmdcv.Value == _parentItemId

            join _wmdcv1 where _wmdcv1.Path == _wmdcv.Path
                            && _wmdcv1.ID == m_id
                            && _wmdcv1.InventTransId == _wmdcv.InventTransId
                            && _wmdcv1.Number == _wmdcv.Number
                            && _wmdcv1.Parameter == "SemiFinishedItemId"
                            && _wmdcv1.Value == _MeMySelfItemId;

    while select * from _wmdcv2 where _wmdcv2.Path == _wmdcv.Path
                        && _wmdcv2.ID == m_id
                        && _wmdcv2.InventTransId == _wmdcv.InventTransId
                        && _wmdcv2.Number == _wmdcv.Number{
            if (conFind(meAttributes_Con, _wmdcv2.Parameter) != 0){
                subattribute_parametri1 = component.getAttribute(_wmdcv2.Parameter);
                subattribute_parametri1.assignValue(_wmdcv2.Value);
                //info(strFmt("-->%1 ja value: %2",_wmdcv2.Parameter, _wmdcv2.Value));
            } else {
                info(strFmt("-->Ei löytynyt"));
                info(strFmt("-->%1 ja value: %2",_wmdcv2.Parameter, _wmdcv2.Value));
            }
    }
```

KUVA 18. D365 @BSD triggerin SQL-kysely

Kuvassa 13 on kuvassa värikynällä merkattu samaan ryhmään path arvon omaavat rivit ja SQL –lauseita tehneet näkevät select kuviosta ylempänä miten kohdistus tapahtuu. Nyt kun oikea lohko on löytynyt, lopuksi ajetaan vain kaikki parametrit kannasta ulos ja jos ne löytyvät mallista niin arvot voidaan asettaa parametrille.

3. @BQU

Tuotemallin rakenteen osan määrän päivitys triggeri. Tämä pohjautuu SheetMetalsParts –osan JSON palautus datasta haettavaan tietoon ParentItemId:n (malli itse) ja ComponentItemId:n parametrin arvolla _MeMySelfItemId:llä (osan ItemId). Arvoilla etsin datasta Path kentän SheetMetalsParts ryhmää, jossa ehdot kohdistuvat. Löydetystä ryhmästä saadaan osan BOMQty eli osan määrä. Katso kuva 19.

```
while (meAttributesBOM.moveNext())
{
    adaptorBOMLine = meAttributesBOM.currentValue();
    if (strScan(adaptorBOMLine.getName(),"@BQU",1,strLen(adaptorBOMLine.getName()))){
        //info(strFmt("BOM: %1 ItemId: %2",adaptorBOMLine.getName(),"")); //adaptorBOMLine.parmItemNumber()));
      //or: %6",adaptorBOMLine.includeInGeneration(),adaptorBOMLine.parmCondition(),adaptorBOMLine.parmTurnoverQuantity(), adaptorBOMLine.getName(), PC
      select * from _wmdcv where _wmdcv.InventTransId == m_inventTransId
                              && _wmdcv.Number == m_number
                              && _wmdcv.Path like "*BOMLine*"
                              && _wmdcv.Parameter == "ParentItemId"
                              && _wmdcv.Value == _parentItemId
            join _wmdcv1 where _wmdcv1.Path == _wmdcv.Path
                              && _wmdcv1.InventTransId == _wmdcv.InventTransId
                              && _wmdcv1.Number == _wmdcv.Number
                              && _wmdcv1.Parameter == "ComponentItemId"
                              && _wmdcv1.Value == _MeMySelfItemId
            join _wmdcv2 where _wmdcv2.Path == _wmdcv.Path
                              && _wmdcv2.InventTransId == _wmdcv.InventTransId
                              && _wmdcv2.Number == _wmdcv.Number
                              && _wmdcv2.Parameter == "BOMQty";
      if (_wmdcv.RecId)
      {
          adaptorBOMLine.parmTurnoverQuantity(str2Num(_wmdcv2.Value));
          //info(strFmt("-->%1 ja value: %2",_wmdcv.RecId, _wmdcv2.Value));
      }
      else {
          info(strFmt("@BQU-->Ei löytynyt -->BOM: (%1) ParentItemId: %2 MeMySelfItemId; %3",adaptorBOMLine.getName(),_parentItemId,_MeMySelfItemId));//
          //info(strFmt("BOM: %1 ItemId: %2",adaptorBOMLine.getName(),"")); //adaptorBOMLine.parmItemNumber()));
      }
```

KUVA 19. D365 @BQU triggerin SQL-kysely

4. @SQU

Alamallin määrän muunto on toistaiseksi suunniteltu tarpeettomaksi triggeriksi, sen vuoksi koodi on UniversalPCAdaptor –luokassa kommentoituna. Tarkoitus oli muuttaa ylämallilla tarvittavan alamallin määrää. osan määrä. Katso kuva 20.

```
else
{
    i = strScan(attribute.getName(),"@SQU",1,strLen(attribute.getName()));
    if (i)
    {
        /*
        //sijainti = str2Int(subStr(_translation_T.Description,1,i));
        //m_collectedParameters = conPoke(m_collectedParameters,sijainti,attribute.getName()); //meAttributes.currentKey());
        //m_collectedValues = conPoke(m_collectedValues,sijainti,attribute.getValueAsLocalizedString());
        select * from _wmdcv where _wmdcv.InventTransId == m_inventTransId
                            && _wmdcv.Number == m_number
                            && _wmdcv.Path like "*SubComponents"
                            && _wmdcv.Parameter == "ParentItemId"
                            && _wmdcv.Value == attribute.//adaptorRouteOperation.parmRouteGroup()
        join _wmdcv1 where _wmdcv1.Path == _wmdcv.Path
                        && _wmdcv1.InventTransId == _wmdcv.InventTransId
                        && _wmdcv1.Number == _wmdcv.Number
                        && _wmdcv1.Parameter == "ComponentItemId"
                        && _wmdcv.Value == operNum
        join _wmdcv2 where _wmdcv2.Path == _wmdcv.Path
                        && _wmdcv2.InventTransId == _wmdcv.InventTransId
                        && _wmdcv2.Number == _wmdcv.Number
                        && _wmdcv2.Parameter == "SetupTimeCuttingMin";
        if (_wmdcv.RecId)
        {
            attribute.assignValue(_wmdcv.Value);                            //.parmTurnoverQuantity(str2Num(_wmdcv1.Value));
            info(strFmt("-->%1 ja value: %2",_wmdcv.RecId, _wmdcv.Value));
        }
        else info(strFmt("-->Ei löytynyt"));

        info(strFmt("Attribute: (%1) Attribute %2: Value: %3 Description: %4",sijainti, meAttributes.currentKey(), (attribute.getValueAsLocalizedStr
        */
    }
```

KUVA 20. D365 @SQU triggerin SQL-kysely

5. @RTE_Bending

Taivutustyövaiheen reitin tietojen päivitys triggeri. Tämä pohjautuu SheetMetalsParts –osan JSON palautus datasta haettavaan tietoon ParentItemId:n (malli itse) ja Semi-

FinishedItemId (osan ItemId) arvoilla. Löydetään datasta Path kentän SheetMetalsParts ryhmää, jossa ehdot kohdistuvat ja saadaan tietyn osan taivutukselle oikeat reitin ajat, koska taivutus riippuu kohteesta. Arvot asetetaan tulleen datan samannimisille attribuuteille, jotka on kohdistettu oikeisiin reitin parametreihin (SetupTime & RunTime). Katso kuva 21.

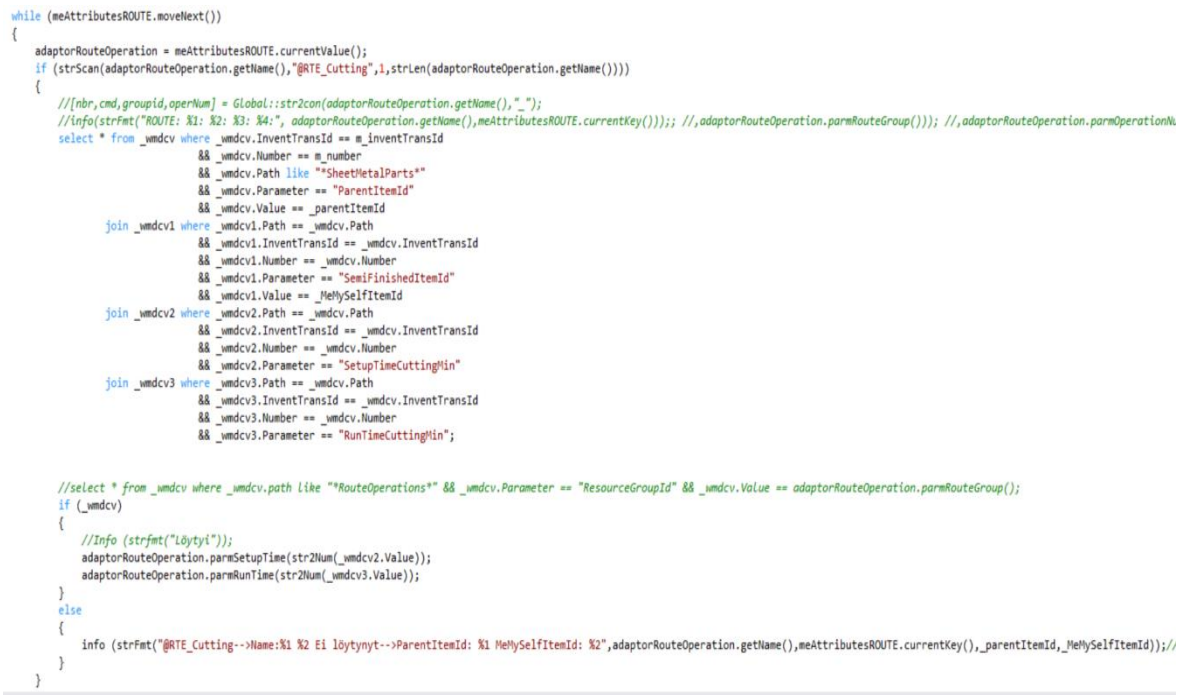
```
else if (strScan(adaptorRouteOperation.getName(),"@RTE_Bending",1,strLen(adaptorRouteOperation.getName())))
{
    [nbr,cmd,groupid,operNum] = Global::str2con(adaptorRouteOperation.getName(),"_");
    //info(strFmt("ROUTE: %1: %2: %3: %4:", adaptorRouteOperation.getName(),meAttributesROUTE.currentKey()));; //,adaptorRouteOperation.parmRouteGroup())); //,adaptorRouteOperation.parmOperationN
    select * from _wmdcv where _wmdcv.InventTransId == m_inventTransId
                          && _wmdcv.Number == m_number
                          && _wmdcv.Path like "*SheetMetalParts*"
                          && _wmdcv.Parameter == "ParentItemId"
                          && _wmdcv.Value == _parentItemId
          join _wmdcv1 where _wmdcv1.Path == _wmdcv.Path
                          && _wmdcv1.InventTransId == _wmdcv.InventTransId
                          && _wmdcv1.Number == _wmdcv.Number
                          && _wmdcv1.Parameter == "SemiFinishedItemId"
                          && _wmdcv1.Value == _MeMySelfItemId
          join _wmdcv2 where _wmdcv2.Path == _wmdcv.Path
                          && _wmdcv2.InventTransId == _wmdcv.InventTransId
                          && _wmdcv2.Number == _wmdcv.Number
                          && _wmdcv2.Parameter == "SetupTimeBendingMin"
          join _wmdcv3 where _wmdcv3.Path == _wmdcv.Path
                          && _wmdcv3.InventTransId == _wmdcv.InventTransId
                          && _wmdcv3.Number == _wmdcv.Number
                          && _wmdcv3.Parameter == "RunTimeBendingMin";


    //select * from _wmdcv where _wmdcv.path like "*RouteOperations*" && _wmdcv.Parameter == "ResourceGroupId" && _wmdcv.Value == adaptorRouteOperation.parmRouteGroup();
    if (_wmdcv)
    {
        //Info (strfmt("Löytyi"));
        adaptorRouteOperation.parmSetupTime(str2Num(_wmdcv2.Value));
        adaptorRouteOperation.parmRunTime(str2Num(_wmdcv3.Value));
    }
    else
    {
        info (strFmt("@RTE_Bending-->Name:%1 %2 Ei löytynyt-->ParentItemId: %1 MeMySelfItemId; %2",adaptorRouteOperation.getName(),meAttributesROUTE.currentKey(),_parentItemId,_MeMySelfItemId));/
    }
}
```

KUVA 21. D365 @RTE_Bending triggerin SQL-kysely

6. @RTE_Cutting

Leikkaustyövaiheen reitin tietojen päivitys triggeri. Tämä pohjautuu SheetMetalsParts –osan JSON palautus datasta haettavaan tietoon ParentItemId:n (malli itse) ja SemiFinishedItemId (osan ItemId) arvoilla. Löydetään datasta Path kentän SheetMetalsParts ryhmää, jossa ehdot kohdistuvat ja saadaan tietyn osan taivutukselle oikeat reitin ajat, koska taivutus riippuu kohteesta. Arvot asetetaan tulleen datan samannimisille attribuuteille, jotka on kohdistettu oikeisiin reitin parametreihin. (SetupTime & RunTime). Katso kuva 22.



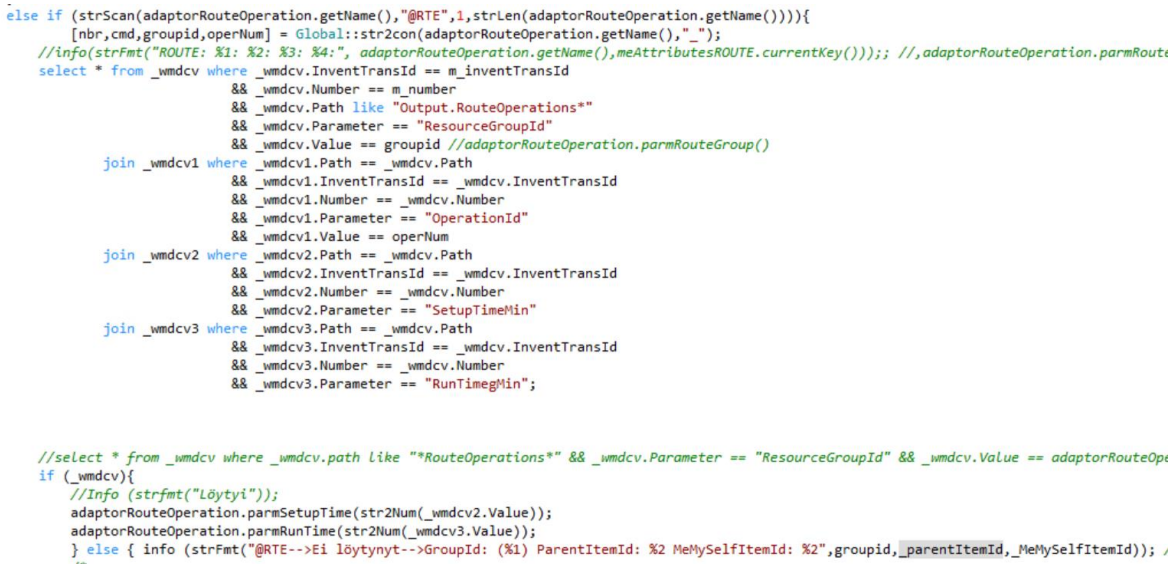
```
while (meAttributesROUTE.moveNext())
{
    adaptorRouteOperation = meAttributesROUTE.currentValue();
    if (strScan(adaptorRouteOperation.getName(),"@RTE_Cutting",1,strLen(adaptorRouteOperation.getName())))
    {
        //[nbr,cmd,groupid,operNum] = Global::str2con(adaptorRouteOperation.getName(),"_");
        //info(strFmt("ROUTE: %1: %2: %3: %4:", adaptorRouteOperation.getName(),meAttributesROUTE.currentKey()));; //,adaptorRouteOperation.parmRouteGroup())); //,adaptorRouteOperation.parmOperationN
        select * from _wmdcv where _wmdcv.InventTransId == m_inventTransId
                                && _wmdcv.Number == m_number
                                && _wmdcv.Path like "*SheetMetalParts*"
                                && _wmdcv.Parameter == "ParentItemId"
                                && _wmdcv.Value == _parentItemId
                join _wmdcv1 where _wmdcv1.Path == _wmdcv.Path
                                && _wmdcv1.InventTransId == _wmdcv.InventTransId
                                && _wmdcv1.Number == _wmdcv.Number
                                && _wmdcv1.Parameter == "SemiFinishedItemId"
                                && _wmdcv1.Value == _MeMySelfItemId
                join _wmdcv2 where _wmdcv2.Path == _wmdcv.Path
                                && _wmdcv2.InventTransId == _wmdcv.InventTransId
                                && _wmdcv2.Number == _wmdcv.Number
                                && _wmdcv2.Parameter == "SetupTimeCuttingMin"
                join _wmdcv3 where _wmdcv3.Path == _wmdcv.Path
                                && _wmdcv3.InventTransId == _wmdcv.InventTransId
                                && _wmdcv3.Number == _wmdcv.Number
                                && _wmdcv3.Parameter == "RunTimeCuttingMin";


        //select * from _wmdcv where _wmdcv.path like "*RouteOperations*" && _wmdcv.Parameter == "ResourceGroupId" && _wmdcv.Value == adaptorRouteOperation.parmRouteGroup();
        if (_wmdcv)
        {
            //Info (strfmt("Löytyi"));
            adaptorRouteOperation.parmSetupTime(str2Num(_wmdcv2.Value));
            adaptorRouteOperation.parmRunTime(str2Num(_wmdcv3.Value));
        }
        else
        {
            info (strFmt("@RTE_Cutting-->Name:%1 %2 Ei löytynyt-->ParentItemId: %1 MeMySelfItemId: %2",adaptorRouteOperation.getName(),meAttributesROUTE.currentKey(),_parentItemId,_MeMySelfItemId));//
        }
    }
```

KUVA 22. D365 @RTE_Cutting triggerin SQL-kysely

7. @RTE

Perusreittivaiheen tietojen haku triggeri. Tämä pohjautuu RouteOperations dataan, josta haetaan työvaiheen resourcegroupId:llä (kuormituspaikan ryhätunnuksella) ja työvaihenumerolla oikeat ajat reitille (SetupTime & RunTime). Ei ole nimikkeestä riippuvainen kuten SheetMetalsParts –osat. Katso kuva 23.



```
else if (strScan(adaptorRouteOperation.getName(),"@RTE",1,strLen(adaptorRouteOperation.getName()))){
        [nbr,cmd,groupid,operNum] = Global::str2con(adaptorRouteOperation.getName(),"_");
    //info(strFmt("ROUTE: %1: %2: %3: %4:", adaptorRouteOperation.getName(),meAttributesROUTE.currentKey()));; //,adaptorRouteOperation.parmRoute
    select * from _wmdcv where _wmdcv.InventTransId == m_inventTransId
                            && _wmdcv.Number == m_number
                            && _wmdcv.Path like "Output.RouteOperations*"
                            && _wmdcv.Parameter == "ResourceGroupId"
                            && _wmdcv.Value == groupid //adaptorRouteOperation.parmRouteGroup()
            join _wmdcv1 where _wmdcv1.Path == _wmdcv.Path
                            && _wmdcv1.InventTransId == _wmdcv.InventTransId
                            && _wmdcv1.Number == _wmdcv.Number
                            && _wmdcv1.Parameter == "OperationId"
                            && _wmdcv1.Value == operNum
            join _wmdcv2 where _wmdcv2.Path == _wmdcv.Path
                            && _wmdcv2.InventTransId == _wmdcv.InventTransId
                            && _wmdcv2.Number == _wmdcv.Number
                            && _wmdcv2.Parameter == "SetupTimeMin"
            join _wmdcv3 where _wmdcv3.Path == _wmdcv.Path
                            && _wmdcv3.InventTransId == _wmdcv.InventTransId
                            && _wmdcv3.Number == _wmdcv.Number
                            && _wmdcv3.Parameter == "RunTimegMin";


    //select * from _wmdcv where _wmdcv.path like "*RouteOperations*" && _wmdcv.Parameter == "ResourceGroupId" && _wmdcv.Value == adaptorRouteOpe
    if (_wmdcv){
        //Info (strfmt("Löytyi"));
        adaptorRouteOperation.parmSetupTime(str2Num(_wmdcv2.Value));
        adaptorRouteOperation.parmRunTime(str2Num(_wmdcv3.Value));
        } else { info (strFmt("@RTE-->Ei löytynyt-->GroupId: (%1) ParentItemId: %2 MeMySelfItemId: %2",groupid,_parentItemId,_MeMySelfItemId)); /
```

KUVA 23. D365 @RTE triggerin SQL-kysely

8. @ABS

Alun perin ajatus oli muuttaa konfiguroitutivia attributteja eli perusparametreja. Tämäkin on myöhemmin tunnustettu tarpeettomaksi ominaisuudeksi, koska kaikkia tarpeellisia ominaisuuksia pystytään muokkaamaan riittävästi suoraan ja Solver ei ota enää PCAdaptor –vaiheessa muutoksia huomioon. Katso kuva 24.

```
while (meAttributes.moveNext())
{
    i = 0;
    attribute = meAttributes.currentValue();
    meAttributes_Con_solvername += attribute.getName(); //Testi
    _attribute_T = attribute.getAttribute();
    _translation_T = EcoResAttributeTranslation::findByAttributeAndLanguage(_attribute_T.RecId, LanguageTable::defaultLanguage());
    //_attribute_T.getFieldValue(
    //info(strFmt("-Attribute: (%1) Attribute %2: Value: %3 Description: %4",sijainti, meAttributes.currentKey(), (attribute.getValueAsLocalizedString() == "" ?
    if (_translation_T.Description != ""){
        meAttributes_Con += _translation_T.Description; //TESTI
        meAttributes_Con_recid += _attribute_T.RecId; //test
    }
    i = strScan(_attribute_T.Name,"@ABS",1,strLen((_attribute_T.Name)));
    if (i)
    {
        //sijainti = str2Int(subStr(_translation_T.Description,1,i));
        //m_collectedParameters = conPoke(m_collectedParameters,sijainti,attribute.getName()); //meAttributes.currentKey());
        //m_collectedValues = conPoke(m_collectedValues,sijainti,attribute.getValueAsLocalizedString());
        select * from _wmdcv where _wmdcv.InventTransId == m_inventTransId
                                && _wmdcv.Number == m_number
                                && _wmdcv.Path == "Output.BooleanSwitches"
                                && _wmdcv.Parameter == "ParentItemId"
                                && _wmdcv.Value == _parentItemId
                join _wmdcv1 where _wmdcv1.Path == _wmdcv.Path
                                && _wmdcv1.InventTransId == _wmdcv.InventTransId
                                && _wmdcv1.Number == _wmdcv.Number
                                && _wmdcv1.Parameter == "AttrSolverName"
                                && _wmdcv1.Value == attribute.getName()
                join _wmdcv2 where _wmdcv2.Path == _wmdcv.Path
                                && _wmdcv2.InventTransId == _wmdcv.InventTransId
                                && _wmdcv2.Number == _wmdcv.Number
                                && _wmdcv2.Parameter == "BooleanValue";
        if (_wmdcv.RecId)
        {
            attribute.assignValue(_wmdcv2.Value);                               //.parmTurnoverQuantity(str2Num(_wmdcv1.Value));
            //info(strFmt("-->%1 ja value: %2",_wmdcv.RecId, _wmdcv.Value));
        }
        else
        {
            info(strFmt("@ABS-->Ei löytynyt -->ParentItemId: (%1) Attribute %2: Value: %3 Description: %4",_parentItemId, meAttributes.currentKey(), (attribute.
            //info(strFmt("Attribute: (%1) Attribute %2: Value: %3 Description: %4",sijainti, meAttributes.currentKey(), (attribute.getValueAsLocalizedString()
        }
    }
```

KUVA 24. D365 @ABS triggerin SQL-kysely

5.3 Tuotemallin testaus

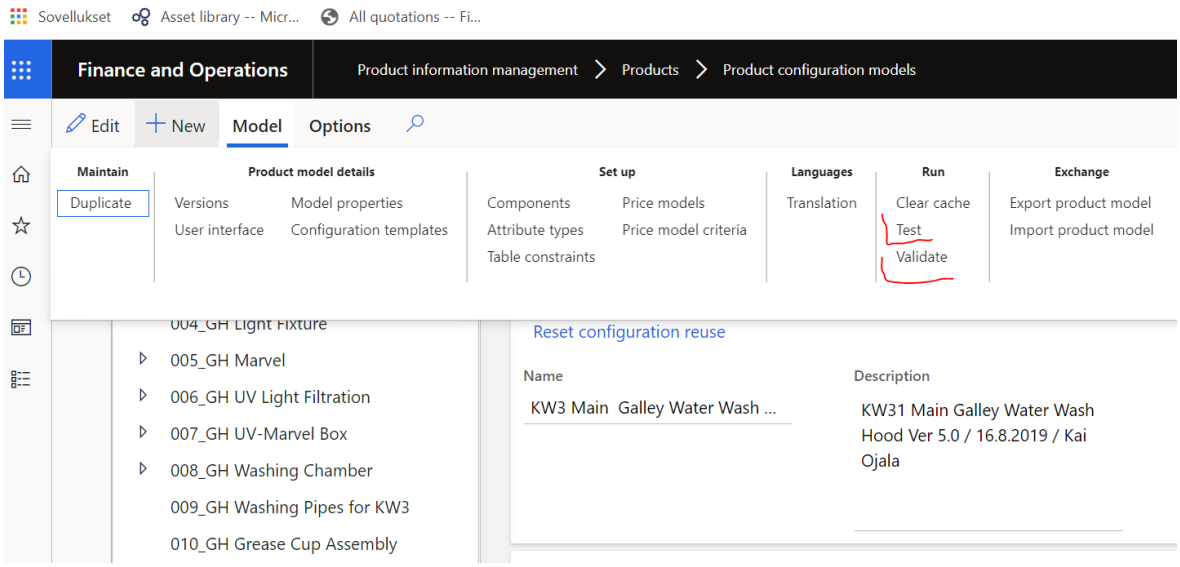
Tuotemallien testaus on tärkeää, jotta voidaan varmistaa, että D365:n tuotemallin lähtevät triggerit ja HMPCE –laskukoneen vastaus triggerit vastaavat toisiaan, koska vastaus sanoman tulisi vastata tuotemallilla esitettyihin datan vastaanotto pyyntöihin. Universaalissa integraatio toteutuksessa data kulkee lävitse ilman, että koodi varsinaisesti tietää mitä välittää eli ei ole business logiikkaa datan suhteen.

UniversalPCAdaptor –laajennus on tehty niin universaaliksi kuin mahdollista, jolloin ns. kovakoodausta eli tarkkoja määritettyjä tunnuksia muuttujilla on pyritty välttämään, jotta prosessiin saadaan lisättyä parametreja tuotemalliin ja vastaussanomaan suoraan ilman lisäkoodaustarvetta. Laajennus lähettää vain päämallilla sovitusti @ -merkillä nimetyt parametrit lähtevään sanomaan ymmärtämättä niitä ilman @ -merkkiä ja vastaukset tulevat

sitten sovituilla trigger –toiminto nimillä pitkin mallia ja sen alamalleja. Vastauksiin hyväksyttiin erilaisia kovakoodattuja triggereitä, koska data haetaan erilaisilla SQL –lauseilla triggeristä riippuen, mutta kuitenkin niin, että vain tämän kaltaisia asioita halutaan HMPCE –laskukoneen kautta muuttaa.

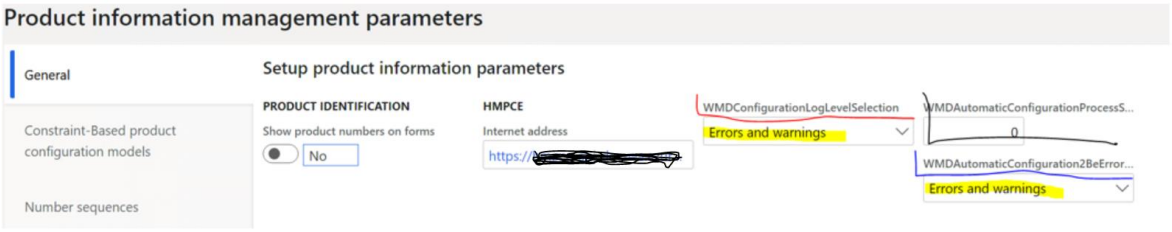
Kuvassa 26 on asetus HMPCE – Internet Address, jolla voitiin Test –ympäristössä kohdistaa HMPCE –laskukoneelle lähtevät viestit Test –version HMPCE:lle, koska mallit ja vastaussanomat on testattava ennen PROD (tuotannon) julkaisua. Prod ympäristössä osoitteena on PROD ympäristön testattu HMPCE mallidatoineen.

Kuvassa 25 on asetus myös testaus ja tarkastus toiminnoille ko. mallille. Tällä kuvan Testaus –napilla käynnistetään normaali testaus prosessi, jossa ajetaan UniversalPCAdaptor laajennus, jossa sanomat lähetetään HMPCE –laskukoneelle ja otetaan vastaussanoma vastaan. Samassa Testi –toiminnossa lähetyksen yhteydessä on mukana sekä seuraavan kappaleen 5.3.1 BOM JSON –sanoman lähetys ja liitetiedostoksi luonti, että automaattisen konfiguroinnin tarvitseman esitallennetun XML –tiedoston luonti, josta on tarkempi kuvaus kappaleessa 5.3.2 Esitallennettu konfiguroinnin XML. (Microsoft Dynamics Community 2020b.)



KUVA 25. D365 Mallin Test ja tarkastus toiminto

Testaukseen liittyy myös opinnäytetyössä tehdyt valinnat syntyville logiviesteille. Logiviestien tarkkuus valitaan seuraavasti:



KUVA 26. D365 Logiviestien, HMPCE ja automaattisen konfiguroinnin asetukset

Punaisella viivalla on kuvassa 26 alla olevassa vetovalikossa valitaan lokin tarkkuus. Jos valitaan kaikki ilmoitukset, niin järjestelmä kertoo kaikenlaista lisätietoa konfiguroidusta rivistä, jolloin päästään helpommin kiinni, parametrien nimeämiseen ja malliin josta on kyse (kuva 27)

Message details

Requirement calculation for Quotation: QT00000012, ITM004857

1000_KW3_Main_Galley Water Wash Hood

Sent Base64 to Hmpce: ey-Jwcm9kdWN0bmFtZSI6IktXMzEiLCJhdHRyaWJ1dGVzIjpbeyJhdHRyaWJ1dGUiOiJTIiwib3JkZXIiOjEsInZhbHVlIjoiTSJ9LHsiYXR0cmlidXRlIjoiTCIsIm9yZGVyIjoyLCJ2YWx1ZSI6IjExMDAifSx7ImF0dHJpYnV0ZSI6IlciLCJvcmRlciI6MywidmFsdWUiOiIxMTAwIn0seyJhdHRyaWJ1dGUiOi-JIliwib3JkZXIiOjQsInZhbHVlIjoiMzUwIn0seyJhdHRyaWJ1dGUiOiJCIiwib3JkZXIiOjUsInZhbHVlIjoiNTAwIn0seyJhdHRyaWJ1dGUiOiJTVyIsIm9yZGVyIjo2LCJ2YWx1ZSI6IjAifSx7ImF0dHJpYnV0ZSI6IldGIiwib3JkZXIiOjcsInZhbHVlIjoiTiJ9LHsiYXR0cmlidXRlIjoiVVYiLCJvcmRlciI6OCwidmFsdWUiOi-JOIn0seyJhdHRyaWJ1dGUiOiJGRCIsIm9yZGVyIjo5LCJ2YWx1ZSI6Ik4ifSx7ImF0dHJpYnV0ZSI6IlVCIiwib3JkZXIiOjEwLCJ2YWx1ZSI6Ik4ifSx7ImF0dHJpYnV0ZSI6IlC1iwib3JkZXIiOjExLCJ2YWx1ZSI6Ik1BIiwib3JkZXIiOjEyLCJ2YWx1ZSI6IlNTIn0seyJhdHRyaWJ1dGUiOiJTMSIsIm9yZGVyIjoxMywidmFsdWUiOiJYIn0seyJhdHRyaWJ1dGUiOiJRMSIsIm9yZGVyIjoxNCwidmFsdWUiOiJCIn0seyJhdHRyaWJ1dGUiOiJMRiIsIm9yZGVyIjoxNSwidmFsdWUiOiJCIn0seyJhdHRyaWJ1dGUiOiJFVE8iLCJvcmRlciI6MTYsInZhbHVlIjoiTiJ9LHsiYXR0cmlidXRlIjoiQUNfTDIiLCJvcmRlciI6MTcsInZhbHVlIjoiTm8ifSx7ImF0dHJpYnV0ZSI6IkFDX0JMIiwib3JkZXIiOjE4LCJ2YWx1ZSI6Ik5vIn0sey-JhdHRyaWJ1dGUiOiJBQ19TMSIsIm9yZGVyIjoxOSwidmFsdWUiOiJObyJ9LHsiYXR0cmlidXRlIjoiQUNfUzIiLCJvcmRlciI6MjAsInZhbHVlIjoiTm8ifSx7ImF0dHJpYnV0ZSI6IkFDX0JMIiwib3JkZXIiOjIxLCJ2YWx1ZSI6Ik5vIn0seyJhdHRyaWJ1dGUiOiJBQ19IRSIsIm9yZGVyIjoyMiwidmFsdWUiOiJObyJ9LHsiYXR0cmlidXRlIjoiQUNfUzIiLCJvcmRlciI6MjMsInZhbHVlIjoiTm8ifV0seyJhdHRyaWJ1dGUiOiJVc2VyIiwib3JkZXIiOjAsInZhbHVlIjoiIn1d

KW31/M-1100-1100-350-500,WF=N,UV=N,FD=N,MV=N,UB=N,MA=SS,S1=X,Q1=1,LF=B,ETO=N

KW3 Main Galley Water Wash Hood

Start--> ParentItemId: (ITM004857) MeMyselfItemId: ITM004857

@ABS--> Not found, ParentItemId: (ITM004857) Attribute SW_01: Value: No Description: 303 Test Switch

@RTE--> Value is empty, GroupId: (W42F) OperNum: 90 solvername--> SetupTimeMin ja value: 0 Route: (001_@RTE_W42F_90) ParentItemId: ITM004857 MeMySelfItemId: ITM004857 path: Output.RouteOperations[4]

Sheet Metal Part Model 01

Start--> ParentItemId: (ITM004945) MeMyselfItemId: ITM008597

@BSD--> Value is empty, SolverName or name--> RawMaterialItemId and value: "" BOM: (001_@BSD) ParentItemId: ITM004945 MeMySelfItemId: ITM008597 Path: Output.SheetMetalParts[15]

@BSD--> Value is empty, SolverName or name--> SetupTimeBendingMin and value: "0" BOM: (001_@BSD) ParentItemId: ITM004945 MeMySelfItemId: ITM008597 Path: Output.SheetMetalParts[15]

@RTE_Bending--> Value is empty, solvername--> SetupTimeBendingMin and value: "0" Route: (002_@RTE_Bending) ParentItemId: ITM004945 MeMySelfItemId: ITM008597 Path: Output.SheetMetalParts[15]

KUVA 27. D365 Logiviestin leikkaus

Ja loppuosassa on aina ajoajat ja jos loki tarkkuus on All niin myös lähetetty sanoma ja vastaus sanoma tulevat lisätietoineen InfoLogille näkyviin (kuva 28). Lokisanomasta leikattiin 14 sivua pois.

JSON output: {"product-
name":"KW31","attributes":[{"attribute":"S","order":1,"value":"M"},{"attribute":"L","order":2,"value":"1100"},{"attribute":"W","order":3,"value":"1100"},{"attribute":"H","order":4,"value":"350"},{"attribute":"B","order":5,"value":"500"},{"attribute":"SW","order":6,"value":"0"},{"attrib-
ute":"WF","order":7,"value":"N"},{"attribute":"UV","order":8,"value":"N"},{"attribute":"FD","order":9,"value":"N"},{"attribute":"MV","order":10,"value":"N"},{"attribute":"UB","order":11,"value":"N"},{"attribute":"MA","order":12,"value":"SS"},{"attribute":"S1","order":13,"value":"X"},{"attribute":"Q1","order":14,"value":"1"},{"attribute":"LF","order":15,"value":"B"},{"attribute":"ETO","order":16,"value":"N"},{"attribute":"AC_L2","order":17,"value":"No"},{"attribute":"AC_L3","order":18,"value":"No"},{"attribute":"AC_S1","order":19,"value":"No"},{"attribute":"AC_S2","order":20,"value":"No"},{"attribute":"AC_BL","order":21,"value":"No"},{"attribute":"AC_HE","order":22,"value":"No"},{"attribute":"AC_RK","order":23,"value":"No"}],"bom":[]}
{"In-
put":{"ConfigurationString":"eyJwcm9kdWN0bmFtZSI6IktXMzEiLCJhdHRyaWJ1dGVzIjpbeyJhdHRyaWJ1dGUiOiJTIiwib3JkZXIiOjEsInZhbHVlIjoiTSJ9LHsiYXR0cmlidXRlIjoiTCIsIm9yZGVyIjoyLCJ2YWx1ZSI6IjExMDAifSx7ImF0dHJpYnV0ZSI6IlciLCJvcmRlciI6MywidmFsdWUiOiIxMTA-
wIn0seyJhdHRyaWJ1dGUiOiJIIiwib3JkZXIiOjQsInZhbHVlIjoiMzUwIn0seyJhdHRyaWJ1dGUiOiJCIiwib3JkZXIiOjUsInZhbHVlIjoiNTAwIn0sey-
JhdHRyaWJ1dGUiOiJTVyIsIm9yZGVyIjo2LCJ2YWx1ZSI6IjAifSx7ImF0dHJpYnV0ZSI6IldGIiwib3JkZXIiOjcsInZhbHVlIjoiTiJ9LHsiYXR0cmlidXRlI-
joiVVYiLCJvcmRlciI6OCwidmFsdWUiOiJOIn0seyJhdHRyaWJ1dGUiOiJGRCIsIm9yZGVyIjo5LCJ2YWx1ZSI6Ik4ifSx7ImF0dHJpYnV0ZSI6Ik1WIiwib3JkZXIiOjEwLCJ2YWx1ZSI6Ik4ifSx7ImF0dHJpYnV0ZSI6IlVCIiwib3JkZXIiOjExLCJ2YWx1ZSI6Ik4ifSx7ImF0dHJpYnV0ZSI6Ik1BIiwib3JkZXIi-
OjEyLCJ2YWx1ZSI6IlNTIn0seyJhdHRyaWJ1dGUiOiJTMSIsIm9yZGVyIjoxMywidmFsdWUiOiJYIn0seyJhdHRyaWJ1dGUiOiJRMSIsIm9yZGVyIjoxNCwidmFsdWUiOiIx-
In0seyJhdHRyaWJ1dGUiOiJMRiIsIm9yZGVyIjoxNSwidmFsdWUiOiJCIn0seyJhdHRyaWJ1dGUiOiJFVE8iLCJvcmRlciI6MTYsInZhbHVlIjoiTiJ9LHsiYXR0cmlidXRlIjoiQUNfTDIiLCJvcmRlciI6MTcsInZhbHVlIjoiTm8ifSx7ImF0dHJpYnV0ZSI6IkFDX0wzIiwib3JkZXIiOjE4LCJ2YWx1ZSI6Ik5vIn0seyJhdHRyaWJ1dGUiOiJBQ19TMSIsIm9yZGVyIjoxOSwidmFsdWUiOiJObyJ9LHsiYXR0cmlidXRlIjoiQUNfUzIiLCJvcmRlciI6MjAsInZhbHVlIjoiTm8ifSx7ImF0dHJpYnV0ZSI6IkFDX0JMIiwib3JkZXIiOjIxLCJ2YWx1ZSI6Ik5vIn0seyJhdHRyaWJ1dGUiOiJBQ19IRSIsIm9yZGVyIjoyMiwidmFsdWUiOi-
JObyJ9LHsiYXR0cmlidXRlIjoiQUNfUksiLCJvcmRlciI6MjMsInZhbHVlIjoiTm8ifV0sImJvbSI6W119"},"Output":{"BOMLines":null,"InternalComponents":null,"BOMStructure":null,"SalesLine":{"HOTCode":"KW31/M-1100-1100-350-
500,WF=N,UV=N,FD=N,MV=N,UB=N,MA=SS,S1=X,Q1=1,LF=B,ETO=N","UnitGrossPrice":37648.83,"Currency":"EUR","UnitCostPrice":7192.47,"UnitAddedEtoCostPrice":0,"UnitNetWeightKg":0},"SubComponents":null,"Constants":{"Scrap":30,"RouteQueueTimeBeforeMin":8},"SheetMetalParts":[{"x_mm":1115,"RawMaterialItemId":"ITM001990","RunTimeBendingMin":5.75,"SetupTimeCuttingMin":0.79,"RunTimeCutting-
Min":7.38,"SemiFinishedItemId":"ITM004946","SetupTimeBendingMin":0,"z_mm":1,"y_mm":650.0,"ParentItemId":"ITM004944","BomQtyKg":7.093},{"x_mm":515,"RawMaterialItemId":"ITM001990","RunTimeBendingMin":5.75,"SetupTimeCuttingMin":0.62,"RunTimeCuttin

gMin":6.75,"SemiFinishedItemId":"ITM004947","SetupTimeBendingMin":0,"z_mm":1,"y_mm":1100.0,"ParentItemId":"ITM004944","Bom
QtyKg":5.545},{"x_mm":1100,"RawMaterialItemId":"ITM001990","RunTimeBendingMin":5.75,"SetupTimeCuttingMin":0.72,"RunTimeCut
ting-
Min":7.13,"SemiFinishedItemId":"ITM004948","SetupTimeBendingMin":0,"z_mm":1,"y_mm":605.0,"ParentItemId":"ITM004944","BomQt
yKg":6.514{"x_mm":870,"RawMaterialItemId":"","RunTimeBendingMin":5.75,"SetupTimeCuttingMin":0.07,"RunTimeCuttingMin":1.32,"S
emiFinishedItem-
Id":"ITM008636","SetupTimeBendingMin":0,"z_mm":1,"y_mm":75.0,"ParentItemId":"ITM009368","BomQtyKg":0.511},{"x_mm":631,"Raw
Mate-
rialItemId":"","RunTimeBendingMin":5.75,"SetupTimeCuttingMin":0.14,"RunTimeCuttingMin":1.16,"SemiFinishedItemId":"ITM008637",
"SetupTimeBending-
Min":0,"z_mm":1,"y_mm":200.0,"ParentItemId":"ITM009368","BomQtyKg":0.988}],"RouteOperations":[{"ResourceGroupId":"W23W","Se
tup-
TimeMin":0,"OperationId":30,"RunTimeMin":310.5},{"ResourceGroupId":"W23W","SetupTimeMin":0,"OperationId":40,"RunTimeMin":6
32.5},{"ResourceGroupId":"W33S","SetupTimeMin":0,"OperationId":80,"RunTimeMin":86.25},{"ResourceGroupId":"W42F","SetupTimeM
in":0,"OperationId":80,"RunTimeMin":86.25},{"ResourceGroupId":"W42F","SetupTimeMin":0,"OperationId":90,"RunTimeMin":689.33}],"
System":{"HMPCE version":"1.0.1 (2020-11-03)","DB version":"Sqlite 1.0.0 (2020-11-03)"}}}
Hmpce send 613 lines data (ID: QT00000012, LotId: I000003679, Nbr: 3)
Configuration done - Output processtime: 2 - Total processtime: 3

KUVA 28. D365 Logiviestin leikkaus

Kuvassa 28, lopussa olevalla toiseksi viimeisellä rivillä olevasta tiedosta nähdään, että HMPCE palautti 613 riviä avain-arvo -paria ja viimeisellä lokin rivillä olevista lokin aikaleimoista voidaan huomata, että triggereiden etsiminen, lähetys, HMPCE -laskenta ja vastauksen saaminen otti 2 sekuntia. Vastaus sanoman purkaminen tietokantaan ja arvojen purkaminen kohteisiin vei 1 sekunnin, joka on kokonaisajassa loppuosa.

Aiemmin kuvassa 26 nähty sininen valinta valitsee Automaattisen konfiguroinnin virheen aiheuttavan InfoLog tason eli voidaan valita Warning & Error –taso, josta vasta hermostuu ja kääntää automaattisen konfiguroinnin rivin menneen virheeseen tai sitten jokin toinen taso, esimerkiksi riippuen testauksen tasosta, jota halutaan simuloida. Oletus tuotanto käytössä olisi juuri Warning & Errors, jonka pohjalta tuotanto käytön virheet on suunniteltu lähetettäväksi kaikki muut ovat vain InfoLog ja Debug tasoista lisätietoa.

Seuraavissa kappaleissa kuvataan tarkemmin testaustoiminnon kautta tapahtuvia prosesseja.

5.3.1 BOM JSON

Tämän ominaisuuden tarve pohjautuu siihen, että tuoterakenteet ovat erilaisia ja osassa on paljon ostettavia kalliita sähkötoimilaitteita ja jos koko tarjous laskettaisiin sovitulla peruskateprosentilla, niin silloin nämä huomattavan kalliit osat nostaisivat tuotteen kokonaishintaa merkittävästi, jos kate on kaikilla osilla sama. Tämän tarpeen vuoksi haluttiin

D365:n mallin tuoterakenne HMPCE –laskukoneen tietoon, jolloin voidaan hankintahinnaltaan kalliiden tuotteiden kateprosentti laskea yksittäisten tuotteiden osalta alemmaksi. Tällä toiminnolla halutaan varmistaa tarjouksien kilpailukykyisyys. Vaikka BOM –lohko lähetetään HMPCE:lle niin sitä ei voida hyödyntää, koska HMPCE:n DBLite –tietokanta on readOnly –tilassa Azuressa. Tämän vuoksi BOM JSON –tiedosto tallennetaan Test- napin käynnistyksen yhteydessä tuotemallille liitetiedostoksi, josta HMPCE:n ylläpitäjä voi poimia tiedoston ja ladata sen HMPCE:n puolelle TEST –ympäristössä. (Microsoft Dynamics Community 2020b.)

Viimeisin lisäys tähän tuoterakenne sanomaan on mahdollinen Default_ItemId arvon lisäys JSON –sanomaan. Tämä haluttiin, koska se helpottaisi huomaamaan tilanteita, joissa HMPCE:n päässä tarvitaan nimikkeitä paluusanomalle, koska D365:n tuotemallilla osaluettelossa kysyttiin component item Id:tä tai RawMaterialItemID:tä eli ko. komponentti ratkaistaan HMPCE:n päässä.

Esimerkkisanoma nähtävillä kuvassa 29:

{"product-
name":"KW31","attributes":[{"attribute":"S","order":1,"value":"M"},{"attribute":"L","order":2,"value":"1100"},{"attribute":"W","order":3,"value":"1100"},{"attribute":"H","order":4,"value":"350"},{"attribute":"B","order":5,"value":"500"},{"attribute":"SW","order":6,"value":"0"},{"attribute":"WF","order":7,"value":"N"},{"attribute":"UV","order":8,"value":"N"},{"attribute":"FD","order":9,"value":"N"},{"attribute":"MV","order":10,"value":"N"},{"attribute":"UB","order":11,"value":"N"},{"attribute":"MA","order":12,"value":"SS"},{"attribute":"S1","order":13,"value":"X"},{"attribute":"Q1","order":14,"value":"1"},{"attribute":"LF","order":15,"value":"B"},{"attribute":"ETO","order":16,"value":"N"},{"attribute":"AC_L2","order":17,"value":"No"},{"attribute":"AC_L3","order":18,"value":"No"},{"attribute":"AC_S1","order":19,"value":"No"},{"attribute":"AC_S2","order":20,"value":"No"},{"attribute":"AC_BL","order":21,"value":"No"},{"attribute":"AC_HE","order":22,"value":"No"},{"attribute":"AC_RK","order":23,"value":"No"}],**"bom":[{"order":1,"bom":{"parentitemid":"ITM004857","bom":[{"order":1,"type":"product","itemid":"ITM004984","default_itemid":"","name":"GHAccessories","qty":0.0},{"order":2,"type":"product","itemid":"ITM004983","default_itemid":"","name":"GHAssemblySupplyAirChamberKW3","qty":1.000000000000000},{"order":3,"type":"product","itemid":"ITM009386","default_itemid":"","name":"GHComponentsKW3","qty":1.000000000000000},{"order":4,"type":"product","itemid":"ITM007953","default_itemid":"","name":"GHGreaseCupAssembly","qty":1.0},{"order":5,"type":"product","itemid":"ITM004975","default_itemid":"","name":"GHLightFixture","qty":1.000000000000000},{"order":6,"type":"product","itemid":"ITM004976","default_itemid":"","name":"GHMarvel","qty":0.0},{"order":7,"type":"product","itemid":"ITM004977","default_itemid":"","name":"GHUVLightFiltration","qty":0.0},{"order":8,"type":"product","itemid":"ITM005093","default_itemid":"","name":"GHUVMarvelBox","qty":0.0},{"order":9,"type":"product","itemid":"ITM004978","default_itemid":"","name":"GHWashingChamber","qty":0.0},{"order":10,"type":"product","itemid":"ITM009380","default_itemid":"","name":"GHWashingPipesforKW3","qty":0.0},{"order":11,"type":"product","itemid":"ITM004945","default_itemid":"","name":"KW3AssemblyCeilings","qty":1.000000000000000},{"order":12,"type":"product","itemid":"ITM004944","default_itemid":"","name":"KW3AssemblyWalls","qty":1.000000000000000}]}},

KUVA 29. D365 BOM JSON esimerkki

Tästä välistä on leikattu 4 sivua tuoterakennetta pois, jotta ajatus säilyy, eli oder 13-58 leikattiin (kuva 30).

{"oder":59,"bom":{"parentitemid":"ITM004948","bom":[{"order":1,"type":"item","itemid":"RawMaterialItemId","default_itemid":"","name":"001_@BSD","qty":1.00}]}},{"order":60,"bom":{"parentitemid":"ITM004949","bom":[{"order":1,"type":"item","itemid":"RawMaterialItemId","default_itemid":"","name":"001_@BSD","qty":1.00}]}},{"order":61,"bom":{"parentitemid":"ITM030106","bom":[{"order":1,"type":"item","itemid":"RawMaterialItemId","default_itemid":"","name":"001_@BSD","qty":1.00}]}}]}

KUVA 30. D365 BOM JSON pienempi esimerkki

Tämän tuoterakenteen pohjalta HMPCE –laskentamoottori saa selville päämallin koko tuoterakenteen. Jos tyyppi on ItemId ja ItemId kentässä on arvo ja Default_ItemId kentässä on arvo, niin sitten nimikettä on tarkoitus vaihtaa. Jos arvoa ei löydy, niin sitten menee default_ItemId arvolla. Jos ItemId on, mutta ei Default_ItemId arvoa, niin sitten mitään ei vaihdeta.

Toinen vaihto ehto on, että tuoterakeenteen ItemId:n tyyppi on product, jolloin tällä arvolla löytyy uusi tuoterakenne BOM JSON sanomasta, jossa product tyyppinen ItemId on itse parentItemId osassa ja sillä on ItemId osansa tai product tyyppisillä taas omat alimallinsa omissa JSON osissaan.

5.3.2 Esitallennettu konfiguroinnin XML

Tämäkin oli yksi ns. ”Show Stopper”, jollei opinnäytetyössä oltaisi pystytty ratkaisemaan, kuinka jo aiemmin HOT –Clientin puolella konfiguroidut tarjoukset voisi tuoda D365 FO:n puolelle ilman käyttäjän tekemää konfigurointia eli tuotteen valittujen ominaisuuksien uudelleen konfigurointia.

Tämä esitallennettu konfiguroinnin XML tallennetaan myöskin päämallin TEST –toiminnon yhteydessä uuteen WMDPreConfiguredModelValues –tauluun.

Esimerkki esitallennetusta konfiguroinnin XML –tiedostosta, josta leikattiin 20 sivua pois (kuva 31):

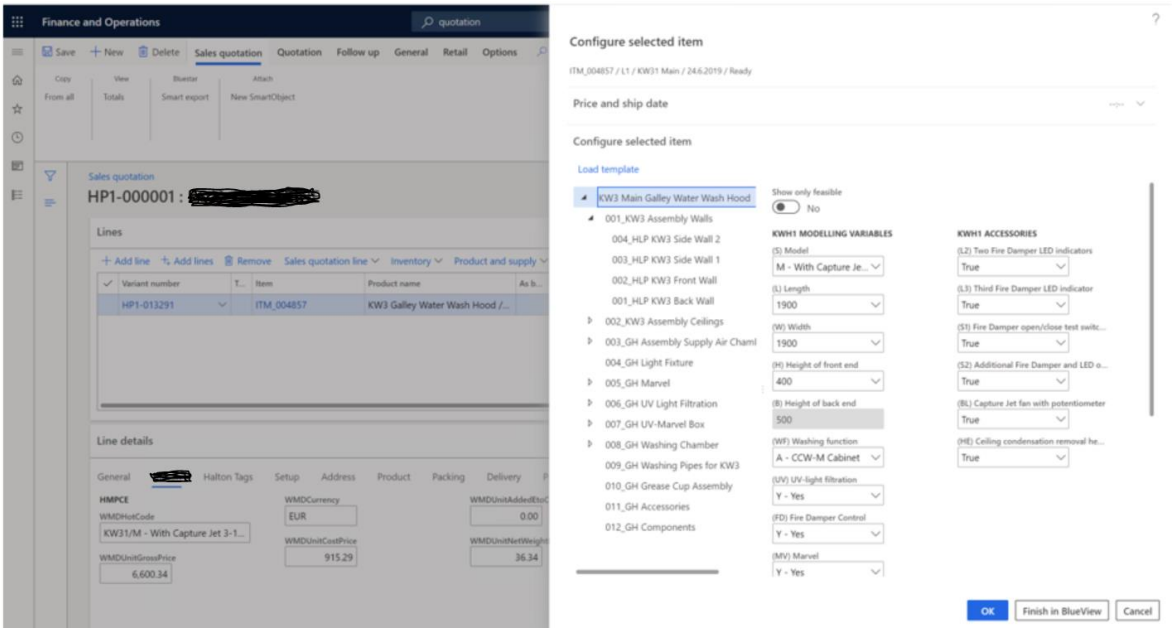
<Session><Component name="KW3 Main Galley Water Wash Hood" uniqueId=""><Attribute name="P" uniqueId="5637149993" isUserSelected="1" type="enum" value="KW31" /><Attribute name="BomQty_SC_005" uniqueId="5637149986" isUserSelected="1" type="decimal" value="0" /><Attribute name="BomQty_SC_006" uniqueId="5637149987" isUserSelected="1" type="decimal" value="0" /><Attribute name="BomQty_SC_007" uniqueId="5637149988" isUserSelected="1" type="decimal" value="0" /><Attribute name="BomQty_SC_008" uniqueId="5637149989" isUserSelected="1" type="decimal" value="0" /><Attribute name="BomQty_SC_009" uniqueId="5637149990" isUserSelected="1" type="decimal" value="0" /><Attribute

name="BomQty_SC_010" uniqueId="5637149991" isUserSelected="1" type="decimal" value="1" /><Attribute name="BomQty_SC_011" uniqueId="5637149992" isUserSelected="1" type="decimal" value="0" /><Attribute name="SW_01" uniqueId="5637149985" isUserSelected="1" type="boolean" value="false" /><Attribute name="OperTimeFinal" uniqueId="5637149983" isUserSelected="1" type="decimal" value="0.10" /><Attribute name="OperTimeDelivery" uniqueId="5637149984" isUserSelected="1" type="decimal" value="0.10" /><Attribute name="**S**" uniqueId="5637149965" isUserSelected="1" type="enum" value="**M**" /><Attribute name="**L**" uniqueId="5637149966" isUserSelected="1" type="integer" value="**1100**" /><Attribute name="**W**" uniqueId="5637149967" isUserSelected="1" type="integer" value="**1100**" /><Attribute name="**H**" uniqueId="5637149968" isUserSelected="1" type="integer" value="350" /><Attribute name="**B**" uniqueId="5637149994" isUserSelected="1" type="integer" value="**500**" /><Attribute name="**SW**" uniqueId="5637149997" isUserSelected="1" type="enum" value="**0**" /><Attribute name="**WF**" uniqueId="5637149969" isUserSelected="1" type="enum" value="**N**" /><Attribute name="**UV**" uniqueId="5637149970" isUserSelected="1" type="enum" value="**N**" /><Attribute name="**FD**" uniqueId="5637149995" isUserSelected="1" type="enum" value="**N**" /><Attribute name="**MV**" uniqueId="5637149971" isUserSelected="1" type="enum" value="**N**" /><Attribute name="**UB**" uniqueId="5637149972" isUserSelected="1" type="enum" value="**N**" /><Attribute name="**MA**" uniqueId="5637149973" isUserSelected="1" type="enum" value="**SS**" /><Attribute name="**S1**" uniqueId="5637149974" isUserSelected="1" type="enum" value="**X**" /><Attribute name="**Q1**" uniqueId="5637149996" isUserSelected="1" type="enum" value="**1**" /><Attribute name="**LF**" uniqueId="5637149975" isUserSelected="1" type="enum" value="**B**" /><Attribute name="**ETO**" uniqueId="5637149976" isUserSelected="1" type="enum" value="**N**" /><Attribute name="**AC_L2**" uniqueId="5637149978" isUserSelected="1" type="boolean" value="**false**" /><Attribute name="**AC_L3**" uniqueId="5637149977" isUserSelected="1" type="boolean" value="**false**" /><Attribute name="**AC_S1**" uniqueId="5637149980" isUserSelected="1" type="boolean" value="**false**" /><Attribute name="**AC_S2**" uniqueId="5637149979" isUserSelected="1" type="boolean" value="**false**" /><Attribute name="**AC_BL**" uniqueId="5637149981" isUserSelected="1" type="boolean" value="**false**" /><Attribute name="**AC_HE**" uniqueId="5637149982" isUserSelected="1" type="boolean" value="**false**" /><Attribute name="**AC_RK**" uniqueId="5637149998" isUserSelected="1" type="boolean" value="**false**" />

KUVA 31. D365 Esitallennettu konfigurointi XML esimerkki

Esimerkki tiedoston leikkaukseen jätettiin vain alkuosa, jossa päämallin HOTCode –arvot sijaitsevat. Parametrit ja arvot ovat kuvassa 31 vahvistettuna (bold). Juuri nämä parametrit ovat niitä, joita konfiguroinnin ensimmäisellä sivulla kysytään ja tämän XML -tiedoston muokkaukseen perustuu koko toiminnallisuus, kuinka voidaan muokata valintoja ohjelmallisesti ilman käyttäjälle aukeavaa valintalistaa.

Standardi version konfigurointi D365:n puolella vaatii käyttöliittymän kautta syötettäviä parametrejä ja se olisi taas ns. ”Show Stopper”, jollei pystyttäisi ratkaisemaan kuinka konfigurointi voitaisiin toteuttaa ilman käyttäjän valintoja tarjouksen sisään tuonnissa, koska jos konfigurointi on tehty jo tarjouksen tekovaiheessa HOT –clientin puolella, niin tuntuisi turhalta siirtää integraation yli tarjous D365:n puolelle, mutta esimerkiksi 100 –rivisen tarjouksen kohdalla pelkästään käsin tehtävään uudelleen konfigurointiin kuluisi valtavasti henkilötyöaikaa, koska kaikki valinnat myytäville tuotteille on jo tehty HOT –clientin puolella. Lisäksi ongelmaksi syntyisi, että mistä saataisiin helposti lista konfigurointia täyttävälle henkilölle valituista valinnoista. Kaikki konfiguroinnin valinnat on tehty jo kertaalleen myyntihenkilön toimesta.



KUVA 32. D365 Konfiguroinnin korvattavat parametrit XML -sanomasta

Yllä olevassa kuvassa 32 on osa päämallin valinta parametreistä ja loput ovat vierityspalkin takana. Jos näitä valinta listoja pitäisi tehdä 100 kertaa ison tarjouksen yhteydessä, ja ko. valinnat olisi jo tehty ja tallennettu HOTCode –tarjouksen kenttään merkkijonoksi. Automaattisella konfiguroinnilla vältetään myös turhaatyötä.

Tämä oli todellinen Show Stopper, kuten koko kehitystyön alussa ollut konfiguroinnin UniversalPCAdaptor –laajennuskin. Ilman näihin löydettyjä ratkaisuja arvoja ei olisi ohjelmallisesti voinut asettaa. Tästä lisää seuraavassa kappaleessa.

5.4 Automaattinen konfigurointi

Automaattinen konfigurointi tarkoittaa toimintoa, jossa tarjoukset on konfiguroitu jo HOT-Clientin puolella myyntihenkilön tekemänä esimerkiksi jossain telakalla (offline –tilassa). Lopulta myyntihenkilö on tulostanut paikanpäällä tarjouksen (offLine) ja sitten päässyt hotelliin, josta lähettää vahvistetun tarjouksen eteenpäin HOTApille (online –tilassa), josta tarjous siirtyy D365:n puolelle OData / REST rajapinnan kautta QuotationTableEntiteetin kautta edelleen QuotationTableStatingTablen –kautta oikeaan QuotationTable & QuotationLineTable –riveiksi D365:n puolelle, jossa kaikki etenee kun tämäkin tarjousriveineen tulee asetuksella WMDAutomaticConfiguration = ON on päällä ja WMDConfigurationStatus on käännetty ReadyForConfiguration asentoon. (Dynamics 365 2020a.)

Jotta ajastetut erätyöt, ts. AutomaticConfiguration batch job –prosessit, voisivat toimia ja mennä lävitse jokaiselle myytävälle tuotemallille, on oltava olemassa WMDPreConfigu-

redModelValue –taulussa XML –sanoma, joka on luotu kuten kuvattu kappaleessa 5.3.2 Esitallennettu konfiguroinnin XML.

Siinä kuvatussa tuotemallin Test- toiminnossa, jossa ko. XML –sanoma luodaan ja tallennetaan samalla kuin aiemmin esitelty BOM JSON –sanoma, tallennetaan liitetiedostoksi mallille. Tämä WMDPreConfiguredModelValue -XML-tiedosto tallennetaan siis omaan tauluun juuri tätä automaatista konfigurointi prosessia varten. Kaikille myytäville tuotteille täytyy olla olemassa tämä esikonfiguroitu –XML tiedosto. Lisäksi on muistettava että, jos alamalleja muutetaan, niin myytävälle päätason tuotemallille on luotava esitallennettu konfiguroinnin XML –tiedosto aina uudelleen, jotta alimallien muutokset tulevat mukaan päämallin esitallennettuun XML –tiedostoon. (Dynamics 365 2020b; Dynamics 365. 2020c; Microsoft Dynamics Community 2020d.)

5.4.1 Automaattisen konfiguroinnin toiminta

Automaattinen erätyö on laitettu käyntiin ja käynnistyy 2 minuutin syklillä.

Prosessi etenee niin että kun automaattinen prosessointi riville ja otsikolla tunnistetaan kentästä WMDAutomaticConfiguration = ON ja otsikolla ja rivillä WMDConfigurationStatus = ReadyForConfiguration niin otetaan tarjous ja rivit käsittelyyn. Kun prosessi käynnistyy, niin konfiguroinnin ajossa ohitetaan standardi version WEB –sivu pohjainen konfigurointi suoraan ja siirrytään kohtaan, jossa loppu tulokset eli esitallennettu konfiguroinnin XML tiedosto olisi valmis WEB –sivuston Solverin pohjalta. Tunnistuksen jälkeen luetaan mallille kuuluva esitallennettu konfiguroinnin XML –tiedosto ja aloitetaan muokkaamaan XML –sisältöä. Opinnäytetyössä tehtiin tätä tarkoitusta varten myös sopiva moottori WMDHOTCoden kentän arvojen pohjalta seuraavasti (kuva 33):

```
//Rakennetaan HOTCoden pilkkoja ja XML:n muuttaja
// KW31/N-2000-1600-350,WF=A,UV=N,FD=Y,MV=N,UB=A,MA=SS,DS=200
        _WMDHOTCode_con = str2con(_WMDHotCode,",");
        for (i=1;i<=conLen(_WMDHOTCode_con);i++)
        {
            if (i==1)
            {
                _AC_prefix = ""; value = "";
                _name_and_dimension = conPeek(_WMDHOTCode_con,i);
                _WMDHotCode_value_pair = str2con(_name_and_dimension,"-");
                if (conLen(_WMDHotCode_value_pair)>=3)
                {
                    l = conPeek(_WMDHotCode_value_pair,2);
                //if (_XMLRead.attributes().getNamedItem('name').value() =="L")
                //  _XMLRead.attributes().getNamedItem('value').value(l);
                    _key_con += "L"; _value_con += l;
                }
                //mlAttributeRead.name() == "SIGN".
                //rootNode.attributes('L');
                //xmlDoc.selectSingleNode('Product/@action).value()
                //XMLRead = rootNode.selectSingleNode('"//"+XmlName);
                if (conLen(_WMDHotCode_value_pair)>=3)
                {
```

```
            w = conPeek(_WMDHotCode_value_pair,3);
        //if (_XMLRead.attributes().getNamedItem('name').value() == "W")
        //  _XMLRead.attributes().getNamedItem('value').value(w);
            _key_con += "W"; _value_con += w;
        }
        if (conLen(_WMDHotCode_value_pair)>=4)
        {
            h = conPeek(_WMDHotCode_value_pair,4);
        //if (_XMLRead.attributes().getNamedItem('name').value() == "H")
        //  _XMLRead.attributes().getNamedItem('value').value(h);
            _key_con += "H"; _value_con += h;
        }
        if (conLen(_WMDHotCode_value_pair)>=5)
        {
            b = conPeek(_WMDHotCode_value_pair,5);
        //if (_XMLRead.attributes().getNamedItem('name').value() == "B")
        //  _XMLRead.attributes().getNamedItem('value').value(b);
            _key_con += "B"; _value_con += b;
        }
    }
    else
    {
        if (_AC_prefix == "")
        {
            _WMDHotCode_value_pair = str2con(conPeek(_WMDHOTCode_con,i),"=");
            key = conPeek(_WMDHotCode_value_pair,1);
            value = conPeek(_WMDHotCode_value_pair,2);

            if (strScan(key,"AC",0,strLen(key)))
            {
                _AC_prefix = "AC_"; value="true";
            }
       //if (_XMLRead.attributes().getNamedItem('name').value() == key)
       //     _XMLRead.attributes().getNamedItem('value').value(value);
            _key_con += key; _value_con += value;
        }
        else
        {
            _WMDHotCode_value_pair = str2con(conPeek(_WMDHOTCode_con,i),"=");
            key = _AC_prefix;
            key += conPeek(_WMDHotCode_value_pair,1);
            value = "true"; //conPeek(_WMDHotCode_value_pair,2);

        //if (_XMLRead.attributes().getNamedItem('name').value() == key)
        //     _XMLRead.attributes().getNamedItem('value').value(value);
            _key_con += key; _value_con += value;
        }

    }

}
//if (conLen(_WMDHotCode_con) < 9) error(strFmt("HOTCode is not proper: %1",_WMDHotCode));
// Get the XML document
doc = new XMLDocument();
//doc.async(FALSE);
//doc.load(@""+_out_filename+".tmp"); //C:\\FileNameToRead.xml");
doc.loadXml(_xml_session);
xmlError = doc.parseError();

if (xmlError && xmlError.errorCode() != 0)
{
    warning(strFmt("Error: %1",xmlError.reason()));
    //pause;
    //return;
}

rootNode = doc.documentElement();


_XMLRead = rootNode.selectSingleNode("Component"); //+XmlName);
```

```
    NumberElement = _XMLRead.childNodes().length();
    for(x = 1;x<=NumberElement;x++)
    {
        if(x==1)
        {
            _XMLRead = _XMLRead.firstChild();
        }
        else
        {
            _XMLRead = _XMLRead.nextSibling();
//if (_XMLRead.name() == "ORDER_REFERENCE")_XMLRead =_XMLRead.nextSibling();
        }
//payee_reference = rootNode.getNamedElement("PAYEE_REFERENCE");
    NumberOfAttributes = _XMLRead.attributes().length();
    _key_name = _XMLRead.attributes().getNamedItem('name').value();
        if (_key_name == "L" || first_found)
        {
            first_found = true;
            sijainti = conFind(_key_con,_key_name);
            if (sijainti)
            {
                _key_value = conPeek(_value_con,sijainti);
                _XMLRead.attributes().getNamedItem('value').value(_key_value);
            } else if (strScan(_key_name,"AC_",0,strLen(_key_name))) _XMLRead.attributes().getNamedItem('value').value("false");
        }
```

KUVA 33. D365 HOTCoden pilkkoja ja XML koodin muuttaja

Koodissa kuvassa 33 aluksi pilkotaan tarjouksen HOTCode –kentän sisältö vain osiin ja aluksi otetaan mitat ja sen jälkeen kaikki loput parametrit ihan suoraan nimellä, joiden odotetaan olevan myös tuotemallinnus attribuuttien nimiä suoraan eli ei mitään kovakoodauksia vaan käytetään samoja nimiä kuin oli HOTCode:ssa ja tuotemallilla. Yritetään löytää XML sanomasta sopiva parametri ja vaihtaa sille haluttu arvo HOTCoden mukaiseksi. Jollei kohdistus onnistu niin tehdään Error –tasoinen virhe asiasta, jonka erätyön virheen keräys kerää ja osaa tarvittaessa päättää rivin mahdollisesti Error –tilaan. Virheilmoitus tietenkin helpottaa testaushenkilöitä löytämään nimeämisvirheet tästä hyvin universaalista järjestelmästä, jossa läpilähetyksen myötä vain datavirheet ovat mahdollisia.

5.4.2 Automaattisen konfigurointi –prosessin kiihdytys

Automaattiselle konfigurointi-prosessin kiihdytykselle on suuri tarve, koska jo vanhan AX2009 version kanssa on ollut tilanteita, että myyntihenkilöt soittavat vientiassistentille, että joko heidän lähettämänsä tarjoukset ovat lähteneet eteenpäin. Tämän haasteen vuoksi D365:n puolelle luodaan vähintään kaksi AutomaticConfiguration batch Job –eräajo prosessia, jotka käynnistyvät 2 minuutin syklillä ja molemmat ajastetaan 1 minuutin välein käynnistymään eli parillisilla ja parittomilla minuuteilla. Saadaan aikaan 1 minuutin välein kiihtyvä prosessointi.

AutomaticConfiguration batch Job –eräajo siis pyrkii kahden, tai jos kyseessä todella iso tarjous, niin useamman prosessin voimin eräajojen voimin 1 minuutin välein käynnistymi-

sillä tarjoamaan apua ja nopeutusta kaikkeen eli tarttuvat kaikkiin tilauksiin ja jos ko. käynnistys prosessi olisin vielä menossa niin aloittaa 1min syklillä taas uuden prosessin, jossa tarjouksen otsikolla (header) on asetukset WMDAutomaticConfiguration = ON & WMDConfigurationStatus = ReadyForConfiguration ja riveillä on vastaavat asetukset samoilla asetuksilla eli isotkin tarjoukset voidaan kiihtyvällä tahdilla konfiguroida todella nopeasti. (Dynamics 365 2020d.)

Tässä suunnitellussa järjestelyssä on hienoa se, että jos esim. haluaisimme konfiguroida 100 rivin tarjouksen ja aiemmin tehdyn prosessin kellotuksen mukaan yhden rivin konfigurointi kestää 20 sekuntia per rivi yhdellä prosessilla, niin useampi prosessi nopeutta valmistumista logaritmisesti eli isommissa konfiguroinneissa uusi erätyö 1 minuutin syklillä aiheuttaa sen, että joka minuutti ilmestyy uusi prosessi käsittelemään rivejä. Saavutamme yhden prosessin ajolla, ajan 20s X 100 riviä on 2000 sekuntia eli noin 33 minuuttia, mutta kun käytetään rinnakkaisajoa, niin lopullinen aika saadaan pilkottua alle 9 minuuttia ja vielä suuremmissa tarjouksissa yhden prosessin ero on vielä suurempi. Kehitetyille ratkaisuille ei ole vielä tehty todellista kellotusta, mutta sellainen saadaan pian, koska tuotannon pilot generointi on alkanut.

5.4.3 Common Business Event perusta tilanteen jakamiseen

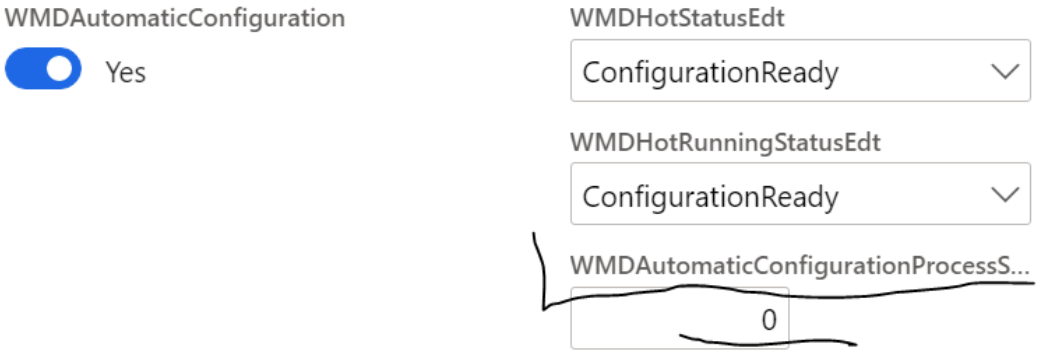
D365 tarjoaa Business Eventejä moneen tauluun standardina, mutta nyt ei voitu hyödyntää tätä ominaisuutta täysin, koska ei haluttu montaa erilaista sanomaa. Haluttiin Universaali Business Event, koska ei ole tarkoituksen mukaista saada jokaisesta eritaulusta täysin erilaista sanomaa Azure Bus –jonoon. Ajatus oli, että jokainen Common Business Eventiä kutsuva tietää asiansa, jota haluaa mainostaa Azure Bussissa, jolloin ymmärrettiin, että tämän kaltainen Business Eventti voitaisiin kytkeä, vaikka napin painallukseen tai kentän muutokseen.

Tarjouksen tai rivin lisäys ja päivitys kytkettiin tähän uuteen Common BusinessEventtiin. HOTApi –server tarkkailee tätä tiettyä jonoa eikä useita erillisiä, jolloin tilatietojen vaihtuessa palvelin ymmärtää kysyä tarjouksen ja sen konfiguroinnin tilaa tarvittaessa. Lopulta HOTApi –server tietää mihin tilaan importoitu tarjous on konfiguroinnin osalta päättynyt. Nämä tiedot välittää HOT –Clientille, jottei myyntihenkilö hätäile ja soita vientiassistentille turhaan.

5.4.4 Tarjouksen konfiguroinnin tilanteen jakaminen

Myyntihenkilö näkee oman Java -pohjaisen HOT Clientin kautta tilanteen, koska HOT client kysyy tilannetta HOTApi -rajapinnalta ja tämä kysyy konfiguroinnin tilannetta D365:n

puolelta kysymällä lähetetyn tarjouksen WMDRunningConfigutationStatus –kentän tilaa. Prosessin ajotilan kyselytarpeesta johtuen useamman eräajon kiihdytys ominaisuuden mahdollistamisen vuoksi ei voida päivittää suoraan WMDConfigurationStatus –kenttää, koska jos tämä kenttä päivitettäisiin ConfigurationIsRunning tilaan, niin avustavat erätyöt eivät tarttuisi tekemättömään työhön.



KUVA 34. D365 Automaattisenkonfiguroinnin yhtäaikaisten prosessien parametri

Tarjoukselle lisättiin kuvassa 34 myös prosessien yhtäaikaisien ajojen pinon ilmoittava määrä, jonka maksimi on esitetty kappaleessa 5.3 eli jokainen erätyö, joka tarttuu toimeen, lisää itsensä tarjouksen otsikolle WMDUnderGoingProcess –kenttän +1 arvolla.

Kun prosessi päättää toimet ja kun ne eivät enää löydä uutta työtä, niin vähentävät pinon arvoa aina -1. Lopulta viimein se viimeinen prosessi on lopettamassa työtään ja huomaa että arvo meni 0 –arvoon, niin silloin lopuksi laskee virheelliset rivit yhteen ja generoi riveistä ilmoituksen tarjouksen otsikolle menikö ERROR vai ConfigurationReady arvoon. Tämä loppuraportti ja tila generoidaan keräämällä asetuksen mukaiset statuksella olevat rivit konfiguroinnin loki –sanomasta. Tähän vaikuttavat aiemmat asetukset, joilla määritettiin mikä oli Automaattisen konfiguroinnin virhe taso, jonka tyyppiset määritetään virheeksi. Lisäksi tämä muodostettava AutomaticErrorStatus –tiedosto lisätään otsikolle liitetiedostoksi samoilla aputoiminnoilla kuin aiemmatkin konfiguroinnin liitetiedostot, koska näin pystytään välttämään virheen sattuessa InfoLog –transaktionin peruuntuminen. (Microsoft Dynamics Community 2020b.)

6 YHTEENVETO JA JATKOKEHITYS

Digitalisaatio on edennyt siihen pisteeseen, että pilvipalvelut yleistyvät entistä nopeammin ja kuitenkin näidenkin järjestelmien välille tarvitaan integraatiota tapahtumien ja datan välittämiseksi, jotta digitalisaation edut saavutetaan, eli laaja sähköinen asioiminen. Samanlainen tarve oli myös yritys-x:n HOT –clientin ja D365:n välillä, eli tehty data täytyy saada helposti eteenpäin, jotta se voi jalostua eteenpäin ja saavutetaan etuja.

Suurimpia haasteita opinnäytetyön tekemisessä oli tutkia asiaa, josta ei ole paljon mitään dokumentaatiota. Jouduttiin ajamaan ohjelmallisia toimintoketjuja lävitse Visual Studiolla Debugger toiminnolla, jolloin ajettava koodi on näkyvissä ja muuttujat luettavissa. Lisävaikeutta teki se tosiasia, sekä Visual Studio oli uutta D365 kehityksessä, että itse D365 ympäristönä, joka on paikallispilvessä tai pilvikeskuksessa. Todella monta uutta asiaa oli omaksuttava ja oppimiskäyrä oli melko jyrkkä.

Itse koodausvaiheessa ongelmaksi muodostui, ettei kaikkiin paikkoihin saanut lisätä koodia, koska kohde oli joko protected tai private määrittelyllä. Kaiken lisäksi koodausmahdollisuudet ovat Pre, Post ja Cag eli ennen, jälkeen ja lisätään koodia niin että alkuperäinen ajetaan ensin tai jälkeen päin. Vain tällä Cag (Confidentiality Advisory Group) pystytään pureutumaan suljettuihin osiin riittävästi, jotta muutos saadaan aikaan. Ei kuitenkaan pystyttäisi estämään olemassa olevaa koodia ajautumasta.

Opinnäytetyön aluksi asetettiin tutkimuskysymyksiä, joita tutkittiin laajasti. Esimerkiksi sekä yrityksen-x:n nykyinen tuotemallinnus, että miten saataisiin reitit ja rakenteet laskennallisesti ylläpidettäväksi HMPCE moottorin puolelle, onnistuivat lopulta ongelmitta. Tutkimuskysymyksistä monet olivat ns. ”Show Stoppereita”, jolloin jos näitä ei pystyttäisi ratkaisemaan, niin koko projektivaiheen alkaminen olisi mahdotonta, ja sen vuoksi aluksi tutkittiin paljon koko D365:n mahdollisuuksista vastata haasteisiin, joita oltiin esitetty alkukysymyksissä. Alun ongelmallisten alkututkimusten pohjalta saatiin positiivisia ratkaisuja kaikkiin opinnäytetyössä esitettyihin kysymyksiin. Konfiguroinnin kytkentä ulkoiseen HMPCE –laskukoneeseen onnistui ongelmitta ja konfiguroinnin nopeus oli ehkä merkittävämpiä ongelmia, joka ratkaistiin. Yksi suurimmista ratkaisuista oli se, että miten voitaisiin konfiguroida tarjouksen rivi ilman käyttäjän tekemiä uudelleen valintoja? Lopulta tämä oli myös kaikkein hankalammin ratkaistavissa oleva ongelma, koska ko. mahdollisuutta ei standardi D365 tarjoa ollenkaan. Kuitenkin ratkaisu saatiin kaikkiin mahdollisiin ongelmiin. Projektin edistyessä lisäksi ongelmia tuotti tilanteet, joissa tuoterakenteet sisälsivät ns. kalliita ostettavia osia. Näissä kalliissa tuoterakenteissa loppuhinta oli ongelmallinen ja näihin tarvittiin ratkaisu, joka ratkaistiin onnistuneesti.

Kokonaisuutena opinnäytetyön tutkimuskysymykset tulivat ratkaistua täysin.

Varsinaista jatkokehitystarpeita ei jätetty projektin ulkopuolelle, koska tahtotila oli selvä alusta alkaen, johtuen siitä, että opinnäytetyö kuvasi ERP –päivitys projektia.

Pieniä lisäkehitystarpeita tuli alun jälkeen monesti, mutta mitään ei jäänyt toteuttamatta. Lähtökohtaisesti tulleet lisäkehitystarpeet olivat asiakkaan oman toiminnan käytön helpottamisominaisuuksia. Tulevaisuudessa tiedetään, mitä vielä muokataan, kun projekti etenee tuotannon käyttöönoton vaiheeseen. Kaikki lisämuutokset ovat asiakkaan tarpeista ja tilauksesta riippuvaisia.

LÄHTEET

Developer Guide. 2020. Run a simple program using Customer Engagement web services. Viitattu 3.12.2020. Saatavissa https://docs.microsoft.com/en-us/dynamics365/customerengagement/on-premises/developer/simple-program-web-services

Dynamics 365. 2020a. Data entities overview. Viitattu 3.12.2020. Saatavissa https://docs.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/data-entities/data-entities

Dynamics 365. 2020b. Developer Guide for Dynamics 365 Customer Engagement (on-premises), version 9. Viitattu 3.12.2020. Saatavissa https://docs.microsoft.com/en-us/dynamics365/customerengagement/on-premises/developer/overview

Dynamics 365. 2020c. Open Data Protocol (OData). Viitattu 3.12.2020. Saatavissa https://docs.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/data-entities/odata

Dynamics 365. 2020d. Batch processing overview. Viitattu 3.12.2020. Saatavissa https://docs.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/sysadmin/batch-processing-overview

Dynamics 365 Community. 2020a. Using the UserConnection class to create a new transaction scope. Viitattu 3.12.2020. Saatavissa https://community.dynamics.com/ax/b/klaasdeforche/posts/using-the-userconnection-class-to-create-a-new-transaction-scope

Dynamics 365 Community. 2020b. X++ code for document attachment. Viitattu 3.12.2020. Saatavissa https://community.dynamics.com/365/financeandoperations/b/axaptavsme/posts/x-code-for-document-attachment

Dynamics 365 Community. 2020c. Difference between Connection Class and UserConnection Class for connecting to AX database to run stored procedure. Viitattu 3.12.2020. Saatavissa https://community.dynamics.com/ax/f/microsoft-dynamics-ax-forum/113395/difference-between-connection-class-and-userconnection-class-for-connecting-to-ax-database-to-run-stored-procedure?pifragment-96834=1

Dynamics 365. 2020e. What is ERP. Viitattu 8.12.2020. Saatavissa https://dynamics.microsoft.com/fi-fi/erp/what-is-erp/

JSON. 2020. Wikipedia. Viitattu 3.12.2020. Saatavissa https://fi.wikipedia.org/wiki/JSON

Konfigurointi. Sivistyssanakirja. Viitattu 8.12.2020. Saatavissa https://www.suomisanakirja.fi/konfigurointi

Lukka, K. 2001. Konstruktiivinen tutkimusote. Viitattu 3.12.2020. Saatavissa https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/

Massatuotanto. Wikipedia. 2017. Viitattu 8.12.2020. Saatavissa https://fi.wikipedia.org/wiki/Massatuotanto

Microsoft D365. 2020a. Business events overview. Viitattu 3.12.2020. Saatavissa https://docs.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/business-events/home-page

Microsoft D365. 2020b. Mikä ERP on ja miksi sitä tarvitaan. Viitattu 3.12.2020. Saatavissa https://dynamics.microsoft.com/fi-fi/erp/what-is-erp/

Microsoft Dynamics AX. 2020. Wikipedia. Viitattu 3.12.2020. Saatavissa https://en.wikipedia.org/wiki/Microsoft_Dynamics_AX

Microsoft Dynamics 365. 2020. Solver strategy for product configuration. Viitattu 3.12.2020. Saatavissa https://docs.microsoft.com/en-us/dynamics365/supply-chain/pim/solver-strategy-product-configuration

Moduuli. Wikipedia. 2015. Viitattu 8.12.2020. Saatavissa https://fi.wikipedia.org/wiki/Moduuli

Moottorin viritys. Qaz.Wiki. 2020. Viitattu 8.12.2020. Saatavissa https://fi.qaz.wiki/wiki/Engine_tuning

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V. & Bragge, J. 2006. The Design Science Research Process: A Model for Producing and Presenting Information Systems Research.

Säännöllinen lauseke. 2019. Wikipedia. Viitattu 3.12.2020. Saatavissa https://fi.wikipedia.org/wiki/S%C3%A4%C3%A4nn%C3%B6llinen_lauseke

Web service. 2020. Wikipedia. Viitattu 3.12.2020. Saatavissa https://en.wikipedia.org/wiki/Web_service

What is Azure? 2020. Viitattu 3.12.2020. Saatavissa https://azure.microsoft.com/en-gb/overview/what-is-azure/

What is JSON? 2020. Viitattu 3.12.2020. Saatavissa https://www.w3schools.com/whatis/whatis_json.asp