# MATERIAALIMÄÄRÄLASKENTAOHJELMAN OFFLINE-TILA

**Eetu Tolvanen**

**Opinnäytetyö**

**Ammattikorkeakoulututkinto**

Savonia
ammattikorkeakoulu

SAVONIA-AMMATTIKORKEAKOULU OPINNÄYTETYÖ

Tiivistelmä

| | |
|---|---|
| Koulutusala<br>Tekniikan koulutusala | |
| Koulutusohjelma<br>Tietotekniikan koulutusohjelma | |
| Työn tekijä(t)<br>Eetu Tolvanen | |
| Työn nimi<br>Materiaalimäärälaskentaohjelman offline-tila | |
| Päiväys 31.5.2011 | Sivumäärä/Liitteet 46 |
| Ohjaaja(t)<br>Jukka Kinnunen | |
| Toimeksiantaja/Yhteistyökumppani(t)<br>Foster Wheeler Energia Oy, Timo Toropainen | |


Tiivistelmä

Opinnäytetyön aiheena oli kehittää offline-tila materiaalimäärälaskentaohjelmaan. Tavoitteena oli luoda ohjelmalle asetus, jota käyttämällä mallinnusarvot tallennetaan PDMS:n -tietokannan sijaan muualle.

Materiaalimäärälaskentaohjelma on AVEVA PDMS -suunnitteluohjelman alla toimiva aliohjelma. Se tuottaa 3D-mallinnuksia voimalaitoskattilan osista ja laskee niiden materiaalimäärät. Materiaalimäärälaskentaohjelma on toteutettu C# -ohjelmointikielellä. Ohjelmaa kehitetään Foster Wheeler Energia Oy:lle.

Offline-tilan toteutukseen valittiin XML. Ensimmäisenä oli opeteltava XML-rakennekuvauksien käyttö C# -ohjelmointikielellä. Seuraavaksi tutustuttiin materiaalimäärälaskentaohjelman toimintaan ja toteutukseen. Tämän jälkeen alkoi offline-tilan kehitys C# -ohjelmointikielellä käyttäen Microsoftin Visual Studio 2005 -ohjelmaa. Offline-tilan kehittäminen ei muuttanut alkuperäisen ohjelman toimintaa mitenkään. Sen tuomat ominaisuudet näkyvät ainoastaan kun offline-tila on kytketty päälle.

Työn lopputuloksena oli toimiva materiaalimäärälaskentaohjelman offline-tila, joka voidaan kytkeä ohjelmassa päälle tarvittaessa.



Avainsanat
AVEVA PDMS, .NET, C#, XML

SAVONIA UNIVERSITY OF APPLIED SCIENCE THESIS

Abstract

| | | | |
|---|---|---|---|
| Field of Study<br>Technology, Communication and Transport | | | |
| Degree Programme<br>Degree Programme in Information Technology | | | |
| Author(s)<br>Eetu Tolvanen | | | |
| Title of Thesis<br>Offline mode of the material take off program | | | |
| Date | 31.5.2011 | Pages/Appendices | 46 |
| Supervisor(s)<br>Jukka Kinnunen | | | |
| Project/Partners<br>Foster Wheeler Energia Oy, Timo Toropainen | | | |


Abstract

The topic of this thesis was to develop an offline mode for material take off program. The aim was to create a setting for the program which, when used, allows the modeling values to be saved elsewhere instead of the PDMS database.

The material take off program is a subprogram of AVEVA PDMS design software. It creates 3D models from power plant parts and does material calculations. The material take off program is implemented with C# programming language. The program is being developed for Foster Wheeler Energia Oy.

XML was selected for the implementation of the offline mode. The first step was to study the use of XML in C# programming language. The next step was to get familiarized with the operation and implementation of the material take off program. After that the development of the offline mode with C# programming language was started by using Microsoft Visual Studio 2005 software. The development of the offline mode didn't change the operation of the original program in any way. The additional features brought by the offline mode appear only when the offline mode is switched on.

The result of this project was a functional offline mode for the material take off program which can be turned on when necessary.



Keywords
AVEVA PDMS, .NET, C#, XML

ALKUSANAT

Tämä opinnäytetyö tehtiin Foster Wheeler Energia Oy:lle. Työn tarkoituksena oli toteuttaa offline-tila materiaalimäärälaskentaohjelmaan. Haluan kiittää opinnäytetyön ohjaajaani Jukka Kinnusta saamistani neuvoista ja ohjauksesta. Lisäksi tahdon kiittää Timo Toropaista ja Ari Ojalaa mahdollisuudesta työskennellä tämän projektin parissa, sekä annetuista neuvoista sitä tehdessäni.

Uskon, että tämän projektin parissa työskentely on kehittänyt minua paremmaksi ohjelmoijaksi.

SISÄLTÖ

# KÄSITTEET

## .NET

.NET Framework on Microsoftin kehittämä kokoelma luokkakirjastoja, jotka tarjoavat ohjelmistokehittäjille yleisiä toimintoja ja palveluita. .NET Frameworkia käyttävät Microsoftin VisualStudio.NET -ympäristössä kehitetyt ohjelmistot. Se sisältää Common Language Runtimen, joka hallitsee .NET ohjelmien suorittamista.

## BCL

Base Class Library on luokkakirjasto, jota kaikki .NET Frameworkin ohjelmointikielet voivat käyttää. Se sisältää valtavan määrän yleisiä funktioita helpottaen ohjelmoijien työtä.

## C#

Microsoftin .NET konseptia varten kehittämä ohjelmointikieli. C# on tehokas, mutta helppokäyttöinen, olio-pohjainen ohjelmointikieli.

## CFB

Circulating Fluidized Bed laitokset polttavat polttoainetta alhaisemmassa lämpötilassa kuin normaalit laitokset. Polttoaineeseen lisätään kalkkikiveä, jonka kuumuus kattilassa muuttaa kalsiumoksidiksi. Se imee rikkidioksidia itseensä, jonka ansiosta CFB-laitokset saastuttavat vähemmän.

## CI-sheet

CI-sheetit sisältävät painerungon päämittoja, jotka on määritelty lämpölaskennan mukaan.

## CLR

Common Language Runtime on erityinen ajoympäristö, joka tarjoaa infrastruktuurin Microsoftin .NET Frameworkille.

CSV

Comma-Separated Values on tiedostomuoto, jolla voidaan tallentaa taulukkodataa tekstitiedostoon. Taulukkorakenteen muodostamiseksi eri kentät on eroteltu toisistaan pilkuilla.

DLL

Dynamic-link library on Microsoftin toteutus jaetusta kirjastosta. Jaetut kirjastot ovat tapa jakaa ohjelmakoodia ja dataa usean ohjelman kesken.

DSM

Design Standard Manual sisältää yksityiskohtaista mitoitustietoa painerungon osille ja suuntaa-antavia minimi- ja maksimiarvoja. Ohjeissa on kerrottu myös lujuuslaskennan tarpeita tiettyjen osien mitoitusten ja materiaalien määrittelyn suhteen.

DTD

Document Type Definition määrittää rakenteisen dokumentin ilmenemismuodot elementeille ja attribuuteille ja muodostaa määrittelyistä merkintäkielen.

GML

Generalized Markup Language on makrokokoelma, jolla voidaan toteuttaa marginaali tageja IBM:n tekstimuotoilijalle.

GUI

Graphical User Interface, eli graafinen käyttöliittymä tarkoittaa tekstiin, kuviin ja käyttöliittymäelementteihin perustuvaa tapaa käyttää tietokoneohjelmaa.

HTML

Hypertext Markup Language on kuvauskieli, jolla voidaan kuvata hypertekstiä. HTML tunnetaan erityisesti kielenä, jolla internetsivut on koodattu.

KKS

KKS on voimalaitoksien tunnistusjärjestelmä, joka auttaa tunnistamaan eri laitokset, systeemit, osat ja komponentit, sekä niiden tarkoituksen, tyypin ja sijainnin.

PDMS

Plant Design Management System on 3D-suunnitteluohjelma, jolla voidaan suunnitella tehtaita ja laitoksia.

SGML

Standard Generalized Markup Language on metakieli, jolla voidaan määritellä dokumenttien merkintäkieliä. Se pohjautuu Generalized Markup Languageen.

TAG

Tietokone järjestelmissä tag on avainsana tai termi joka on asetettu palalle tietoa. Tällainen metatieto auttaa kuvaamaan osiota ja auttaa löytämään sen uudelleen.

XML

Extensible Markup Language on rakenteellinen kuvauskieli, jolla voidaan jäsentää tietoa selkeästi. Sitä käytetään formaattina tiedonvälitykseen järjestelmien välillä ja dokumenttien tallentamiseen.

XML PARSER

XML-parserin tehtävä on tarkistaa, että XML-dokumentti täyttää määritetyt rakenteet, kelpuutukset ja rajoitukset.

# 1 JOHDANTO

Opinnäytetyön aiheena oli kehittää offline-tila Foster Wheeler Energia Oy:llä kehitysvaiheessa olevaan materiaalimäärälaskentaohjelmaan. Ohjelma on AVEVA PDMS –suunnitteluohjelman alla toimiva aliohjelma. Sen tarkoituksena on nopeuttaa ja tarkentaa painerungon tarjousvaiheen materiaalilaskentaa hinnoittelua ja projektiostoja varten. Ohjelma tekee halutusta kattilan osasta annetuilla mallinnusarvoilla karkean mallin Avevan PDMS-ohjelmaan, josta saaduilla arvoilla se laskee materiaalimääriä. Ohjelma on toteutettu C# -ohjelmointikielellä. Offline-tilan tarkoituksena on tarjota vaihtoehtoinen mallinnusarvojen tallennuspaikka. Jos PDMS:n schematic-tietokantaan ei saada yhteyttä, tai sitä ei muuten haluta käyttää, voidaan offline-tila kytkeä päälle. Offline-tilan toteutukseen valittiin XML-teknologia.

Opinnäytetyötä tehdessä täytyi tutustua moniin asioihin. Ohjelmointi tekniikoista täytyi tutustua .NET Frameworkiin, C# -ohjelmointikieleen, olio-ohjelmointiin ja XML-rakennekuvauksiin. Ohjelmistoista uutena opinnäytetyötä tehdessä tuli AVEVA PDMS 3D suunnitteluohjelma, sekä tietenkin itse materiaalimäärälaskentaohjelma.

Materiaalimäärälaskentaohjelmaa oli ennen minua ollut kehittämässä kaksi ohjelmoijaa, joten ohjelmassa oli jo hyvä pohja valmiina. Ohjelmalla pystyttiin mallintamaan ekonomaisereita ja convection cage, sekä laskemaan niiden materiaalimääriä. Itse kehitin ohjelmaan wing wall tulistimien ja evaporaattoreiden mallintamisen offline-tilan lisäksi, sekä aloitin reheatereiden ja tulistimien mallinnuksen toteuttamisen. Näiden lisäksi ohjelmaan on tarkoitus tulla ainakin tulipesän ja separaattorin mallinnus mahdollisuudet. Tähän mennessä ohjelmalla on saatu aikaan hyviä tuloksia. Materiaalimääriä on pystytty laskemaan ±5 % tarkkuudella, joissain osissa on päästy jopa ±2 % tarkkuuteen.

# 2 FOSTER WHEELER JA CFB-TEKNOLOGIA

## 2.1 Foster Wheeler Energia Oy

Foster Wheeler Energia Oy Group (FWEOY Group) kehittää tehokkaita ja ympäristöystävällisiä energiaratkaisuja. Se on erikoistunut voimalaitos- ja teollisuuskattiloihin ja niiden kunnossapitoon ja huoltoon. Yhtiö on energia-alan teknologian ja tuotekehityksen edelläkävijä koko maailmassa.

Foster Wheeler Energia Oy Group on osa maailmanlaajuista Foster Wheeler AG-konsernia. Yhtiö työllistää yhteensä yli 14 000 ihmistä ympäri maailman. Toimipisteitä on esimerkiksi Pariisissa, Madridissa, Milanossa, New Jerseyssä, Delhissä ja Singaporessa. Suomessa Foster Wheeler toimii Espoossa ja Varkaudessa. FWEOY Group työllistää Suomessa lähes 520 henkilöä ja sen pääkonttori Suomessa on Espoossa. Henkilöstöstä suurin osa työskentelee Varkaudessa, jossa sijaitsevat myös yrityksen suunnittelu- ja tuotantotoiminnot. (Foster Wheeler Energia Oy yritysinfo.)

Yhtiön ydinosaamisena ovat korkean hyötysuhteen omaavat ja matalapäästöiset CFB- eli kiertopetikattilat. Yhtiö on noin 40 % markkinaosuudellaan maailman johtava CFB-kattiloiden toimittaja. Foster Wheeler AG-konserniin kuuluvat yritykset ovat toimittaneet asiakkailleen lähes 500 kattilayksikköä, joista yli 300 on CFB-kattiloita.
Vuonna 2008 Foster Wheeler Energia Oy Group:n liikevaihto oli noin 350 miljoonaa euroa. Käynnissä oli 26 projektia 14 eri maassa. (Foster Wheeler Energia Oy yritysinfo.)

## 2.2 CFB-teknologia

CFB- (Circulating Fluidized Bed) eli kiertopetikattilat ovat eräs Foster Wheelerin ydinteknologioista. Foster Wheeler on ollut CFB-teknologian edelläkävijä alusta alkaen. Yhtiö teetti ensimmäisen testi CFB-laitoksensa vuonna 1976 ja toimitti maailman ensimmäisen CFB-yksikön energiantuottoon 1979. (Leading the world in CFB 2009, 4)

Foster Wheeler on kasvanut vakaasti ja kehittänyt teknologiaansa. Nyt se on toimittanut yli 300 CFB-yksikköä asiakkailleen ympäri maailman. Yrityksellä onkin noin 40 % osuus globaaleista markkinoista. Menestys johtuu siitä, että Foster Wheelerin CFB-yksiköillä on erittäin hyvät hyötysuhteet ja saatavuus. Lisäksi yksiköt voivat käyttää useita eri polttoaineita ja ne saastuttavat vähän. (Leading the world in CFB 2009, 2)

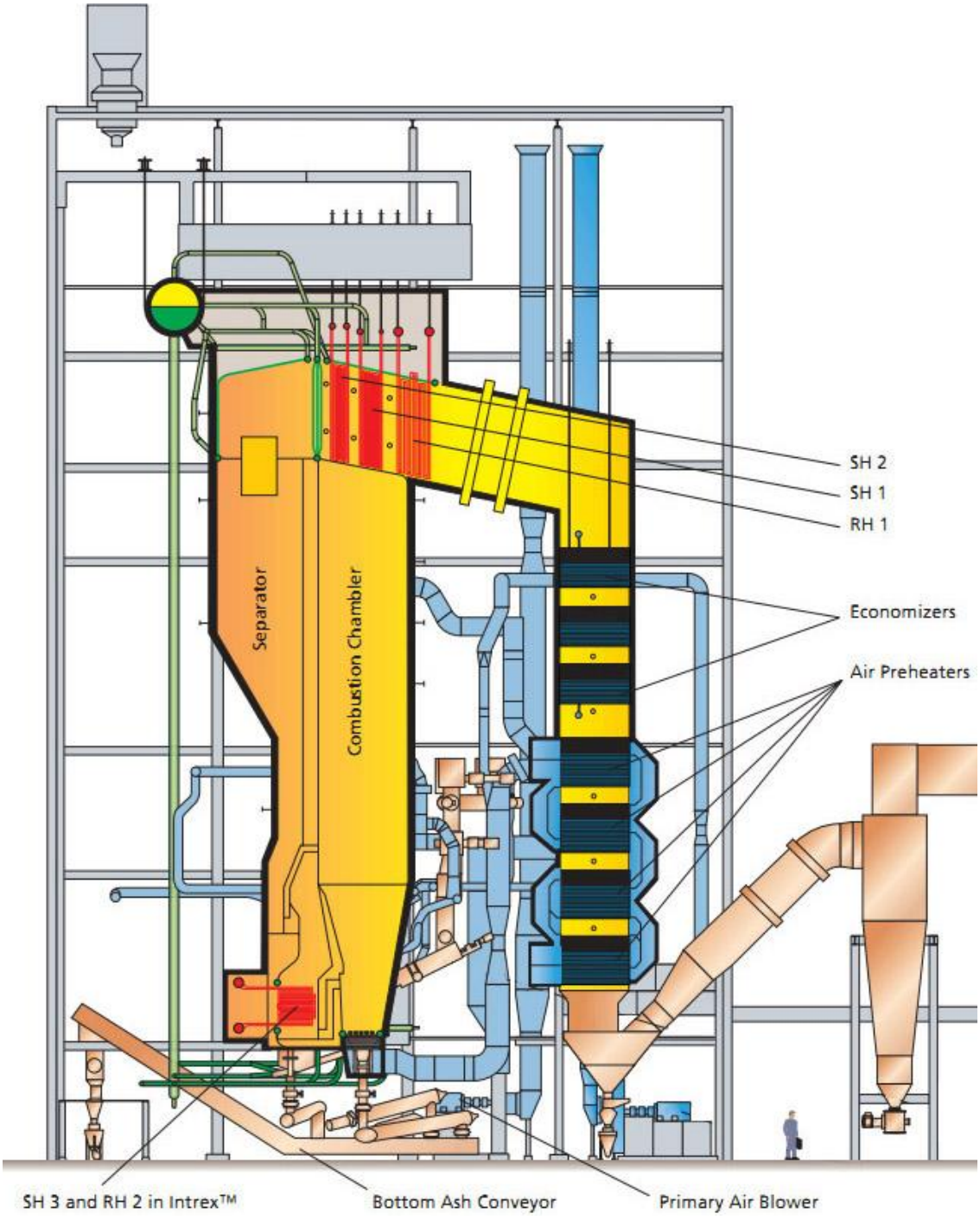
Foster Wheelerin CFB-teknologia on kehittynyt paljon matkan varrella. Kun CFB-yksiköitä alettiin toimittaa 20 vuotta sitten, olivat ne hyviä vain lähinnä hankalasti poltettaville polttoaineille. Nykyisin CFB on valtavirtaa edustava polttoteknologia, jota voidaan käyttää suuremmissakin kattiloissa. Foster Wheelerin yli 20 vuoden CFB-teknologian kehitys on saanut kaksi luonteenpiirrettä: vakaasti kasvavat yksiköiden koot ja entistä edistyksellisempiä ominaisuuksia. (Leading the world in CFB 2009, 4)

CFB-laitokset polttavat polttoaineensa alhaisemmassa lämpötilassa kuin normaalit laitokset. Polttoaineeseen lisätään kalkkikiveä, jonka kuumuus muuttaa kattilassa kalsiumoksidiksi. Kalsiumoksidi imee rikkioksidia itseensä, joten CFB-laitos saastuttaa tavallista laitosta vähemmän. (VTT 2009)

Kiertopetikattilassa leijutetaan hiekkaa tyypillisesti 3 – 10 m/s nopeudella. Hiekan raekoko on noin 0,1 – 0,5 mm. Kiertopetikattilassa on hiukkaset erottava sykloni, joka palauttaa kiertävän materiaalin ja palamattomat hiukkaset takaisin tulipesän pohjalle. Savukaasujen virtaus tulipesästä pohjalle estetään syklonin alle rakennetulla polvella. Jotta hiekka saadaan kiertämään, on polveen asennettu suuttimia, joista päästetään paineilmaa kattilaan (kuva 1). (VTT 2009)

Suuremmissa kattiloissa sykloneita on useampia, sillä niiden erotuskyky huononee, mitä isompi kattilan halkaisija on. Syklonin erotuskyky on parempi, kun savukaasu saadaan virtaamaan siihen tarpeeksi nopeasti. Perinteisesti syklonit ovat jäähdyttämättömiä, mutta uudemmissa kiertopetikattiloissa syklonit jäähdytetään. Jäähdytetyissä sykloneissa hiekan aiheuttamaa kulumista suojaava massaus on ohuempaa, jonka ansiosta kattilat ovat nopeampia ylösajaa ja niitä on helpompi säätää. (VTT 2009)

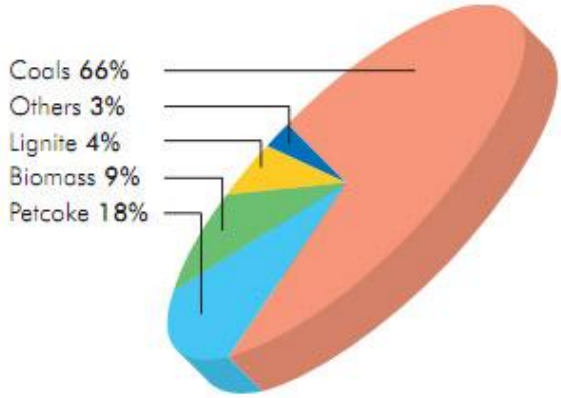
Kiertopetikattilaan syötetään polttoainetta joko etuseinän kautta tai sekoittamalla se syklonista palaavan hiekan sekaan. Palamisilmana käytetään osittain leijutus- eli primääri-ilmaa, jonka paine on noin 15 – 20 kPa. Sekundääri-ilmaa johdetaan muutamalle eri tasolle, pari metriä arinan yläpuolella. (VTT 2009)




KUVA 1. Kuva CFB järjestelmästä. (Leading the world in CFB 2009, 12)

## 2.3 CFB-teknologian etuja

Joustavuus polttoaineen valinnassa on eräs CFB-yksiköiden avainetuja. Foster Wheelerin CFB-kattiloissa voidaankin polttaa melkein mitä tahansa kiinteätä polttoainetta hyvällä hyötysuhteella ja vähäisillä päästöillä. Tämän ansiosta asiakkaat voivatkin tarpeen tullen vaihtaa polttoainetta tai vaikkapa käyttää hiiltä ja biopolttoainetta samanaikaisesti. Polttoainejoustavuuden ansiosta CFB-yksiköt sopivat hyvin useille eri asiakkaille (kuva 2). (Pioneering CFB Technology, 4)




KUVA 2. Foster Wheelerin valmistamien CFB-yksiköiden polttoaineiden käyttöosuudet. (Pioneering CFB Technology, 4)

Alhaiset $SO_x$/$NO_x$ (rikkioksidi/typpioksidi) -päästöt ovat toinen CFB-teknologian eduista. Päästöt ovat niin alhaiset, että ne läpäisevät tiukimmatkin päästövaatimukset. Kun uuniin syötetään kalkkikiveä, voidaan käyttää polttoaineita, jotka sisältävät paljon rikkiä. $NO_x$-päästöjä kontrolloidaan porrastetulla polttamisella ja alhaisella uunilämpötilalla. Uuniin voidaan myös syöttää ammoniakkia tai ureaa päästöjen vähentämiseksi. CFB-teknologian edut tulevat esiin varsinkin uudelleenkäynnistysprojekteissa, joissa teknologia voi tuoda merkittäviä eroja $SO_2$-, pienhiukkas-, $NO_x$- ja $CO_2$-päästöihin. $SO_2$- ja pienhiukkaspäästöjä voidaan yleensä laskea yli 90 %, $NO_x$-päästöjä 50 % ja $CO_2$-päästöjä noin 25 %. (Pioneering CFB Technology, 3)

CFB-yksiköt ovat todella kestäviä ja luotettavia. Vielä tuhansienkin käyttötuntien jälkeen suorituskyky on yleensä yli 98 %. Tämän mahdollistaa ennaltaehkäisevä kunnonvalvonta, asiantunteva ylläpito, nopeasti saatava korjaus ja helposti saatavat varaosat, jotka Foster Wheelerin huoltotiimi tarjoaa. (Pioneering CFB Technology, 4)

# 3 KÄYTETYT TEKNOLOGIAT

## 3.1 .NET Framework

Framework on kokoelma luokkakirjastoja, jotka tarjoavat kehittäjille yleisiä toimintoja ja palveluita, joita voidaan käyttää ohjelmointikielessä. .NET Framework laajentaa tätä konseptia. Siinä on valtava kokoelma luokkakirjastoja, joita voidaan käyttää useissa ohjelmointikielissä kuten C#, Visual Basic ja C++. Lisäksi se tarjoaa CLR:n (*Common Language Runtime*), joka hallitsee .NET kielillä kirjoitettujen ohjelmien suorittamista. .NET Frameworkin CLR hallitsee myös muistinkäyttöä, joka helpottaa ohjelmien kehitystä suunnattomasti. Luokkakirjastojen tarjoama toiminnallisuus helpottaa Windows ja Web -ohjelmien kehittämistä. (Youssef, 2005)

.NET Frameworkin luokkakirjasto on Microsoftin kirjoittama ja sitä on helppo käyttää. Jos esimerkiksi tahdot kirjoittaa konsoliin rivin tekstiä, voit käyttää metodia nimeltä WriteLine(), joka on osa Console-luokkaa. Ohjelmoijan täytyy vain syöttää metodiin string-muuttuja jonka haluaa kirjoitettavan konsoliin. .NET Frameworkissa on käytössä luokkia melkein mihin vain. Luokkia on esimerkiksi tekstinkäsittelyyn, säikeistykseen, XML:n käyttöön, input/outputtiin, Windows GUI:n (*Graphical User Interface*) luomiseen ja tietokantojen hyödyntämiseen. (Youssef, 2005)

On tärkeää huomata, että C# ei ole osa .NET Frameworkia. C# on itsenäinen olio-ohjelmointikieli, joka kuitenkin luotiin käytettäväksi .NET Frameworkin kanssa. C# kääntäjä tuottaa koodia, jota voidaan ajaa .NET ympäristöissä. Tätä koodia kutsutaan Managed Code:ksi, sillä sitä hallitsee CLR-ympäristö. Toisin sanottuna Microsoft suunnitteli C# kielen olemaan .NET alustan keskeisin ohjelmointikieli. (Youssef, 2005)

### 3.1.1 .NET Frameworkin historiaa

Microsoft aloitti .NET Frameworkin kehityksen 1990-luvun lopulla. Aluksi sen nimenä oli NGWS (*Next Generation Windows Services*). Ensimmäiset beta-versiot .NET 1.0:sta julkaistiin 2000-vuoden lopulla. (Layug 2008)

.NET Framework 1.0 oli .NET Frameworkin ensimmäinen julkaisu. Se julkaistiin 13.2.2002 Windows 98:lle, Windows NT 4.0:lle, Windows 2000:lle ja Windows XP:lle. Microsoftin valtavirtatuki versiolle päättyi 10.7.2007 ja pidennetty tuki heinäkuussa 2009. .NET Framework 1.1 oli ensimmäinen suuri .NET Frameworkin päivitys. Se julkaistiin 3.4.2003. Se oli myös osa Microsoft Visual Studio .NET:in toista julkaisua. Tämä oli ensimmäinen .NET Frameworkin versio, joka sisällytettiin Windows käyttöjärjestelmään. .NET Framework 1.1 tuli Windows Server 2003:n mukana. (Layug 2008)

.NET Frameworkin versio 3.0, jota aiemmin kutsuttiin nimellä WinFX, sisältää uuden kokoelman hallinnoidun koodin ohjelmointirajapintoja, jotka ovat osana Windows Vista ja Windows Server 2008 -käyttöjärjestelmiä. Tässä julkaisussa ei ollut suuria arkkitehtuurisia muutoksia. .NET Framework 3.0 käyttää samaa CLR:ää (*Common Language Runtime*) kuin .NET Framework 2.0. Frameworkin 3.5 RTM (*Release To Manufacturing*) versio julkaistiin virallisesti 19.11.2007. Kuten 3.0 versiokin, se käyttää samaa CLR:ää kuin 2.0 versiokin. .NET Framework 3.5 asentaa itsensä lisäksi .NET Framework 2.0 version Service Pack ykkösen, joka lisää joitakin metodeja ja asetuksia BCL (*Base Class Library*) -luokkiin Frameworkin 2.0 versioon. Näitä tarvitaan Frameworkin 3.5 version ominaisuuksissa, kuten LINQ (*Language Integrated Query*), joka mahdollistaa datakyselyt .NET ohjelmointikielessä. Sillä voidaan tehdä kyselyitä XML-dokumentteihin, taulukoihin ja tietokantoihin. .NET Compact Framework 3.5 julkaistiin samanaikaisesti .NET Framework 3.5 kanssa. Se sisältää uusia ominaisuuksia Windows Mobile ja Windows Embedded CE -laitteille. (Layug 2008)

Frameworkin uusin versio on 4.0. Se julkaistiin vuonna 2010. Se sisältää lukuisia päivityksiä aiempiin ominaisuuksiin kuten CLR:ään ja BCL:ään, sekä parannuksia ASP.NET:iin.

## 3.2 C# -ohjelmointikieli

Kun .NET:iä kehitettiin, luokkakirjastot kirjoitettiin SMC (*Simple Managed C*) –kielellä. Tammikuussa 1999 Anders Hejlsberg muodosti ryhmän kehittääkseen uuden kielen nimeltään Cool (*C-like Object Oriented Language*). Microsoft harkitsi Cool nimen säilyttämistä, mutta päätti, että nimi täytyy muuttaa tavaramerkkisyiden takia. Kun .NET projekti julkistettiin yleisölle heinäkuussa vuonna 2000 Professional Developers Conference tapahtumassa, ohjelmointikielen nimeksi oli muutettu C# ja luokkakirjastot sekä ASP.NET ajoympäristö oli siirretty C#:lle. (Baskar 2011)

C# on sekoitus C, C++, Java ja Visual Basic -kieliä. C# -kielen on tarkoitus olla yksinkertaista, modernia ja monikäyttöistä olio-ohjelmointikieltä. Joskus C# -kieltä kutsutaan kirjallisuudessa tai internetissä nimellä C Sharp, kuten ohjelmointikielen nimi lausutaan, sillä jotkut ohjelmat eivät osaa käsitellä merkkiä #. (Baskar 2011)

C# syntaksi on helposti tunnistettavaa kaikille, jotka osaavat C, C++ tai Java -ohjelmointikieliä. Jos ohjelmoija osaa jotakin näistä kielistä, on työskentely C# -kielen kanssa nopeaa oppia. C# yksinkertaistaa monia C++ -kielen monimutkaisuuksia ja tarjoaa voimakkaita ominaisuuksia, joita ei ole Java:ssa. Olio-ohjelmointikielenä C# tukee kapselointia, perinnöllisyyttä ja monimuotoisuutta. (Microsoft)

Vaikka C# ohjelmien on tarkoitus olla kevyitä tietokoneelle, ohjelmointikieltä ei ole tarkoitettu kilpailemaan C tai assembly -ohjelmointikielien kanssa suorituskyvyllään. C# sisältää automaattisen Garbage Collection –prosessin, jonka mukaan CLR vapauttaa muistia, jota ohjelma ei enää tarvitse. (Microsoft)

## 3.3 Olio-ohjelmointi

### 3.3.1 Olio-ohjelmoinnin perusteet

Olio-ohjelmointi eli object oriented programming tarkoittaa, että ohjelma kirjoitetaan käsiteltävän ongelman muodostavien olioiden avulla. Olio-ohjelmoinnissa tieto ja siihen liittyvät toiminnallisuudet yhdistetään luokiksi. Jos ohjelmassa esimerkiksi halutaan kuvata jääkiekko-ottelun tuloksia, tarvitaan tietotyypit Pelaaja ja Joukkue. Perustietotyyppejä käyttämällä ei voida mallintaa elävän elämän olioita kunnolla. Jääkiekkopelaajaa ei voida mallintaa pelkillä perustietotyypeillä, kuten double tai int, vaan tarvitaan useita erilaisia ja erityyppisiä arvoja. Olio-ohjelmoinnin tavoitteena onkin mallintaa tietokoneohjelma mahdollisimman lähelle reaalimaailman olioita, esimerkiksi ohjelmalla ohjattavaa laitetta. (Horton 1999, 578)

Vaihtoehtona olio-ohjelmoinnille on ohjelmien kehittäminen lausekielisenä. Lausekielisessä ohjelman kehittämisessä ohjelma kirjoitetaan peräkkäisinä käskyinä perusrakenteita ja aliohjelmia käyttäen. Tällaiset ohjelmat etenevät suoraviivaisesti ”ylhäältä alas”. Pieniä ohjelmia kirjoitettaessa tämä lähestymistapa onkin ihan hyvä, sillä se on selkeää ja nopeaa luoda. Kun ohjelmiston koko kasvaa, lausekielisenä kehitetyn ohjelman ylläpito muuttuu hankalaksi, muutos yhteen kohtaan johtaa helposti muutostarpeeseen lukuisissa muissa kohdissa. (Halttunen 2005, 8)

Olio-ohjelmoinnin haasteena on oman ajattelutavan kääntäminen olioajattelutapaan. Olio-ohjelmointi voi tuntua todella raskaalta, jos on tottunut kehittämään lausekielisiä ohjelmia. Olio-ohjelmaa kehitettäessä joudutaan miettimään millainen on hyvä olio, jotta saataisiin mallinnettua reaalimaailmaa siten, että se olisi mahdollisimman selkeästi käsiteltävissä ohjelmallisesti. Oliot sisältävät luotavan olion tiedot ja siihen liittyvät käsittelytoiminnot. (Halttunen 2005, 9)

Haluttaessa ohjelmaan oliona auton moottori, luodaan sille luokka, jolla on tietoina vaikkapa teho, lämpötila ja tieto siitä onko moottori käynnissä vai ei. Sitten voidaan lisätä oliolle metodeja kuten moottorin käynnistäminen ja sammuttaminen sekä moottorin lämpötilan tai kierroslukumittarin lukeminen. Metodit on helppo määritellä jäljittelemään oikeaa elämää. Voidaan vaikkapa laittaa moottorin käynnistykseen ehto, joka määrää, että lämpötilan ollessa alle -30 °C auto ei käynnisty. Moottori-olio voidaan sisällyttää auto-olioon, jolloin auton toiminnan ohjelmallinen simuloiminen menee eteenpäin. (Halttunen 2005, 10)

### 3.3.2 Olio-ohjelmoinnin edut

Lausekielisessä ohjelmassa muuttujia on mahdollista käsitellä missä tahansa kohdassa ohjelmaa, siispä muuttujiin kohdistuvat muutokset saattavat edellyttää moneen kohtaan tehtäviä korjauksia eri puolilla ohjelmaa. Eräs olio-ohjelmoinnin keskeisimpiä periaatteita on, että luokan tiedot piilotetaan niin, että ne ovat näkyvissä ainoastaan kyseisen luokan sisällä. Piilottaminen onnistuu esittelemällä kaikki olion muuttujat private-määreellä. Näin muuttujat jäävät luokan sisäisiksi eikä tietoja voida käsitellä esimerkiksi pääohjelmassa, vaan niiden käsittely onnistuu ainoastaan luokkaan tehtyjen metodien avulla. Menetelmää kutsutaan kapseloinniksi. Pääohjelmassa kutsutaan vain luokan metodeja, joiden avulla pystytään käsittelemään olion tietoja. Jos olion tietoihin tai niiden käsittelyyn joudutaan tekemään muutoksia, ne tarvitsee tehdä pelkästään olion luokan sisältämiin tietoihin ja metodeihin. Näin pääohjelmaan tarvittavat muutokset jäävät paljon pienemmiksi. Jos ohjelman käyttöliittymä täytyy uusia täysin, se ei vaikuta itse ohjelman toimintoihin mitenkään, mikäli ne on luotu olioina. Luodaan vain uusi käyttöliittymä ja kutsutaan sen avulla jo olemassa olevia olioita. (Halttunen 2005, 11)

Yksi olio-ohjelmoinnilla saatava etu on, että valmista ohjelmakoodia voidaan mahdollisesti käyttää uudelleen jossakin toisessa projektissa. Esimerkiksi henkilö-olio, joka sisältää tiedot henkilön nimestä, osoitteesta, puhelinnumerosta ja muista perustiedoista, voi olla hyödyllinen useissa eri ohjelmissa. Kun olion on kerran tehnyt, voi sen kopioida suoraan uuteen ohjelmaprojektiin ja hyödyntää oliota siellä. (Halttunen 2005, 12)

## 3.4 XML

### 3.4.1 Mikä on XML?

Extensible Markup Language eli XML on yksinkertainen tekstipohjainen formaatti, jolla voidaan esittää tietoa rakenteellisesti. XML on nykyisin yksi yleisimmin käytetyistä formaateista jakaa jäsenneltyä tietoa. XML näyttää hyvin samankaltaiselta kuin HTML (*Hypertext Markup Language*), sillä molemmat kielet käyttävät tageja, mutta XML:n syntaksisäännöt ovat hyvin tiukat. XML-työkalut eivät käsittele XML-tiedostoja, jotka sisältävät virheitä. XML-dokumentit koostuvat yhdestä tai useammista elementeistä, jotka puolestaan sisältävät atribuutteja, toisia elementtejä tai tekstisisältöä.

### 3.4.2 XML:n historiaa

Rakenteiset dokumentit ja hyperteksti ovat olleet olemassa jo kauan. Rakenteisen dokumentin eli nykyisen hypertekstin perusajatuksen esitti ensimmäisen kerran Vanderman Bush jo vuonna 1945. 1960-luvun lopulla joukko IBM:n työntekijöitä alkoi kehittää lakiasiaintoimistoille yhteistä tekstinkäsittelyjärjestelmää. He ottivat lähtökohdakseen dokumentin sisällön ja jättivät sidoksen väline- ja laitealustaan vähemmälle huomiolle. Työryhmän keulakuvaksi nousi C. F. Goldfarb, joka kehitti rakenteisen dokumentin talletusjärjestelmän ja vaikutti myöhemmin oleellisesti menettelytavan standardoimiseen. (Laakkonen & Walkama 2004, 6)

Goldfarb vaikutti paljon Generalized Markup Language (GML) ja Standard Generalized Markup Language (SGML) -kielten kehitykseen. SGML:ää käytetään XML-skeemaa edeltävissä Document Type Definition (DTD) -kuvauksissa. SGML-pohjaisissa DTD-kuvauksissa havaittiin huomattavasti parantamisen varaa ja siksi kehitettiin XML-skeema. (Laakkonen & Walkama 2004, 6)

Kun XML 1.0 määritys saatiin valmiiksi vuonna 1998, W3C:llä oli paljon kehitysideoita tekniikalle. Ideoita kehittelemään ja toteuttamaan perustettiin XML Schema Working Group. Ryhmä kehitteli mm. primitiivi-tietotyyppien lisäämistä XML:ään, nimiavaruuksien integroimista tulevaan XML-skeemaan, rajoitteiden lisäämistä elementtien sisällölle ja elementtien periyttämisen mahdollisuutta. (Laakkonen & Walkama 2004, 6)

### 3.4.3 Document Type Definition

Yleisimmin käytetty tekniikka XML-dokumenttien rakenteen kuvaukseen aiemmin oli Document Type Definition eli DTD. DTD-tekniikka on osa SGML-standardia (Standard Generalized Markup Language). DTD-kuvaukset on siis kirjoitettu SGML-kielellä toisin kuin XML-dokumentit, joiden rakenteen ne määrittelevät. Kuvattava XML-dokumentti täytyy olla yhdistetty tiettyyn SGML-määritykseen ja DTD-kuvaukseen. Ilmentymä yhdistää dokumentin rakennemääräykset ja sen sisällön. (Laakkonen & Walkama 2004, 7)

SGML-määritys asettaa ympäristön perusasetukset, kuten käytetyn merkkivalikoiman (esim. ASCII), syntaksit ja tarvittavat lisäpiirteet. DTD-kuvaukset määrittelevät käsittelysäännöt XML-dokumentin parserille. Kuvaukset määräävät dokumentin rakenteen, rakenne-elementit, elementtien suhteet ja elementtien nimien ja sisällön muodon. DTD-kuvaukset, kuten XML-kuvauksetkin, ovat rakennemäärittelyjä, joita tarvitaan, jotta dokumentista voidaan luoda oikean muotoisia ilmentymädokumentteja. (Laakkonen & Walkama 2004, 7)

DTD-rakennekuvauksien käyttö on melko kömpelöä uusiin XML-kuvauksiin verrattuna. Jokaiseen DTD:tä hyödyntävän rakennekuvauksen alkuun lisätään dokumenttityyppimäärittelyt (kuva 3). Ne määrittävät DTD:n tunnisteen ja sijainnin, sekä tiedon mistä DTD-elementistä rakennekuvaus alkaa.

DTD-rakennekuvauksissa tiedon muotoa ei voida kuvata tietotyypeillä (esimerkiksi int, boolean, jne.), vaan määritysten täytyy perustua elementtien nimiin. Ainoa tapa kuvata sisällön muotoa ovat PCDATA (Parsed Character Data)- ja CDATA (Character Data) -määritykset. XML-skeema perustuu tietotyyppimääritelmiin, pelkkien elementtien sijasta. Tämän ansiosta XML-skeeman käyttöönotto ohjelmointikielissä ja tietokantojen yhteydessä on paljon helpompaa. (Laakkonen & Walkama 2004, 8)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE people_list [
  <!ELEMENT people_list (person)*>
  <!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT birthdate (#PCDATA)>
  <!ELEMENT gender (#PCDATA)>
  <!ELEMENT socialsecuritynumber (#PCDATA)>
]>
<people_list>
  <person>
    <name>Fred Bloggs</name>
    <birthdate>2008-11-27</birthdate>
    <gender>Male</gender>
  </person>
</people_list>
```

KUVA 3. Esimerkki DTD-rakennekuvauksesta

```
- <Superheaters>
  - <Component>
      <Title>FWHAH01</Title>
    </Component>
  - <Component>
      <Title>FWHAHTRN01</Title>
    </Component>
  </Superheaters>
```

KUVA 4. Esimerkki XML-kuvauksesta

Nykyisten XML-rakennekuvausten huomattavin ero DTD-rakennekuvauksiin on se, että rakennekuvauksetkin on kirjoitettu XML-kielellä (kuva 4). Se helpottaa kehittäjien työtä, sillä heidän ei tarvitse opetella erillistä kieltä rakennekuvauksia varten. Lisäksi se helpottaa parserikehittäjien työtä, koska samalla parserilla voidaan nyt jäsentää ilmentymädokumenttien lisäksi myös dokumenttien rakennekuvaukset. (Laakkonen & Walkama 2004, 7)

### 3.4.4 XML ja .NET

.NET Frameworkin XML-luokkien ytimessä on kaksi abstraktia luokkaa: XmlReader ja XmlWriter. XmlReader tarjoaa nopean lukutavan XML-dokumenteille. XmlWriter tarjoaa rajapinnan XML-dokumenttien tuottamiseen noudattaen W3C:n XML 1.0 suosituksia. Sovellukset, jotka haluavat prosessoida XML-dokumentteja, käyttävät XmlReaderiä. Mikäli sovellus haluaa tuottaa XML-dokumentteja, sen täytyy käyttää XmlWriteria. Molemmat luokat edellyttävät suoratoistomallia, jossa ei tarvita isoa välimuistia. (Skonnard 2001)

Sekä XmlReader että XmlWriter ovat abstrakteja pohjaluokkia, jotka määrittävät toiminnallisuuden, jota johdettujen luokkien täytyy tukea. Luokilla on erilaisia konkreettisia implementaatioita, XmlReaderillä kolme: XmlTextReader, XmlNodeReader ja XslReader. XmlWriterilla näitä on kaksi: XmlTextWriter ja XmlNodeWriter. XmlTextReader ja XmlTextWriter tukevat tekstipohjaisen streamin lukemista ja kirjoittamista kun taas XmlNodeReader ja XmlNodeWriter on suunniteltu työskentelemään DOM (Document Object Model) puiden kanssa. Tämän mallin suurin etu on, että itse räätälöityjä lukijoita ja kirjoittajia voidaan kehittää vapaasti laajentamaan toiminnallisuutta. (Skonnard 2001)

Esimerkki XmlWriter funktiosta:

```
public void WriteDocument(XmlWriter writer)
{
  writer.WriteStartDocument();
  writer.WriteComment("esimerkki kommentti");
  writer.WriteStartElement("henkilo");
  writer.WriteStartElement("nimi", "");
  writer.WriteString("Jaakko");
  writer.WriteEndElement();
  writer.WriteEndElement();
  writer.WriteEndDocument();
}
```

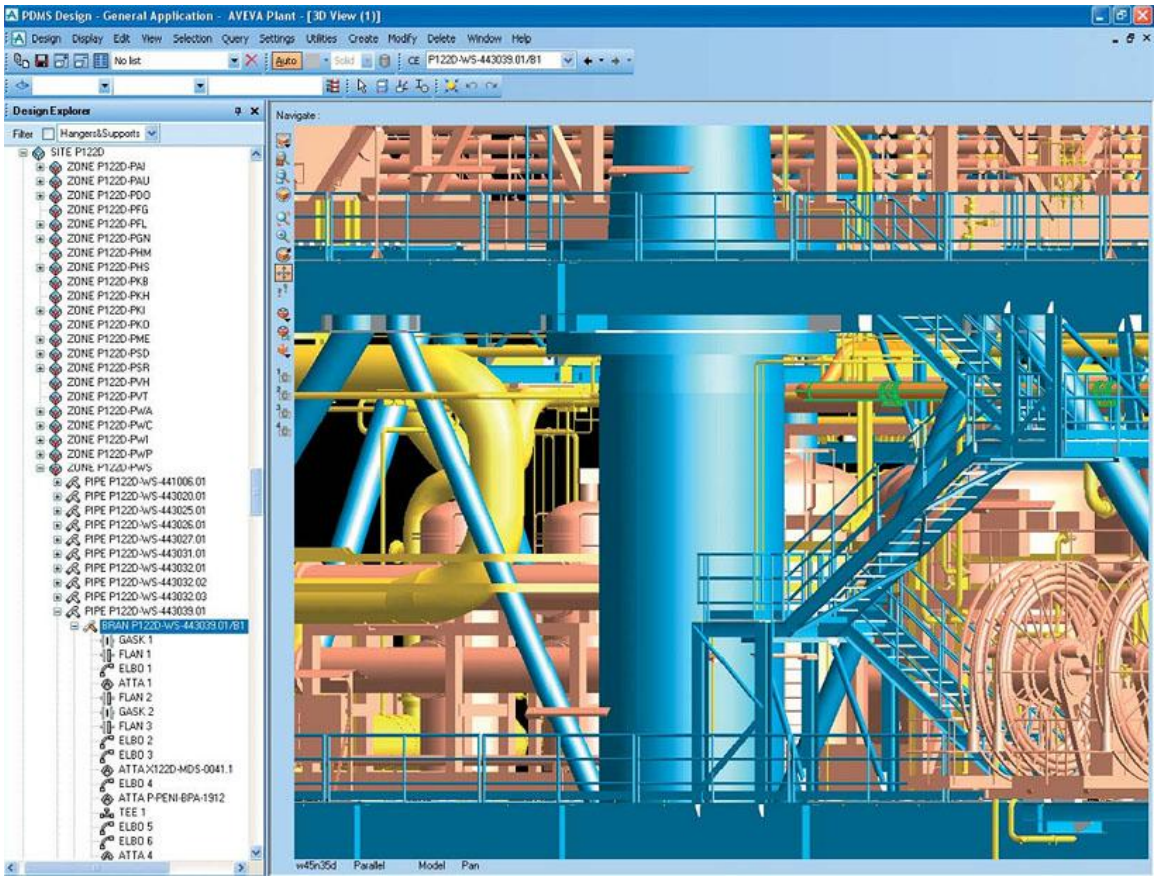
Edellinen funktio tuottaa seuraavanlaisen XML-dokumentin:

```
<?xml version="1.0"?>
<!--esimerkki kommentti-->
<henkilo">
  <nimi>Jaakko</nimi>
</henkilo >
```

## 3.5 AVEVA PDMS

AVEVA on eräs maailman johtavia tietokoneohjelmistotuottajia voimalaitos- ja meriteollisuuden alalla. Viimeisen 10 vuoden aikana yli 80 % suurista tuotantolaitoksista Pohjanmerellä ja Meksikonlahdella suunniteltiin käyttäen AVEVA:n ohjelmistoja. AVEVA:n teknologiaa käytetään 85 % Kiinan A-luokan sertifikaatin sähkölaitossuunnitteluinstituuteissa. (AVEVA)

AVEVA PDMS (*Plant Design Management System*) tarjoaa tehokkaan suunnitteluohjelman energiateollisuuteen. Siinä on helppokäyttöinen .NET teknologiaan perustuva käyttöliittymä (kuva 5). Sadat käyttäjät voivat työskennellä PDMS:llä yhdessä hallitusti ja jokainen voi nähdä täyden suunnitelman koko ajan. PDMS sisältää virheen tunnistukset, jotka auttavat löytämään törmäyskohdat ja epäjohdonmukaisuudet suunnitelmasta. Suunnitelmista voidaan tuottaa raportteja ja piirroksia automaattisesti käyttäen PDMS ohjelmistoja. Lisäksi se sisältää PML:n (*Programmable Macro Language*), jolla voidaan kustomoida ja automatisoida avaintoimintoja. (AVEVA 2008)




KUVA 5. PDMS:n käyttöliittymä. (AVEVA 2008)

# 4 MATERIAALIMÄÄRÄLASKENTAOHJELMA

## 4.1 Materiaalimäärälaskentaohjelma

Materiaalimäärälaskentaohjelmaa kehitetään nopeuttamaan ja tarkentamaan painerungon tarjousvaiheen laskentaa hinnoittelua ja projektiostoja varten. Materiaalimäärälaskentaohjelma tuottaa mallin valitusta kattilan osasta annetuilla arvoilla Aveva PDMS mallinnusohjelmaan, josta materiaalimäärälaskentaohjelma saa tarvittavia arvoja materiaalilaskentaa varten.

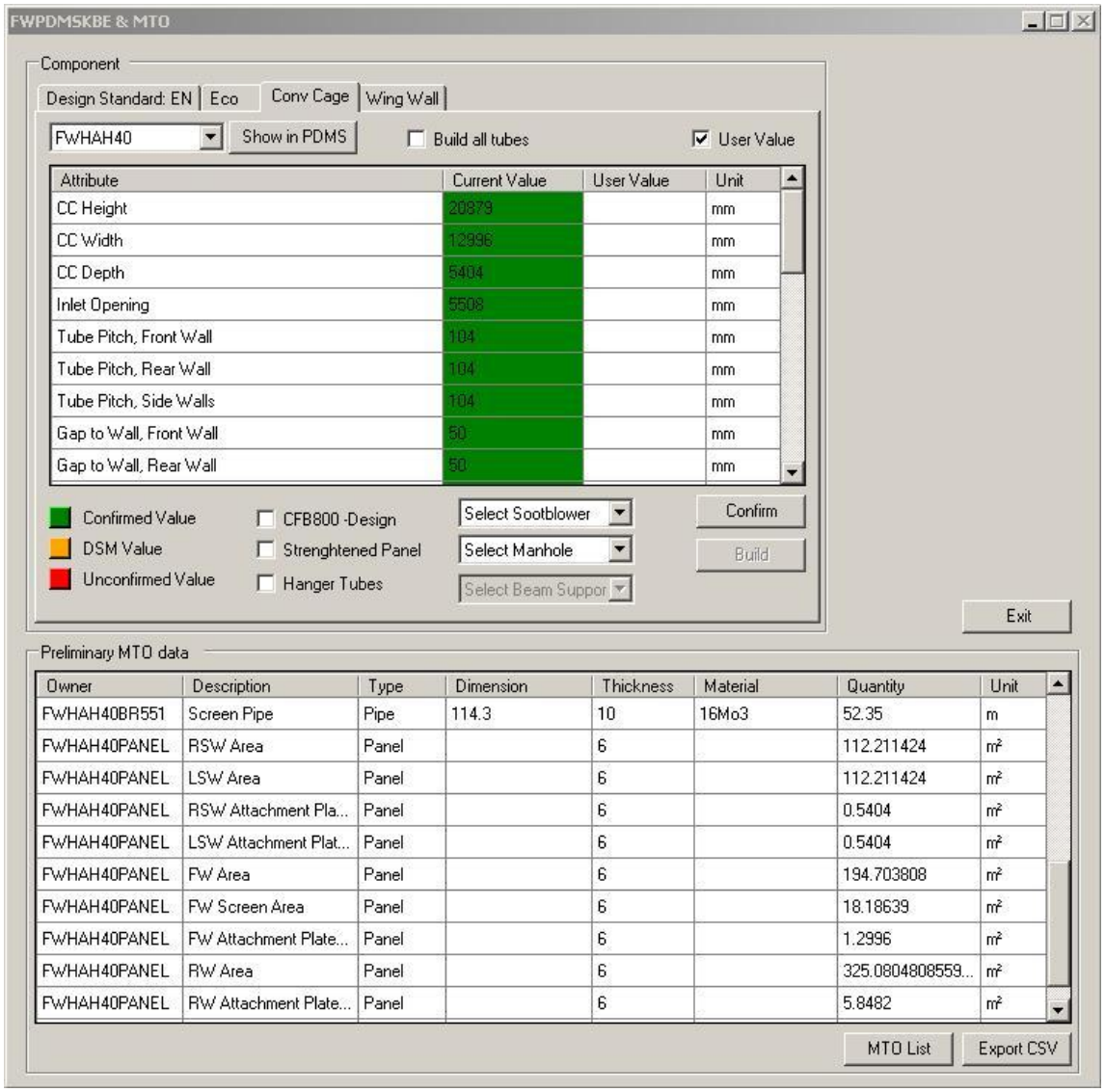
PDMS:stä voidaan hakea halutun osan, kuten vaikkapa putken tai kammion mitat tarkasti. Tämä helpottaa varsinkin putkien pituuksien selville saamista, sillä putkien pituuksia voi olla vaikea hahmottaa, koska ne sisältävät taivutuksia. Saadut arvot ohjelma syöttää Exceliin, jossa ne menevät materiaalimäärälaskijoitten laskukaavoihin. Tämän jälkeen ohjelma hakee saadut tulokset Excelistä ja ne näkyvät ohjelman näytössä. Tulokset voidaan tallentaa CSV (*comma-separated values*) -tiedostoksi ja avata Excelissä lähempää tarkastelua varten.

Ohjelman avulla materiaalimääriä saadaan testien perusteella laskettua noin ±5 % tarkkuudella. Tämä helpottaa tarjouslaskentaa. Lisäksi kattilan osista voidaan esittää karkeita mallinnuksia jo tarjousvaiheessa, sillä mallinnusten tekeminen ohjelmalla on helppoa.

## 4.2 Materiaalimäärälaskentaohjelman käyttöliittymä

Materiaalimäärälaskentaohjelman käyttöliittymänä on Windows-ikkuna, joka on jaettu kehyksillä kahteen osaan (kuva 6). Käyttöliittymä käyttää Windowsin oletusteemaa ulkoasunaan.

Ylempi kehys koostuu välilehdistä, jotka sisältävät välilehdestä riippuen asetuksia komponenttien mallintamista varten. Valittu välilehti määrittää myös mallinnettavan komponentin. Komponentin mallinnusarvot ovat sisällytetty tieto ruudukkoon, jossa arvot ovat selkeästi esillä. Mallinnusarvojen tieto ruudukko sisältää merkki estoja ja sallii vain oikeanlaisten merkkien syöttämisen sarakkeisiin. Välilehdet sisältävät kontrollit asetusten muuttamista varten komponenttikohtaisesti. Asetuksia varten on valintaruutuja ja pudotusvalikoita riippuen määritetäänkö vain jonkin asetuksen päällä olo vai valitaanko useasta vaihtoehtoisesta asetuksesta. Eri toimintojen aktivointi tapahtuu painonappeja käyttämällä.

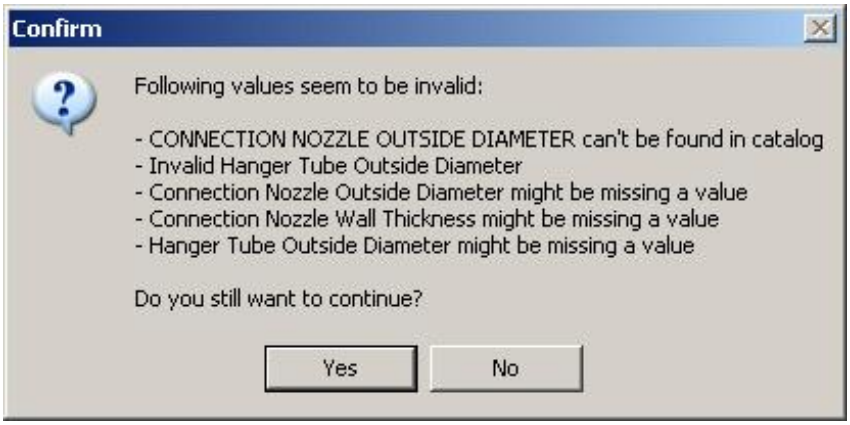



KUVA 6. Materiaalimäärälaskentaohjelman käyttöliittymä.

Alemmassa kehyksessä on tieto ruudukko Excelistä haettavia materiaalimääriä varten, sekä painonapit, joilla materiaalimääränäkymää hallitaan.

Värejä käyttöliittymässä hyödynnetään kuvaamaan arvojen oikeellisuutta. Arvon ollessa kelvollinen sen tieto ruudukko lokero värjätään vihreäksi ja jos arvo on virheellinen, niin lokero värjätään punaiseksi. Mikäli syötetty arvo on ollut virheellinen ja tilalle on haettu uusi toimiva arvo DSM-suunnitteluohjeista, värjätään arvon tieto ruudukko lokero oranssilla. Väreille on myös selitykset käyttöliittymän ylemmän kehyksen alalaidassa.

Ohjelman antamat virheilmoitukset pyritään pitämään mahdollisimman kuvaavina. Mikäli ohjelma havaitsee mallinnusarvoja tarkastettaessa niissä olevan vikaa, annetaan virheilmoitus, joka kertoo hyvinkin tarkasti mikä missäkin arvossa on vialla (kuva 7). Virheilmoitus kertoo esimerkiksi jos arvoa ei ole ollenkaan syötetty, jos sitä ei löydetä katalogista tai mikäli arvo yksinkertaisesti on viallinen.




KUVA 7. Esimerkki ohjelman virheilmoituksesta.

## 4.3 Materiaalimäärälaskentaohjelman käyttäminen

Ohjelman käynnistyttyä ensimmäisenä avautuu välilehti nimeltä Boiler Type. Tästä välilehdestä ei voi edetä komponenttien välilehdille ennen kuin valitaan mitä suunnitelma standardia käytetään. Lisäksi tällä välilehdellä voidaan valita, tuleeko mallista ”inline” vai ”over the top” sekä käytetäänkö offline-tilaa vai ei. Tämän jälkeen ruudulle ilmestyvät välilehdet jokaiselle komponentille, jotka ohjelmalla voidaan mallintaa.

Välilehden valinta määrittää samalla sen, mitä kattilanosaa ryhdytään mallintamaan. Kun jokin välilehdistä valitaan, ilmestyy ylemmälle välilehdelle komponentin mallintamiseen tarvittavat kontrollit. Kun pudotusvalikosta valitaan komponentti sen nimen mukaan, ilmestyvät komponentin arvot tietoruudukkoon. Offline-tilassa valitun komponenttityypin komponentteja voidaan itse lisätä listaan joko täysin uusilla arvoilla

tai kopioimalla arvot jo olemassa olevasta komponentista. Ohjelma hakee komponentille sen osien mitat tietokannasta, ja mikäli ollaan offline-tilassa, arvot haetaan XML-tiedostosta. Arvoja voidaan muuttaa painamalla user value valintaruutu päälle. Kun valintaruutu painetaan päälle, ilmestyy tieto ruudukkoon uusi sarake, johon halutut arvot voidaan syöttää. Offline-tilassa arvot voidaan kirjoittaa suoraan XML:stä haettujen arvojen päälle. Mallinnusarvojen lisäksi eri komponenteilla on erilaisia lisävaihtoehtoja, kuten vaikkapa materiaalivaihdoksia ja virtaussuunnan muutoksia. Nämä voidaan valita valintaruutuja rastittamalla. Lisäksi joissain komponenteissa voidaan valita mallinnetaanko kaikki putket vai ainoastaan malliksi muutamia. Jotkut komponentit voivat sisältää jopa tuhansia putkien taivutuksia, joten ohjelma varoittaa, että tietokone saattaa lakata vastaamasta pitkäksikin aikaa, mikäli kaikki putket halutaan mallintaa.

Haluttujen arvojen syöttämisen ja haluttujen valintojen tekemisen jälkeen painetaan ”Confirm” -painonappia. Tämä tarkistaa, ovatko arvot kunnossa mallintamista varten. Mikäli jotain huomautettavaa löytyy, ilmestyy ruudulle ilmoitus, missä arvoissa tai valinnoissa huomautettavaa oli, sekä viallisen arvon ruutu väritetään tieto ruudukosta huomautusta vastaavalla värillä. Kun arvot ovat kunnossa, voidaan painaa ”Build” -painonappia, joka mallintaa komponentin annetuilla arvoilla ja tehdyillä valinnoilla. Tämän jälkeen 3D-mallia voidaan tarkastella PDMS-ohjelmassa.

Ohjelman alemmassa kehyksessä on ”MTO List” -painonappi. Tätä painamalla ohjelma lähettää mallista saadut arvot taustalla toimivaan Excel-taulukkoon ja hakee sieltä saadut laskutulokset ohjelmaan. Lasketut materiaalimäärät tulevat näkyviin ohjelman alemmassa kehyksessä olevaan tieto ruudukkoon. Näkymään tulee eri osien nimet, niiden mitat, kuinka monta kappaletta tai metriä osaa mallissa on sekä kuinka paljon osa painaa. ”Export CSV” -painonapilla voidaan tallentaa alemman kehyksen näkymästä CSV (*comma-separated values)* -tiedosto, joka voidaan avata Excelissä arvojen tarkastelua varten.

# 5 MATERIAALIMÄÄRÄLASKENTAOHJELMAN TOTEUTUS

Materiaalimäärälaskentaohjelma on toteutettu C#-ohjelmointikielellä. Ohjelmoinnissa on käytetty Microsoftin Visual Studio 2005 -ohjelmaa.

Jokainen mallinnettava komponentti koostuu kahdesta luokasta. Näistä toinen sisältää mallinnukseen tarvittavat koodit ja laskut ja toinen arvojen hakemisen mallinnusarvojen luokasta, arvojen tarkistamisen virheellisten arvojen varalta sekä materiaalimäärä-laskentaan liittyvät funktiot. Mallinnukseen liittyvistä yleisistä funktioista, kuten kammioiden sekä putkien ja niiden taivutusten mallintamisesta on tehty omat luokkansa, joita käytetään jokaisen komponentin mallintamisessa.

Ohjelma hyödyntää Exceliä, jonka tuotantotaloudenharjoittelijat ovat koonneet. Ohjelma lähettää Exceliin arvot mallia ja laskuja hyödyntäen, ohjelmaan syötettyjen arvojen perusteella. Excelissä arvoilla lasketaan materiaalimäärät, jonka jälkeen ohjelma hakee lasketut materiaalimäärät ohjelman materiaalimääränäkymään. Materiaalimäärät voidaan tallentaa CSV (*comma-separated values)* -tiedostoksi.

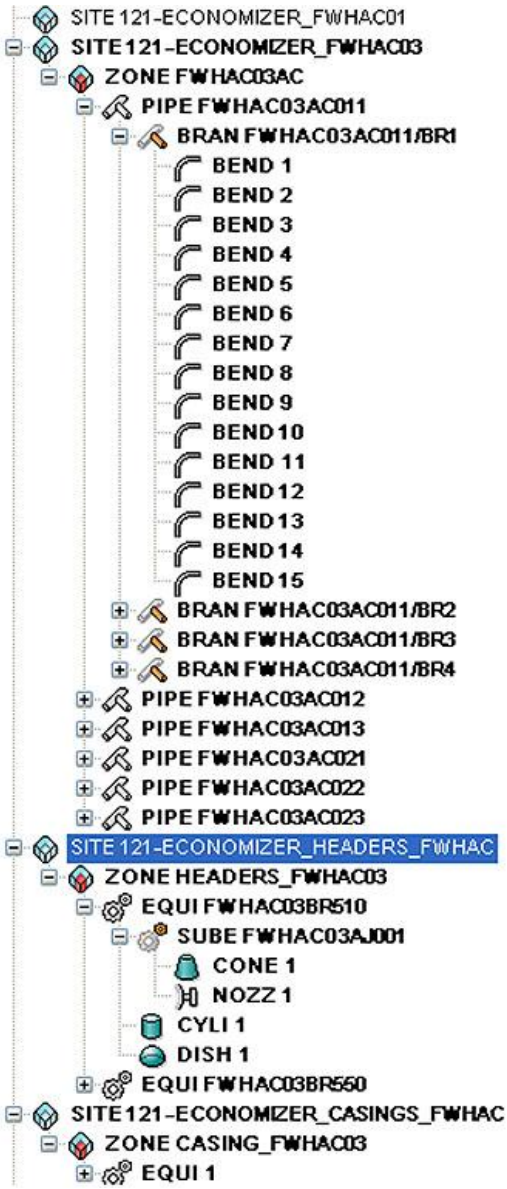
Ohjelmaan on lisätty viitteet Avevan frameworkiin, tietokantaan, geometriaan, grafiikkaan, implementaatioon ja työkaluihin. Näiden avulla voidaan käyttää Avevan PDMS-funktioita, joiden avulla ohjelma pystyy käyttämään PDMS:ää.

Ohjelma tuottaa dll (*Dynamic-link library*) -tiedoston, joka siirretään PDMS:n tiedostoihin ja asetetaan se latautumaan PDMS:n käynnistyessä. Sen jälkeen se lisätään PDMS:n työkaluriviin, josta ohjelma voidaan avata. Materiaalimäärälaskentaohjelma toimii PDMS:n aliohjelmana.

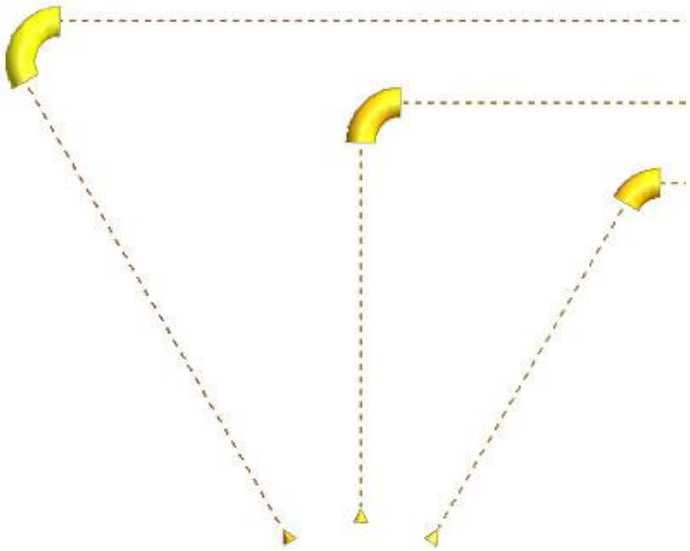
## 5.1 Mallinnuksen toteutus

AVEVA PDMS mallintaminen perustuu kolmiulotteisen koordinaatiston koordinaattien laskentaan. Laskentoihin käytetään trigonometrisiä laskuja, joita tarvitaankin erittäin paljon. Lähes jokainen koordinaattipiste on laskettu trigonometriaa hyödyntäen. Laskentoja varten ohjelmassa on oma luokkansa, jossa on tarvittavat funktiot trigonometrisiin laskuihin. Koordinaattipisteen lisäksi jokainen objekti tarvitsee orientaation, se määrittää objektin suunnan mallissa. Orientaatiota muuttamalla saadaan objekteja käänneltyä mallissa. Orientaatioiden luomista varten on myös oma luokka, jonka avulla voidaan muodostaa orientaation vaatimia string-jonoja.

AVEVA PDMS ohjelmalla luoduissa malleissa on oma hierarkiansa (kuva 8). Aivan aluksi täytyy luoda SITE, jonka alle mallia lähdetään luomaan. Materiaalimäärälaskentaohjelmalla luoduilla kattilan osilla on kaikilla oma SITE, jonka alle osat mallinnetaan. SITE:n alle luodaan ZONE, jonka alle voidaan luoda itse mallinnuksessa käytettäviä asioita. Niitä käytetään esimerkiksi kun halutaan jaotella jokainen kattilan osan seinämä omalle ZONE:lleen. Niiden alle voidaan luoda PIPE tai EQUI. PIPE:n alle luodaan putkistot. Jokaiselle putkelle luodaan ensiksi BRAN, jolle asetetaan pää ja häntä sijainnit. Pää määrittää mistä putki alkaa ja häntä sen, minne putki loppuu. Päälle ja hännälle asetetaan suunnat, johon putkea lähdetään luomaan. BRAN:in alle voidaan luoda BEND, eli mutka. Mutkalle asetetaan koordinaatti piste, kulma, säde, sekä orientaatio. Orientaatio määrää kappaleiden asettelun. Mikäli putken pää ja häntä eivät mutkien avulla tai ilman kohtaa, piirtyy päästä häntään katkoviiva (kuva 9). Jos putken pää ja häntä kohtaavat, piirtyy putki mutkineen PDMS:n mallinnus ruudulle. EQUI:n alle voidaan luoda erilaisia kappaleita, kuten sylintereitä, laatikoita ja puolipalloja. Jos EQUI:ta halutaan jakaa vielä useampaan osaan, voidaan sen alle luoda SUBE, jonka alle voidaan luoda samoja asioita kuin EQUI:n. AVEVA PDMS:n hierarkia auttaa löytämään tietyn kappaleen arvot helposti. Jos esimerkiksi kattilan osan eri seinämät on luotu omille ZONE:illeen, on mallista helppo tarkastella vaikkapa vain tiettyä seinämää kerrallaan.




KUVA 8. AVEVA PDMS hierarkia.



KUVA 9. Esimerkki AVEVA PDMS:llä mallinnetuista putkista, jotka eivät kohtaa.

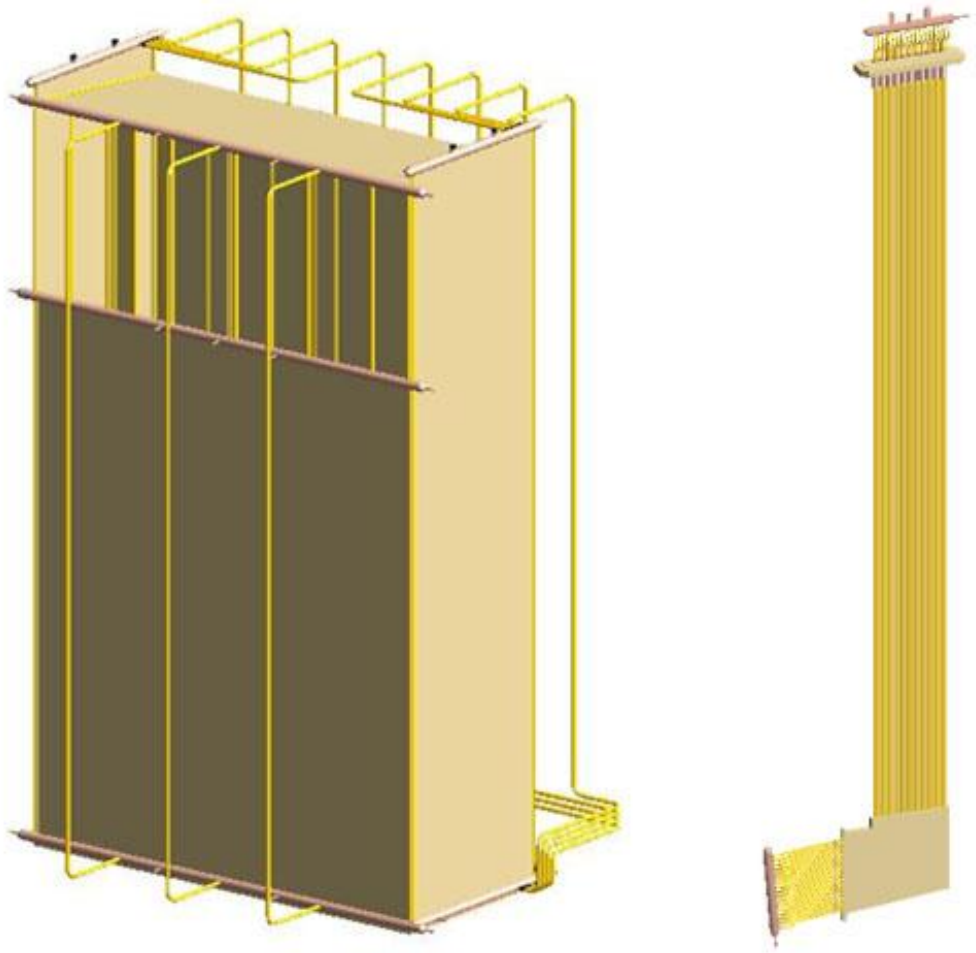
Materiaalimäärälaskentaohjelmalla tuotettavat mallit koostuvat lähinnä putkista, kammioista ja paneeleista. Näistä jokaiselle on omat luokkansa, jotka sisältävät tarvittavat funktiot objektien luomiseen ja hallinnointiin. Esimerkiksi putkiluokalla voidaan luoda putkille oma zone, ja zonen alle pipe, jonka alle putkia voidaan ruveta mallintamaan. Se sisältää myös funktion putkien taivutusta varten (kuva 10). Lisäksi on funktio, jolla voidaan etsiä jokin tietty pipe PDMS-mallista nimen perusteella, sekä funktio, jolla voidaan hakea putki spec PDMS-katalogista. Vastaavanlaiset funktiot löytyvät myös kammio- ja paneeliluokista.

```
public void buildBend(int bendCounter, double bendRad, double bendAngle, double bendXpos,
                      double bendYpos, double bendZpos, string bendOri, string compSpecRef,
                      string tubeSpec)
{
    if (PDMSControl.IsPDMSRunning())
    {
        bend = bran.Create(bendCounter, DbElementType.GetElementType("BEND"));
        bend.SetAttribute(DbAttributeInstance.SPRE, DbElement.GetElement(compSpecRef));
        bend.SetAttribute(DbAttributeInstance.LSTU, DbElement.GetElement(tubeSpec));
        bend.SetAttribute(DbAttributeInstance.SHOP, true);
        bend.SetAttribute(DbAttributeInstance.RADI, bendRad);
        bend.SetAttribute(DbAttributeInstance.ANGL, bendAngle);
        bend.SetAttribute(DbAttributeInstance.ORIL, true);
        bend.SetAttribute(DbAttributeInstance.POSI, true);
        bend.SetAttribute(DbAttributeInstance.POS, Position.Create(bendXpos, bendYpos, bendZpos));
        bend.SetAttribute(DbAttributeInstance.ORI, Aveva.Pdms.Geometry.Orientation.Create(bendOri));
    }
}
```

KUVA 10. Esimerkki putkiluokan funktiosta, joka mallintaa PDMS:ään kulman putkeen. Funktioon syötetään kulman numero, kulman säteen pituus, kulma asteina, koordinaatiston X-kohta, Y-kohta, Z-kohta, orientaatio, Spec referenssi ja putki spekki.

Jokaiselle materiaalimääräohjelmalla mallinnettavalle komponentille on oma mallinnusluokkansa. Nämä mallinnusluokat kutsuvat funktioita eri objektien mallinnusluokista luodakseen kokonaisen 3D-mallinnuksen jostakin CFB-yksikön komponentista kuten vaikkapa Convection Cagesta tai Wing Wallista (kuva 11).

Mallinnus on tietokoneelle melko raskasta, sillä ohjelman täytyy laskea jokaiselle objektille koordinaattipiste. Siksi ohjelma mallintaa vain materiaalilaskennan ja layout suunnittelun kannalta tarpeelliset osat. Silti mallinnettavia putkia saattaa olla satoja ja jokainen putki saattaa sisältää useita mutkia, joille kaikille ohjelma laskee oman koordinaattipisteen. Tämän vuoksi mallinnusluokissa on oletuksena ehdot, että putkistoista mallinnetaan vain muutama esimerkiksi siitä, miten putkistot osassa menisivät. Jos halutaan mallintaa kaikki putket, se täytyy määrittää ohjelman asetuksista erikseen.



KUVA 11. Materiaalimäärälaskentaohjelman tuottamat PDMS mallinnukset. Vasemmalla: Convection Cage. Oikealla: Wing Wall.

Haasteellisinta materiaalimäärälaskentaohjelman mallinnusten kehityksessä on orientaatioiden laskeminen. Mikäli esimerkiksi putken mutka on kahteen eri suuntaan kallellaan, voi orientaation muodostumisen hahmottaminen olla melko hankalaa. Orientaatio määritetään käyttämällä X, Y ja Z suuntia sekä N, S, W ja E, jotka tulevat ilmansuunnista. Lisäksi käytetään kulmia, mikäli objekti on kallellaan. Loppujen lopuksi orientaatio voi olla luokkaa: "X is E 38,23 N 23,12 D and Z is N 85 U", josta kaikki kulmat ovat trigonometrialla laskettuja ja muuttuvat sen mukaan, minkä kokoisena osat mallinnetaan.

## 5.2 Mallinnusarvojen käsittely

Kun materiaalimäärälaskentaohjelmaa käytetään normaalisti, mallinnukseen tarvittavat arvot komponenteille on tallennettu PDMS:n tietokantaan. Ohjelmassa haetaan halutun komponentin arvot ja sijoitetaan ne DimensionValue nimiseen kokoelmaan. Arvoille annetaan luokan alussa nimet staattisina string-muuttujina. Jokaiselle tarvittavalle arvolle annetaan oma nimensä. Sen jälkeen siihen sijoitetaan sen nimi PDMS-ohjelmassa, nimi materiaalimäärälaskentaohjelmassa, oletusarvo, käyttäjän syöttämä arvo, yksikkö, tieto siitä, onko arvo sallittu boolean muuttujana, tieto onko arvosta käytössä dsm-arvo sekä arvon omistaja tietokannassa. Jotkut tietokantaan syötetyistä oletusarvoista ovat peräisin CI-sheeteistä. CI-sheetit sisältävät painerungon päämittoja, jotka on määritelty lämpölaskennan mukaan.

Kun materiaalimäärälaskentaohjelma ei saa yhteyttä schematic-tietokantaan, voidaan käyttää offline-tilaa. Se aktivoidaan valitsemalla Offline Mode checkbox materiaalimäärälaskentaohjelman ensimmäisellä välilehdellä. Kun offline-tila on käytössä, arvot haetaan ja tallennetaan XML-tiedostoina tietokannan sijaan. Offline-tilassa arvot haetaan DimensionValue-kokoelman sijaan OfflineValue-kokoelmaan. OfflineValue-kokoelman ero DimensionValue-kokoelmaan nähden on se, että siinä ei ole PDMS:ään liittyviä arvoja, kuten arvon omistajaa tietokannassa. Kokoelmassa on mallinnusarvoille vain yksi sijoituspaikka kahden sijaan, sillä offline-tilassa oletusarvoja ei ole, vaan käytetään ainoastaan käyttäjän ohjelmaan syötettyjä arvoja.

Osa mallinnukseen tarvituista arvoista määräytyy DSM-suunnitteluohjeiden mukaisesti. DSM-suunnitteluohjeet löytyvät ohjelman käyttämästä Excel-tiedostosta. Tällaisia arvoja ovat esimerkiksi putkien taivutussäteet, jotka määräytyvät putken halkaisijan mukaan.

## 5.3 Excelin hyödyntäminen ohjelmassa

Materiaalimäärälaskentaohjelma hyödyntää Excel-taulukkolaskentaa kahdella tavalla. Mallinnukseen tarvittavat DSM-suunnitteluohjeiden mukaiset arvot haetaan Excelistä ja materiaalimäärien laskenta tapahtuu Excelissä. Excel-taulukon ja sen sisältämät materiaalimäärälaskennat ovat tehneet tuotantotaloutta opiskelevat työharjoittelijat.

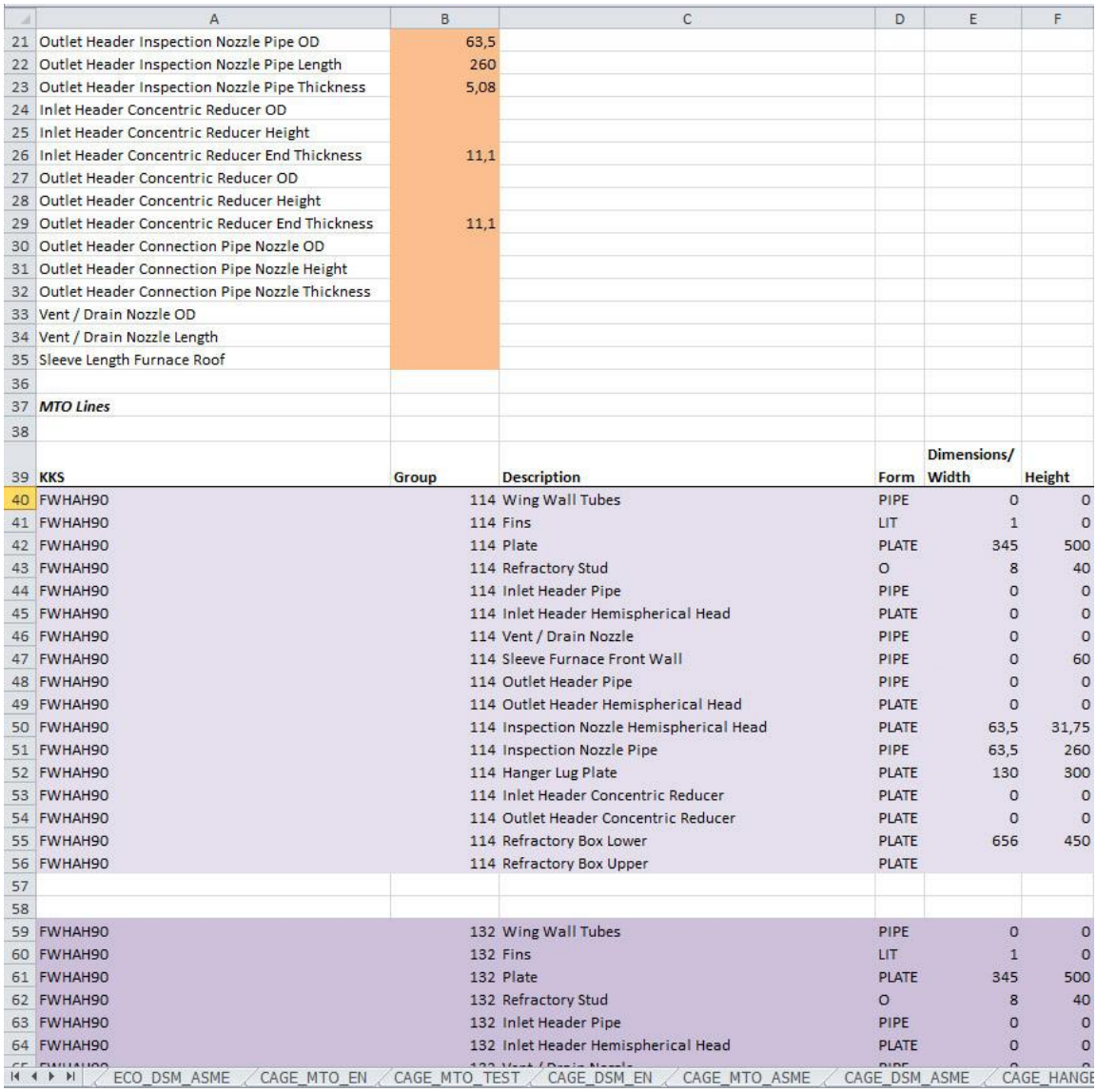
Ohjelmassa on luokka, joka sisältää kaikki Excelin hyödyntämiseen tarvittavat funktiot. Luokka sisältää funktiot, joiden avulla Excel-tiedosto avataan ohjelman taustalle ja funktion, joka sulkee Excelin. Lisäksi se sisältää funktiot, joiden avulla voidaan lukea tai kirjoittaa haluttu tieto kertomalla tiedon sarake, rivi ja Excel-taulukon nimi.

Jokaisen komponentin arvojenhallintaluokasta löytyy funktio DSM-ohjeiden käyttöön. DSM-ohjeiden mukaan määräytyvät esimerkiksi putkien taivutussäteet (kuva 12). Ohjelmaan syötetään putkien halkaisija, jonka avulla ohjelma hakee DSM-suunnitteluohjeista sopivan taivutussäteen. Koodissa määritetään miltä riviltä esimerkiksi putkien halkaisijat löytyvät. Sitten toistetaan rivi riviltä, kunnes syötetyllä putken halkaisijalla löydetään taivutussäde, tai kunnes vertailurivit loppuvat.

| B | C | D | E |
|---|---|---|---|
| **Bend radius** | | | |
| | | | |
| | | | |
| | Tube O.D | R | R/D |
| | 26,9 | 50 | 1,9 |
| | 31,8 | 60 | 1,9 |
| | 33,7 | 55 | 1,6 |
| | 38 | 60 | 1,6 |
| | 42,4 | 80 | 1,9 |
| | 44,5 | 75 | 1,7 |
| | 48,3 | 80 | 1,7 |
| | 51 | 76,5 | 1,5 |
| | 57 | 100 | 1,8 |
| | 60,3 | 120 | 2,0 |
| | 63,5 | 95 | 1,5 |
| | 76,1 | 130 | 1,7 |
| | 88,9 | 200 | 2,2 |
| | | | |

KUVA 12. Esimerkki Excelissä olevasta DSM-ohjeesta. (Kuvakaappaus, KBE Material Calculations 2010.)

Materiaalimäärälaskentoja varten ohjelma kirjoittaa laskentaan tarvittavat arvot Exceliin (kuva 13). Itse laskulausekkeet ovat valmiina Excelissä ja niissä arvot haetaan ohjelman syöttämiltä soluilta. Kun materiaalimäärät on laskettu, ohjelma hakee valmiit materiaalimäärätulokset Excelistä ja sijoittaa ne ohjelman alempaan tieto ruudukkoon, josta tuloksia voidaan tutkia. Tulokset voidaan myös tallentaa CSV (*Comma-Separated Values*) -tiedostoksi ja avata Excelissä.



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 21 | Outlet Header Inspection Nozzle Pipe OD | 63,5 | | | | |
| 22 | Outlet Header Inspection Nozzle Pipe Length | 260 | | | | |
| 23 | Outlet Header Inspection Nozzle Pipe Thickness | 5,08 | | | | |
| 24 | Inlet Header Concentric Reducer OD | | | | | |
| 25 | Inlet Header Concentric Reducer Height | | | | | |
| 26 | Inlet Header Concentric Reducer End Thickness | 11,1 | | | | |
| 27 | Outlet Header Concentric Reducer OD | | | | | |
| 28 | Outlet Header Concentric Reducer Height | | | | | |
| 29 | Outlet Header Concentric Reducer End Thickness | 11,1 | | | | |
| 30 | Outlet Header Connection Pipe Nozzle OD | | | | | |
| 31 | Outlet Header Connection Pipe Nozzle Height | | | | | |
| 32 | Outlet Header Connection Pipe Nozzle Thickness | | | | | |
| 33 | Vent / Drain Nozzle OD | | | | | |
| 34 | Vent / Drain Nozzle Length | | | | | |
| 35 | Sleeve Length Furnace Roof | | | | | |
| 36 | | | | | | |
| 37 | ***MTO Lines*** | | | | | |
| 38 | | | | | | |
| 39 | **KKS** | **Group** | **Description** | **Form** | **Dimensions/ Width** | **Height** |
| 40 | FWHAH90 | 114 | Wing Wall Tubes | PIPE | 0 | 0 |
| 41 | FWHAH90 | 114 | Fins | LIT | 1 | 0 |
| 42 | FWHAH90 | 114 | Plate | PLATE | 345 | 500 |
| 43 | FWHAH90 | 114 | Refractory Stud | O | 8 | 40 |
| 44 | FWHAH90 | 114 | Inlet Header Pipe | PIPE | 0 | 0 |
| 45 | FWHAH90 | 114 | Inlet Header Hemispherical Head | PLATE | 0 | 0 |
| 46 | FWHAH90 | 114 | Vent / Drain Nozzle | PIPE | 0 | 0 |
| 47 | FWHAH90 | 114 | Sleeve Furnace Front Wall | PIPE | 0 | 60 |
| 48 | FWHAH90 | 114 | Outlet Header Pipe | PIPE | 0 | 0 |
| 49 | FWHAH90 | 114 | Outlet Header Hemispherical Head | PLATE | 0 | 0 |
| 50 | FWHAH90 | 114 | Inspection Nozzle Hemispherical Head | PLATE | 63,5 | 31,75 |
| 51 | FWHAH90 | 114 | Inspection Nozzle Pipe | PIPE | 63,5 | 260 |
| 52 | FWHAH90 | 114 | Hanger Lug Plate | PLATE | 130 | 300 |
| 53 | FWHAH90 | 114 | Inlet Header Concentric Reducer | PLATE | 0 | 0 |
| 54 | FWHAH90 | 114 | Outlet Header Concentric Reducer | PLATE | 0 | 0 |
| 55 | FWHAH90 | 114 | Refractory Box Lower | PLATE | 656 | 450 |
| 56 | FWHAH90 | 114 | Refractory Box Upper | PLATE | | |
| 57 | | | | | | |
| 58 | | | | | | |
| 59 | FWHAH90 | 132 | Wing Wall Tubes | PIPE | 0 | 0 |
| 60 | FWHAH90 | 132 | Fins | LIT | 1 | 0 |
| 61 | FWHAH90 | 132 | Plate | PLATE | 345 | 500 |
| 62 | FWHAH90 | 132 | Refractory Stud | O | 8 | 40 |
| 63 | FWHAH90 | 132 | Inlet Header Pipe | PIPE | 0 | 0 |
| 64 | FWHAH90 | 132 | Inlet Header Hemispherical Head | PLATE | 0 | 0 |



KUVA 13. Materiaalimäärälaskentaohjelman taustalla toimivan Excelin toimintaperiaate. Ohjelma lähettää Exceliin arvot oranssilla värjättyihin soluihin, joiden avulla materiaalimäärät lasketaan. Sitten ohjelma hakee materiaalimäärät violetilla värjätyistä soluista ja esittää ne datagridviewissä. (Kuvakaappaus, KBE Material Calculations 2010.)

# 6 OFFLINE-TILA

Päättötyön aiheena oli materiaalimäärälaskentaohjelman offline-työskentelytilan kehittäminen. Normaalisti ohjelma käyttää Aveva PDMS:n tietokantaa mallinnusarvojen tallentamiseen. Jos schematic-tietokanta ei ole saatavilla tai sen sisältämät arvot ovat puutteellisia, voidaan käyttää offline-tilaa, jossa mallinnusarvot tallennetaan muualle. Offline-tila voi olla hyödyllinen, jos esimerkiksi halutaan käyttää materiaalimäärälaskentaohjelmaa kannettavalla tietokoneella yrityksen ulkopuolella.

## 6.1 Miksi XML?

Offline-tilan mallinnusarvojen tallennukseen valittiin XML, koska siinä tiedot kestävät kätevästi järjestyksessä. Jos mallinnusarvot olisi tallennettu vaikkapa normaalina tekstitiedostona, olisi offline-tilan mallinnusarvoja ollut vaikea tutkia tai muuttaa ilman ohjelman apua. XML-tiedostoa on helppo lukea ilman materiaalimääräohjelman tukea ja mallinnusarvoja voidaan muokata kätevästi myös ilman materiaalimäärälaskentaohjelmaa, suoraan XML-tiedostoa muokkaamalla.

Materiaalimäärälaskentaohjelman tuottamat XML-tiedostot koostuvat elementeistä, jotka sisältävät tiedon. Elementit ilmaistaan ”tagisyntaksilla” (esimerkiksi <Value>…</Value>). Jokainen elementti koostuu kolmenlaisesta tiedosta: nimestä, ominaisuuksista ja sisällöstä. Materiaalimäärälaskentaohjelman tuottamat elementit on nimetty kuvaamaan niiden sisältämää/sisältämiä tietoja (esimerkiksi <Unit> ja <Current_Value>).

XML:ssä voitaisiin käyttää attribuutteja tarkentamaan elementin sisältämää informaatiota tai esittää varsinainen informaatio, mutta niitä ei ole materiaalimäärälaskentaohjelman XML tiedostoissa käytetty (kuva 14). XML-tiedostot ovat selkeämmin luettavissa, kun kaikki tieto rakennetaan pelkillä elementeillä ilman attribuutteja.

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

KUVA 14. Yllä: Henkilön sukupuolen tieto toteutettuna XML attribuuttina. Alla: Samat tiedot toteutettuna pelkillä XML-elementeillä.

## 6.2 Offline-tilan toteutus

Offline-tilassa käytettävän XML:n tarvitsemat funktiot sisällytettiin yhteen luokkaan. Luokan nimi on OfflineConfig, joka tulee englannin kielen sanoista offline ja configuration, eli offline rakenne/konfiguraatio.

Kaikissa funktioissa, jotka kirjoittavat XML-tiedostoihin, on käytetty XmlWriter-metodia. XmlWriterille voidaan asettaa asetuksia, miten XML kirjoitetaan (kuva 15). Asetuksia varten luodaan uusi XmlWriterSettings, johon voidaan asettaa erilaisia määrittelyjä. Offline-tilan XML-tiedostojen kirjoitusasetuksiin on määritelty käytettäväksi sisennystä ja rivinvaihtoa. Kun esitellään uusi XmlWriter, syötetään sille tiedostopolku XML-tiedostoon ja XmlWriterSettings määräämät kirjoitusasetukset.

```
/// <summary>
/// Creates a new xml file for the values of a component.
/// </summary>
/// <param name="filePath">Filepath to the xml file (including the desired filename.xml).</param>
/// <param name="ComponentName">Name of the component which is added.</param>
public static void CreateComponentXML(string filePath, string ComponentName)
{
    XmlWriterSettings settings = new XmlWriterSettings();
    settings.Indent = true;
    settings.OmitXmlDeclaration = true;
    settings.NewLineOnAttributes = true;

    using (XmlWriter writer = XmlWriter.Create(filePath, settings))
    {
        writer.WriteStartDocument();
        writer.WriteStartElement(ComponentName);
        writer.WriteEndElement();
        writer.WriteEndDocument();
        writer.Close();
    }

}
```

KUVA 15. Esimerkki funktiosta, joka luo uuden XML-tiedoston mallinnusarvoja varten käyttämällä XmlWriterSettings ja XmlWriter -metodeja.

XML-tiedostot säilytetään projektikohtaisissa kansioissa. Tämän kansion löytämistä varten on OfflineConfig–luokassa oma funktionsa. Funktio hakee tiedostopolun käyttämällä System.Environment.GetEnvironmentVariable -toimintoa ja palauttaa sen.

Offline-arvoille luotiin oma kokoelmaluokka nimeltään OfflineValue. Tämä luokka toimii hyvin pitkälti kuten DimensionValue, joka on varattu tietokannasta haettaville arvoille. Samaa kokoelmaa ei kuitenkaan voida käyttää, sillä DimensionValue kokoelman mallinnusarvot sisältävät viitteitä tietokantaan, joita ei offline-arvoilla ole. Offline-arvoilla on muuttujina nimi, arvo, yksikkö sekä boolean-tieto siitä, onko arvo oikeanlainen. Mikäli

mallinnusarvossa havaitaan jotakin vikaa arvoja tarkistettaessa, muutetaan sen boolean-tieto arvon oikeellisuudesta epätodeksi, jolloin ohjelma osaa antaa virheilmoituksen. Offline-arvoja varten tarvitaan kaksi uutta funktiota jokaisen mallinnettavan komponentin mallinnusarvojen käsittelyluokkaan. Toinen funktio esittelee uuden OfflineValue-kokoelman ja hakee mallinnusarvot XML-tiedostosta kokoelmaan käyttämällä XmlNodeList-metodia ja foreach -toistoa. Toinen tarkistaa ovatko arvot kelvollisia mallintamista varten. Molemmat funktiot ovat hyvin samanlaisia kuin DimensionValue-kokoelman vastaavat.

Offline-tilassa voidaan luoda useita eri KKS-koodilla nimettyjä arvokokonaisuuksia samalle kattilanosalle, tämän takia komponenteille täytyy olla omat XML-tiedostonsa. Komponenttien omat XML-tiedostot nimetään kattilanosan mukaan, kuten esimerkiksi ”ConvectionCages.xml” tai ”Economizers.xml”. Nämä XML-dokumentit sisältävät kaikkien kyseisestä kattilanosasta olemassa olevat mallinnusarvo XML-dokumenttien nimet (kuva 16). Ne alkavat elementillä, joka nimetään kattilanosan mukaan, esimerkiksi <Economizers>. Sitä seuraa elementti <Component> jonka sisällä on elementti <Title> johon KKS-koodi kirjoitetaan. <Component> -elementit ovat dokumentissa selkeyttämässä sitä. Näiden XML-dokumenttien tiedostonimet on esitelty OfflineConfig luokan alussa static stringeinä.

```
- <Economizers>
  - <Component>
      <Title>FWHAC44</Title>
    </Component>
  - <Component>
      <Title>FWHACTEST55</Title>
    </Component>
  </Economizers>
```

KUVA 16. Esimerkki komponentin XML-dokumentista.

Materiaalimäärälaskentaohjelmassa mallinnukseen haluttu kattilanosan arvokokonaisuus valitaan käyttöliittymän ylälaidan pudotusvalikosta. OfflineConfig-luokka sisältää funktion, joka lukee valitun kattilanosan XML-tiedostosta arvokokonaisuudet ja sijoittaa niiden KKS-koodit comboboxiin valintaa varten. Funktiossa on käytetty XmlNodeList-metodia, jolla XML-elementit voidaan lukea järjestyksessä ja sijoittaa comboboxiin yksitellen käyttäen foreach –toistoa.

Itse mallinnusarvot tallennetaan annettujen KKS-koodien perusteella nimettyihin XML-tiedostoihin (kuva 17). Mallinnusarvo XML-dokumentit alkavat KKS-koodin mukaan

nimetyllä elementillä. Sen sisällä on useita <Value> -elementtejä, jotka sisältävät elementit <Attribute>, <Current_Value> ja <Unit>. <Attribute> -elementtiin talletetaan tieto siitä, mikä arvo on kyseessä, esimerkiksi Number of Tubes in a Wing Wall (putkien määrä Wing Wallissa). <Current_Value> -elementtiin talletetaan arvo, esimerkiksi 32. <Unit> -elementti sisältää tiedon siitä, mikä yksikkö arvolla on, esimerkiksi mm (millimetrit) tai pcs (kappalemäärä). OfflineConfig-luokka sisältää funktion, joka lukee datagridviewiin syötetyt arvot ja kirjoittaa ne oikeisiin elementteihinsä XML-tiedostossa. Mikäli arvon paikka on datagridviewissä jätetty tyhjäksi, kirjoittaa funktio tyhjän arvon tilalle nollan ongelmien välttämiseksi. Mikäli nolla arvoja esiintyy, varoittaa ohjelma niistä arvojen tarkistuksen yhteydessä.

```
- <FWHAH01>
  - <Value>
      <Attribute>Number of Tubes in a Wing Wall</Attribute>
      <Current_Value>32</Current_Value>
      <Unit>pcs</Unit>
    </Value>
  - <Value>
      <Attribute>Tube Pitch</Attribute>
      <Current_Value>56</Current_Value>
      <Unit>mm</Unit>
    </Value>
  - <Value>
      <Attribute>Gap to Furnace</Attribute>
      <Current_Value>762</Current_Value>
      <Unit>mm</Unit>
    </Value>
  - <Value>
      <Attribute>Additional Length, Bottom</Attribute>
      <Current_Value>1700</Current_Value>
      <Unit>mm</Unit>
    </Value>
  - <Value>
      <Attribute>Wing Wall Height</Attribute>
      <Current_Value>18000</Current_Value>
      <Unit>mm</Unit>
    </Value>
```

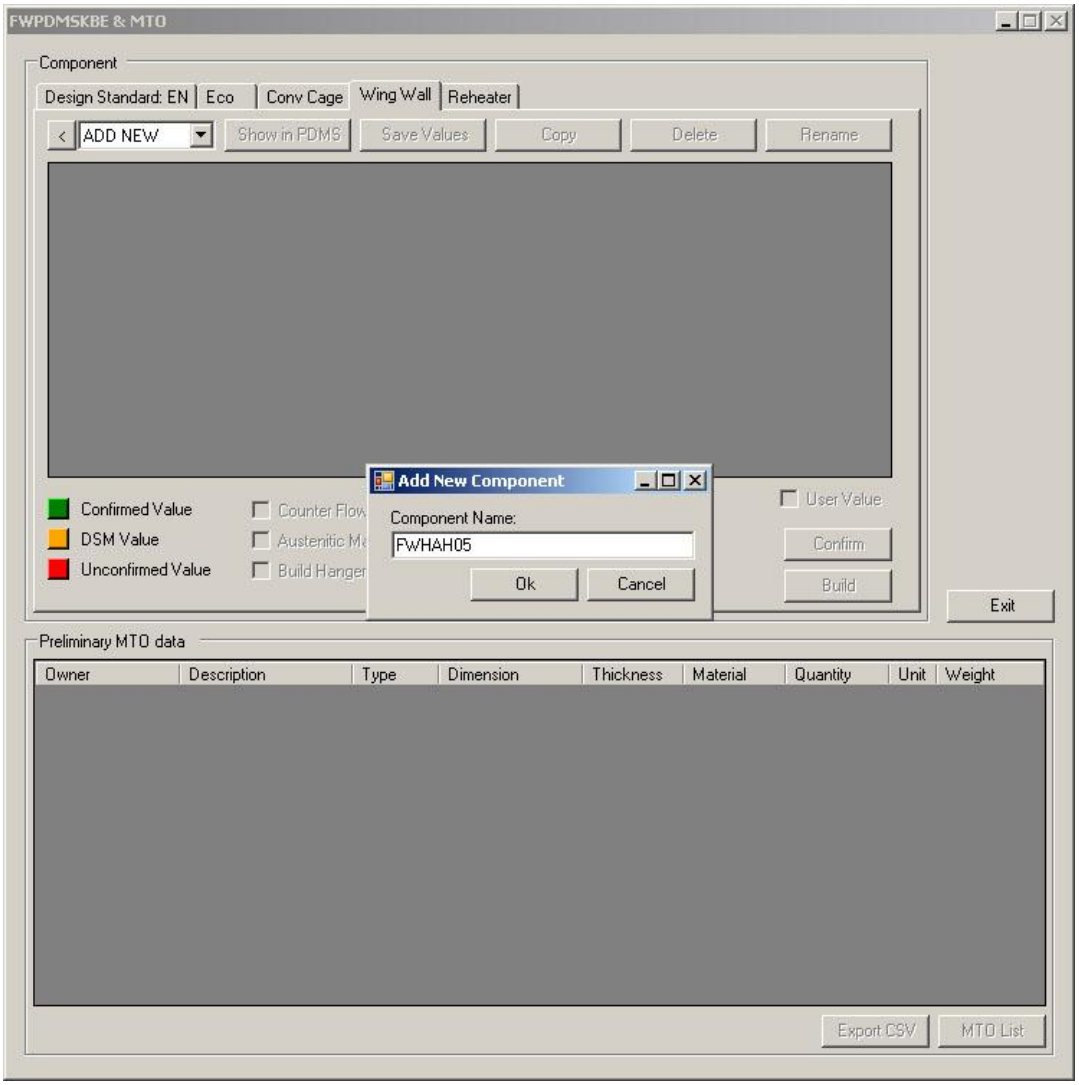
KUVA 17. Esimerkki mallinnusarvo XML-dokumentista.

## 6.3 Offline-tilan vaikutus käyttöliittymään

Offline-tila aktivoidaan käyttöliittymän ylemmän kehyksen yleisien asetusten välilehdeltä. Kun offline-tila on aktivoitu, aiheuttaa se joitakin muutoksia käyttöliittymään.

Komponenttivälilehtien ylälaidassa oleva komponentin valinta pudotusvalikko sisältää valinnan uuden komponentin luomiseen. Uuden komponentin lisäysvalinta avaa uuden ikkunan, johon luotavan komponentin nimikoodi syötetään (kuva 18). Ikkunassa on estoja, jotta nimeämiseen kiellettyjä merkkejä ei voida syöttää. Lopuksi ikkuna tarkistaa, ettei syötetty komponentin nimi ole muuten epäkelpo.

Jokaisen komponenttivälilehden ylälaitaan ilmestyy uusia painonappeja. Niillä hallitaan offline-tilan komponentteja, joita voidaan nimetä uudelleen, kopioida ja poistaa. Lisäksi on painonappi, jota painamalla voidaan tallentaa muutokset komponentin arvoihin.




KUVA 18. Ohjelman käyttöliittymä offline-tilassa. Päällimmäisenä uuden komponentin lisäys ikkuna.

# 7 YHTEENVETO

Opinnäytetyön lopputuloksena oli toimiva materiaalimäärälaskentaohjelman offline-tila, joka voidaan kytkeä ohjelmassa päälle haluttaessa. Kaikki offline-tilalle halutut ominaisuudet ja toiminnallisuudet saatiin toteutettua. Opinnäytetyön aikana offline-tilan toimintaa testattiin hieman, mutta laajempi testaus tehdään opinnäytetyön jälkeisenä aikana.

Tehdessäni tätä opinnäytetyötä minun täytyi tutustua moniin uusiin asioihin sovelluskehityksessä. Suurin näistä oli tutustuminen XML-merkintäkielen käyttöön, joka oli minulle melkein täysin uusi asia. Onneksi internet on täynnä erilaisia XML-käyttöoppaita ja neuvoja, joiden avulla pääsin melko nopeasti sisään XML:n käyttöön. Toisena uutena asiana tuli suurien ohjelmistojen kehittäminen. Koulussa ohjelmat koostuivat maksimissaan muutamasta luokasta ja parista sadasta riviä koodia. Materiaalimäärälaskentaohjelma sisälsi yli 40 luokkaa, jotka pitivät sisällään tuhansia rivejä koodia. Lisäksi tutuksi tulivat AVEVA PDMS–suunnitteluohjelma ja itse materiaalimäärälaskentaohjelma.

Suurempia ongelmia ei opinnäytetyötä tehdessä ilmentynyt, vaan kaikki sujui melko mutkattomasti. Ainoana ongelmana oli materiaalimäärälaskentaohjelmassa oleva mallinnusarvokokoelma, joka sisälsi paikat tietokantatiedoille. Kokoelma ei sallinut tietokantatietojen tyhjäksi jättämistä, joten offline-tilan mallinnusarvoille oli luotava oma kokoelmansa. Tämä johti työmäärän kasvuun, sillä kaikki kokoelmaan kohdistuvat funktiot oli luotava toiseen kertaan offline-kokoelmaa varten. Onneksi funktioita kuitenkin oli vain muutama.

Opinnäytetyön tekeminen oli hyvin mielenkiintoista ja uskon, että se opetti minua paljon. Varsinkin XML:n käyttäminen tuli opittua opinnäytetyön ansiosta. Myös olio-ohjelmoinnin hyödyt tulivat hyvin esille näin suuressa ohjelmassa. Koulun pienimuotoisissa harjoitustehtävissä olio-ohjelmoinnin etuja ei ole yhtä helppoa tajuta.

LÄHTEET

AVEVA 2008. PIPELINE Experiences from the plant and marine industries. [viitattu 2.5.2011] Saatavissa: http://www.aveva.com/pipeline/pipeline_2008.pdf

AVEVA. About Aveva. [viitattu 28.4.2011] Saatavissa: http://www.aveva.com/About_AVEVA.aspx

Baskar, R. 2011. History of C#? [viitattu 19.4.2011] Saatavissa: http://www.java-samples.com/showtutorial.php?tutorialid=1427

Baskar, R. 2011. Design Goals of C#. [viitattu 19.4.2011] Saatavissa: http://www.java-samples.com/showtutorial.php?tutorialid=1425

Baskar, R. 2011. Why it was named C#? [viitattu 19.4.2011] Saatavissa: http://www.java-samples.com/showtutorial.php?tutorialid=1426

Foster Wheeler. Pioneering CFB Technology. [viitattu 27.11.2010] Saatavissa: http://www.fwc.com/publications/pdf/CFBBrochure.pdf

Foster Wheeler Energia Oy 2009. Leading the world in CFB. Moniste. Ei kustannuspaikkaa eikä kustantajaa.

Foster Wheeler Energia Oy yritysinfo. [viitattu 20.1.2011] Saatavissa: http://www.fosterwheeler.fi/fi/Yritys/Default.aspx

Halttunen, M. 2005. C# Olio Ohjelmointi. Moniste. Ei kustannuspaikkaa eikä kustantajaa.

Horton, I. 1999. C++ Ohjelmoijan käsikirja. Helsinki: IT Press.

Laakkonen, A. & Walkama, P. 2004. Inside XML-Skeema. Helsinki: IT Press.

Layug, C. 2008. Versions and History of .NET Framework. [viitattu 19.4.2011] Saatavissa: http://www.articlealley.com/article_548402_4.html

Microsoft. Introduction to the C# Language and the .NET Framework. [viitattu 22.4.2011] Saatavissa: http://msdn.microsoft.com/library/z1zx9t92(VS.100).aspx

Skonnard, A. 2001. XML in .NET: .NET Framework XML Classes and C# Offer Simple, Scalable Data Manipulation. [viitattu 6.4.2011] Saatavissa: http://msdn.microsoft.com/en-us/magazine/cc302158.aspx

VTT. 2009. Leijukerrospoltto. [viitattu 28.4.2011] Saatavissa: https://www.tut.fi/pop/pap/suomi/monipoltt_kattilat/5_leijukattilat/frame.htm

Youssef, M. 2005. The .NET Framework. [viitattu 21.4.2011] Saatavissa: http://www.aspfree.com/c/a/C-Sharp/Introducing-CSharp-and-the-NET-Framework/1/

www.savonia.fi