



**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

# Web Based Customized Design

**Morten Benestad Moi**

Master of Science in Engineering and ICT

Submission date: June 2013

Supervisor: Terje Rølvåg, IPM

Co-supervisor: Ole Ivar Sivertsen, IPM  
Ole Skjølingstad, Summit as  
Qazi Sohail Ahmad, IPM

Norwegian University of Science and Technology  
Department of Engineering Design and Materials



**MASTEROPPGAVE VÅR 2013**  
**FOR**  
**STUD.TECHN. MORTEN BENESTAD MOI**

**WEBBASERT BRUKERTILPASSET DESIGN**  
**Web based customized design**

En ønsker å lage en generisk løsning hvor en kan utvikle en salgskonfigurator på en tynn klient hvor kunden/selger kan endre forhåndsoppsatte parametere, og hvor disse parameterne sendes til en NX sesjon som generere 3D modell og ønskede eksport formater (JT, 3D, DWG). Resultat sendes kunde.

En ser for oss en kombinasjon av PTS(Product Template Studio), noe automatisering av data opprettelse i NX og eksport av ønskede formater, samt en tilpasset webløsning som kan gi input til en NX sesjon. Når det gjelder selve CAD-parten (PTS) så bør ikke den være for komplisert, men brukes bare for å demonstrere løsningen. Dette er en løsning som burde være aktuell for mange bedrifter og produkter. Løsningen kan være rettet mot kundemarkedet eller internt i en bedrift. Oppgaven gjennomføres i samarbeid med Summit as i Oslo.

Opgaven omfatter:

1. Studere aktuelle formater for visualisering av NX-modeller
2. Finne ut hvordan NX kan ta inn data og automatisk oppdaterer og eksporterer modellen
3. Lage en modell som eventuelt kan brukes i Product Template Studio og som kan ta inn og endre på forhåndsoppsatte parametere
4. Lage generisk nettside som kan brukes som salgskonfigurator og som benytter seg av modellen og bruker NX til å lage en oppdatert, tilpasset modell
5. Hvis tiden tillater det, kontakte aktuelle bedrifter for å høre hva de synes om resultatet

Besvarelsen skal ha med signert oppgavetekst, og redigeres mest mulig som en forskningsrapport med et sammendrag på norsk og engelsk, konklusjon, litteraturliste, innholdsfortegnelse, etc. Ved utarbeidelse av teksten skal kandidaten legge vekt på å gjøre teksten oversiktlig og velskrevet. Med henblikk på lesning av besvarelsen er det viktig at de nødvendige henvisninger for korresponderende steder i tekst, tabeller og figurer anføres på

begge steder. Ved bedømmelse legges det stor vekt på at resultater er grundig bearbejdet, at de oppstilles tabellarisk og/eller grafisk på en oversiktlig måte og diskuteres utførlig.

Senest 3 uker etter oppgavestart skal et A3 ark som illustrerer arbeidet leveres inn. En mal for dette arket finnes på instituttets hjemmeside under menyen masteroppgave (<http://www.ntnu.no/ipm/masteroppgave>). Arket skal også oppdateres en uke før innlevering av masteroppgaven.

Besvarelsen skal leveres i elektronisk format via DAIM, NTNUs system for Digital arkivering og innlevering av masteroppgaver.

Kontaktpersoner:

Ole Skjølingstad, Summit as, Email: [skjolingstad@summit.no](mailto:skjolingstad@summit.no)

Qazi Sohail Ahmad, IPM/NTNU

Ole Ivar Sivertsen, IPM/NTNU



Torgeir Welo  
Instituttleder



Terje Rølvåg  
Faglærer



NTNU  
Norges teknisk-  
naturvitenskapelige universitet  
Institutt for produktutvikling  
og materialer

# Preface

This report was written as the 5th grade master thesis for the engineering and ICT (Ingeniørvitenskap & IKT) study program at the Norwegian University of Science and Technology (NTNU) spring 2013. After a three year specialization in Product Development and Materials Engineering at the mechanical engineering program the thesis includes elements from both study programs.

The thesis was given by Summit Systems with Ole Skjølingstad as the company advisor and was written at the Institute for Product Development and Materials with Terje Rølvåg as the main faculty advisor. Thanks for the help and guidance.

Morten Benestad Moi

# Abstract

This thesis studies the methods needed to create a web based application to remotely customize a CAD model. This includes customizing a CAD model by using a graphical user interface to be able to remotely control the inputs to and outputs from the model in NX, and to get the result sent back to the user.

Using CAD systems such as NX requires intensive training, is often a slow process and gives a lot of room for errors. An intuitive, simple user interface will eliminate the need for CAD training or CAD experience. The time used can be reduced by creating smart, parameterizable models that easily can be customized to the user's requirements. Built in rules in the model and the user interface can decrease possibility for errors and help maintain the design intent. By creating a web based solution the user does not need to have a CAD system installed or licensed on its system to update the model and changes can be made on the go.

The result of the thesis is a solution where the user can install an application that can connect to a server that can update a CAD model in NX and send the result back to the user. The method used in this solution is presented in the thesis.

# Sammendrag

Denne masteroppgaven studerer metodene som trengs for å lage en web basert løsning for å tilpasse en CAD modell. Dette inkluderer å tilpasse CAD modellen ved å bruke et brukergrensesnitt som kan fjernstyre innspill til- og eksporterte resultater fra modellen i NX, og sende resultatene tilbake til brukeren.

Å bruke CAD systemer som NX krever grundig opplæring, er en tidkrevende prosess og gir mange muligheter for feil. Et intuitivt og enkelt brukergrensesnitt vil fjerne behovet for CAD opplæring eller tidligere CAD erfaring. Tidsbruken i NX kan bli betraktelig redusert ved å lage smarte, parameteriserbare modeller som enkelt kan tilpasses til brukernes krav. Innebygde regler i modellen og brukergrensesnittet kan redusere muligheten for feil og hjelpe til å opprettholde designets hensikt. Ved å lage en web basert løsning trenger ikke brukeren ha et CAD system installert på sitt system og trenger ikke en egen lisens for å oppdatere modellen, og tilpasninger kan gjøres hvor og når som helst.

Resultatet av masteroppgaven er en løsning hvor en bruker kan installere en applikasjon som kan koble seg til en server som kan oppdatere en CAD modell i NX og sende resultatet tilbake til brukeren. Metoden som er brukt i løsningen presenteres i masteroppgaven.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Custom Board Design . . . . .	1
1.3	Objective . . . . .	6
1.4	Methodology . . . . .	6
1.5	Structure of Thesis . . . . .	7
1.6	Software . . . . .	10
1.6.1	NX 8.5 . . . . .	10
1.6.2	Microsoft Visual Studios 2008 . . . . .	10
1.6.3	JT2GO . . . . .	11
1.6.4	FTP Server . . . . .	11
1.6.5	Batch File . . . . .	11
<b>2</b>	<b>Software Requirements Specification</b>	<b>12</b>
2.1	Purpose . . . . .	12
2.2	Scope . . . . .	12
2.3	Definition of Terms . . . . .	13
2.4	Intended Use and Users . . . . .	14
2.5	Operating Environment . . . . .	14
2.6	Overall System Requirements . . . . .	14
<b>3</b>	<b>File Formats for Visualization</b>	<b>16</b>
3.1	Overview . . . . .	16
3.2	JT . . . . .	17
3.3	STEP . . . . .	18
3.4	3D PDF . . . . .	18
3.5	NX Part Files . . . . .	19



3.6	Chapter Discussion . . . . .	19
<b>4</b>	<b>Importing, Updating and Exporting in NX</b>	<b>20</b>
4.1	NX Open API . . . . .	20
4.2	Importing, Updating and Exporting Results in NX . . . . .	21
4.2.1	NX Open API Wizard . . . . .	21
4.2.2	Macro . . . . .	21
4.2.3	Journal . . . . .	22
4.3	Chapter Discussion . . . . .	23
<b>5</b>	<b>Process Description</b>	<b>24</b>
5.1	Overall Process Description . . . . .	24
5.2	NX . . . . .	27
5.2.1	Overview . . . . .	27
5.2.2	Modeling . . . . .	28
5.2.3	Run Journal . . . . .	29
5.2.4	Update Model . . . . .	29
5.2.5	Export Model . . . . .	29
5.3	Model . . . . .	29
5.3.1	Overview . . . . .	29
5.3.2	Create Model . . . . .	31
5.3.3	Name Expressions . . . . .	32
5.3.4	Create Design Rules . . . . .	34
5.3.5	Change Expressions . . . . .	35
5.3.6	Choose Export Format . . . . .	37
5.4	Journal . . . . .	38
5.4.1	Overview . . . . .	38
5.4.2	Record Journal . . . . .	40
5.4.3	Stop Recording Journal . . . . .	42
5.4.4	Open in Text Editor . . . . .	42
5.4.5	Locate Expressions and File Paths . . . . .	42
5.4.6	Running a Journal File . . . . .	43
5.5	Web Based Customizable Design (WBCD) . . . . .	45
5.5.1	Overview . . . . .	45
5.5.2	Import Journal . . . . .	47
5.5.3	Search for and Replace Expressions . . . . .	47
5.5.4	Write New Journal . . . . .	47
5.5.5	Open Model . . . . .	47

<b>6</b>	<b>Web Based Customizable Design Application</b>	<b>48</b>
6.1	Client-Server Architecture . . . . .	48
6.2	Client . . . . .	51
6.2.1	Graphical User Interface . . . . .	51
6.2.2	ISO Standards for Example Flanges . . . . .	54
6.2.3	Sequence List of Client Application . . . . .	57
6.3	Server . . . . .	60
6.3.1	Server User Interface . . . . .	60
6.3.2	Sequence List of Server Application . . . . .	61
6.4	Communication With Other Software . . . . .	66
<b>7</b>	<b>Using Web Based Customizable Design</b>	<b>68</b>
7.1	Benefits . . . . .	68
7.2	Limitations . . . . .	70
7.3	Implementation . . . . .	70
<b>8</b>	<b>Discussion</b>	<b>72</b>
<b>9</b>	<b>Conclusion and Further Work</b>	<b>80</b>
9.1	Conclusion . . . . .	80
9.2	Further Work . . . . .	81
	<b>Bibliography</b>	<b>82</b>
<b>A</b>	<b>Journal File</b>	<b>85</b>
<b>B</b>	<b>Web Based Customizable Design Application Source Code</b>	<b>89</b>
B.1	Client Application Source Code . . . . .	89
B.2	Client Application Designer Source Code . . . . .	96
B.3	Server Application Source Code . . . . .	102
<b>C</b>	<b>Batch File to Run Journal</b>	<b>107</b>
<b>D</b>	<b>NX Open API</b>	<b>108</b>
D.1	NX Open API . . . . .	108
D.2	NX Open API commands . . . . .	108
D.2.1	Imports . . . . .	108
D.2.2	UndoMarkId “SetUndoMark” . . . . .	109
D.2.3	Unit “FindObject” . . . . .	109
D.2.4	Unit “nullUnit” . . . . .	109

D.2.5	Expressions “CreateWithUnit” . . . . .	109
D.2.6	Expressions “EditWithUnit” . . . . .	110
D.2.7	Session “GetSession” . . . . .	110
D.2.8	Open File . . . . .	111
D.2.9	Save File . . . . .	112
D.2.10	Close File . . . . .	112
D.2.11	Export JT file . . . . .	113

**E Email from Infinity Innovations 117**

# List of Figures

1.1	Firewire's Custom Board Design Webpage . . . . .	3
1.2	Custom Board Design 3D PDF . . . . .	4
1.3	Custom Board Design queue message . . . . .	5
4.1	Macro code example . . . . .	22
4.2	Journal code example . . . . .	22
5.1	Flow chart of the process description . . . . .	25
5.2	Actions performed by developers and actions performed by application . . . . .	26
5.3	Flow chart of the process in NX . . . . .	28
5.4	Flow chart of the modeling in NX . . . . .	30
5.5	Flange model from NX . . . . .	31
5.6	Named expressions in sketch . . . . .	32
5.7	Named expressions in a list . . . . .	33
5.8	Design rules for hole placement . . . . .	34
5.9	Hole placement formula . . . . .	35
5.10	Changing the value of an expression . . . . .	36
5.11	Export JT files . . . . .	37
5.12	Flow chart of tasks for Journal . . . . .	39
5.13	Importing and exporting expressions in NX . . . . .	41
5.14	Exported expressions in Excel . . . . .	41
5.15	Changing the value of an expression example code . . . . .	42
5.16	Code showing file path of exported JT file . . . . .	43
5.17	Run journal in command prompt window . . . . .	44
5.18	Run journal by batch file . . . . .	44
5.19	Flow chart of tasks for Journal . . . . .	46

6.1	Client-server architecture illustration . . . . .	49
6.2	Flow chart of tasks for Client and Server . . . . .	50
6.3	Graphical User Interface of Web Based Customizable Design . . . . .	51
6.4	Selecting flange type in Web Based Customizable Design . . . . .	52
6.5	Design rules in Web Based Customizable Design example . . . . .	52
6.6	Adding range labels in GUI . . . . .	54
6.7	A&N Corporation ISO-LFB Socket Weld Flange . . . . .	55
6.8	Results from WBCD using ISO standards . . . . .	56
6.9	Sequence list of Client Application . . . . .	57
6.10	Customized flange examples . . . . .	59
6.11	Server Console Window . . . . .	60
6.12	Sequence List of Server Application . . . . .	61
6.13	Search string and replace string examples . . . . .	63
6.14	Batch file used to run Journal file . . . . .	64
6.15	Sequence diagram of communication between software . . . . .	66
7.1	Implementation changes . . . . .	71

# List of Tables

1.1	Digital attachments . . . . .	9
2.1	List of Abbreviations . . . . .	13
2.2	Overall system requirements . . . . .	15
3.1	File formats that NX can export . . . . .	16
3.2	Use of file formats . . . . .	17
5.1	List of important expressions in model . . . . .	35
5.2	List of available export formats from NX . . . . .	38
6.1	Pre-made LFB Socket Weld Flanges . . . . .	55
6.2	Parameters received from client . . . . .	63
8.1	Overall system requirements status . . . . .	74

# Chapter 1

## Introduction

### 1.1 Background

A surfboard company from Carlsbad, California, called Firewire Surfboards has created a website where customers can customize stock surfboards to their own specifications and models are automatically updated by NX. Details about this can be found in Section 1.2. The request from Summit Systems for this thesis is to create a solution to do the same thing to other models with a focus on the method of remotely updating a CAD model.

### 1.2 Custom Board Design

Firewire Surfboards is one of the top rated surfboard companies in the world. Recently they teamed up with ShapeLogic to create Custom Board Design (CBD), a web service where surfers can customize stock Firewire surfboards to their preferred specifications and view the result in a 3D PDF file within just a few minutes[1].

Firewire is known in the surfing community for their unique high-tech materials such as expanded polystyrene (EPS) foam, aerospace composites, epoxy resins, carbon rod suspensions and bamboo decks [2]. This gives lighter and more responsive surfboards but it offers challenges in the production phase. To produce a custom surfboard hours upon hours in CAD systems were needed by the company's engineers because the surfboards are produced using a Computer Numerical Control (CNC) machine [3]. To be able to compete for the

leading position in the market they need to offer custom shaped surfboards for their sponsored athletes and the rest of their customers. If every board requires several hours of CAD work the prices wouldn't be competitive and the backlog would build up due to the company's limited engineering staff so they needed to find another option [4].

Firewire teamed up with ShapeLogic who already had CAD expertise from creating parameterizable models of sets of golf clubs. They created complex CAD models of Firewire's stock surfboard models that allowed for customization within limits set by rules in the web page. The model also automatically created the CAM files that includes the CNC paths needed. The system they created is called "ShapeLogic Design-to-Order Live! for NX." The models were created using Product Template Studio (PTS) which allows predefined models to be parameterized with inputs from a web service to give the wanted results. CBD users can tweak the board's length, wide point, nose and tail width, and thickness to tailor the board to their preferences. The volume of the board, which is extremely important, yet hard to calculate, could now be calculated by NX before creating it which is a huge benefit [3].

In addition to the cost and efficiency benefits it also makes it possible to create repeatable designs. Repeatable designs have been a problem for custom made surfboards for a long time. Other surfboard companies usually use polyurethane (PU) foam cores for their surfboards. These can be machined to shapes that are up to 85 percent of the design and then they rely on a professional shaper to shape it to its specifications. This means that it is no guarantee that they will be able to recreate the exact shape if they want a replacement as surfboards does not last forever. Firewire is able to machine the pre-shaped boards to 97-98 percent of the design using CNC machines but the design process for the CAM production files was time consuming and needed to be improved [1][2].



**BASE OUTLINE**    SHOW    HIDE

LENGTH: 6'2"

NOSE WIDTH: 14.06

NOSE ADJ: .250

WIDTH: 20.87

WIDTH ADJ: .125

TAIL WIDTH: 16.44

TAIL ADJ: .125

THICKNESS: 2.75

THICKNESS ADJ: .125

Decimal	Fraction
.06	1/16
.12	1/8
.19	3/16
.25	1/4
.31	5/16
.38	3/8
.44	7/16
.50	1/2
.56	9/16
.62	5/8
.69	11/16
.75	3/4
.81	13/16
.88	7/8
.94	15/16
1.00	1

STOCK MODEL: DOMINATOR

LENGTH: 62"

MAIN WIDTH: 20.75

NOSE WIDTH: 13.69

TAIL WIDTH: 16.25

THICKNESS: 2.62

VOLUME: 38.1 L

THICKNESS: 2.75

WIDTH: 20.87

THICKNESS: 2.75

THICKNESS ADJ: .125

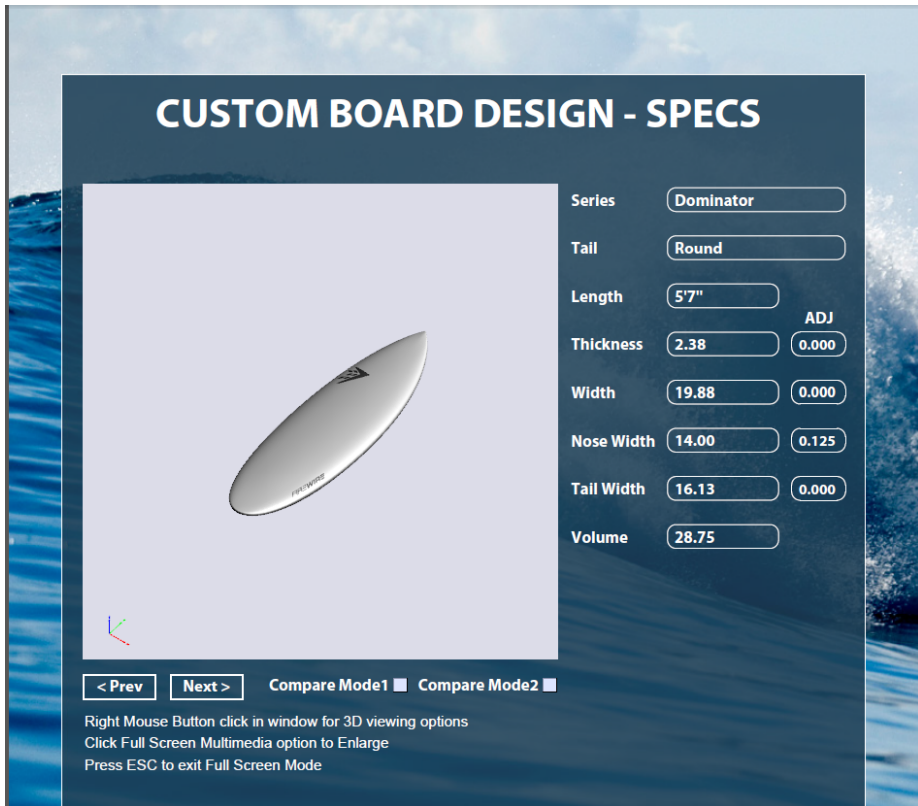
Source: [5]

Figure 1.1: Firewire's Custom Board Design Webpage

CBD uses ASP.NET scripting to create a dynamic website that can connect with NX through NX Open Application Programming Interface (API). The NX Open API is the critical link between the application in the browser interface and the 3D parametric models in NX. This allows the custom dimensions that customers input to be fed directly to the live CAD model[2].

To be able to visualize the design they use Anark Core server to automatically generate a 3D PDF of the custom board. The user can download the 3D PDF [4], which can be viewed in Adobe Acrobat Reader v9.0 and later versions, which use the Siemens PLM Software's JT Open data file format [6] to create

the 3D model. In the 3D PDF document the user is able to rotate the surfboard to see it from different angles. Figure 1.2 shows what the 3D PDF file looks like after customizing a board. The adjustments made from the original model are shown in the menu at the right side of the model in Figure 1.2.

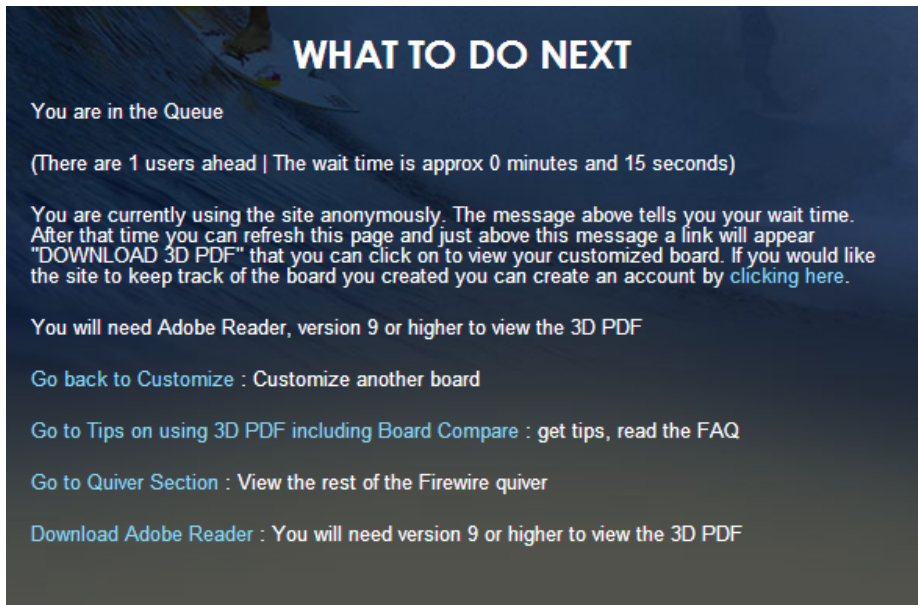


Source: [5]

Figure 1.2: Custom Board Design 3D PDF

For handling several customer clients at the same time CBD uses a queue system where the 3D model requests are put in a queue and it is estimated 15 second waiting time to handle each request. After testing the site a couple of times the actual time from pressing the button to request the 3D model to

the model being downloadable is about 1 minute and 30 seconds. This system makes it so that CBD only needs to use one NX license for handling all the requests. Figure 1.3 shows the message the user sees when waiting in the queue.



Source: [5]

Figure 1.3: Custom Board Design queue message

## 1.3 Objective

The main objective of this project is to create a method for creating a web based application to remotely customize a CAD model. This includes customizing a CAD model by using expressions and journaling in NX, to be able to remotely control the inputs to- and outputs from the model in NX, and to get the result sent back to the user.

## 1.4 Methodology

Because of a background from two different study programs, “mechanical engineering” and “engineering and ICT” this master thesis will have a multidisciplinary approach on the problem. From the engineering and ICT perspective the focus is on exploring the application programming interface of NX and creating the software application that can work over the internet and communicate with NX and other software such as JT2GO. From the mechanical engineering perspective is the method of simplifying tasks performed by the user, increase effectiveness and efficiency, and creating a description of the process based on product development.

Part of the references in this thesis is from software documentation. These references don’t have a listed author but will be referred to with the name of the company (e.g. Siemens PLM). Because there is little published research on file formats for visualization, blog posts, articles, software documentation and conversations with NX application engineers were used to reach a conclusion.

A course for introduction to Product Template Studio (PTS) [7] was completed to research how PTS works but this method was later discarded. The reason for discarding PTS was that changes made in PTS aren’t recorded in the journal file used to update the 3D model.

By attending a training course in NX held by Summit Systems in February 2013 it was possible to ask for advice and directions from the NX application engineers. This helped in deciding what direction to continue in concerning choice of visualization format and tool for customizing models.

After the training course in NX a model containing several different expressions and design rules was created. The focus of the model was to keep it simple but still have enough parameters and rules to control the design so that the end result can be demonstrated to the user.

The next step was to research what methods were available for giving inputs to NX and how to update models and export the results. Functions researched

in this part of the project were PTS, macro and journaling. It was decided to use journaling because it is a good way to generate code by performing actions in NX without getting all the unnecessary overhead information a macro file creates. This allows the user to record a journal, and perform a specific action and then look at the code generated to see what code is connected to the actions. . After deciding to use journaling the next step was to figure out how to communicate with the application programming interface of NX. This had to be done by programming in the NX Open API and at this point the development of the software application had started. With almost no previous knowledge of programming in Visual Basic except for a short lecture on it in a previous course at NTNU it was hard to decide what programming language to use for this thesis. After researching that the NX Open API is compatible with Java, C++ and .NET languages (VB and C#) but journaling only is compatible with VB and C# the choice was easier. After a quick consult with Ole Skjølingstad at Summit Systems it was decided to do the programming in VB. This was because Summit Systems often uses VB when doing projects for customers, so it would be beneficial for them to have the application programmed in VB.

When developing the application the research for how to create the connection between the server and client application was the main focus. This was done by watching tutorials and searching through developers forums. Other methods that were researched were how to import a journal file to the application, how to search in the journal file, how to upload and download files from a FTP server, and how to execute other programs like NX and JT2GO from the application.

When developing the server application a sample code from a “Multi Threaded Server Socket Programming in VB.net” tutorial [8] was used as a base to handle multiple incoming connections from different clients.

## 1.5 Structure of Thesis

**Chapter 1** starts with an introduction to- and the background for the problem including a detailed description of CBD.

**Chapter 2** goes through the software requirements specification which describes the scope, what is expected, and the requirements of the application.

**Chapter 3** presents a literature study of file formats for visualization of 3D models.

**Chapter 4** discusses how to access the NX Open API to be able to access NX through its programming interface. Appendix D goes through this in greater detail by describing the NX Open API.

**Chapter 5** gives an overview over the process of the Web Based Customizable Design application by giving a birds eye-view at first and then going into detail on the different working environments.

**Chapter 6** describes the software application in detail.

**Chapter 7** describes the use of the WBCD application by discussing benefits, limitations and implementation of the application.

**Chapter 8** discusses the results from the thesis. This includes revisiting the software requirements, comparing the result to CBD, explain choices made during the thesis.

**Chapter 9** concludes the thesis and presents some possibilities for further work.

**Appendix A** contains the source code for the journal file generated by NX.

**Appendix B** contains the source code for the WBCD client application, client application design and server application.

**Appendix C** contains the source code for the batch file used to run the journal file in the WBCD application

**Appendix D** describes the functions of the code used in the journal file to be able to understand the NX Open API.

**Appendix E** is an email from Infinity Innovations describing their quote for implementing Anark Core Server (Norwegian email)

**Digital Attachments** A .zip file is available as a digital attachment on the NTNU DAIM system. This .zip file contains the files listed in Table 1.1.

File name
Flange_by_WBCD.jt
Flange_for_WBCD.prt
journal file.vb
WBCD client application designer source code.vb
WBCD client application source code.vb
WBCD server application source code.vb

Table 1.1: Digital attachments

The questions listed in the thesis are answered in the following chapters:

**Problem 1** is answered in Chapter 3.

**Problem 2** is answered in Chapter 4 where the possible ways to access the NX programming interface is discussed, in section 5.4 where the use of journaling is described, and in section 6.3 where the ways the server handles updating and running journal files are described.

**Problem 3** is answered in Section 5.3 where the model used for the thesis is discussed. There is also included a guide for what is important when creating the parametrizable models.

**Problem 4** is answered in chapter 6 where an application for remotely customizing and updating 3D models in NX is described. This differs some from the problem as an application that only works for this solution was created where as the problem specified a generic web site. Chapter 8 discusses why this choice was made. Source code for the solution is found in Appendix B.

**Problem 5** was not answered in this thesis. After trying to get in contact with companies through Summit Systems for several weeks and getting no where it was decided to drop this problem after discussion with the main faculty advisor, Terje Rølvåg.

## 1.6 Software

This section gives a description of the software used in this thesis.

### 1.6.1 NX 8.5

“NX offers the industry’s broadest suite of integrated, fully associative CAD/-CAM/CAE applications. NX touches the full range of development processes in product design, manufacturing and simulation, allowing companies to encourage the use of best practices by capturing and re-using product and process knowledge” [9].

### 1.6.2 Microsoft Visual Studios 2008

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications[10]. Visual Studio is a suite of component-based software development tools and other technologies for building powerful, high-performance applications [11].

This is the program that was used to create the application that can change the parameters of an NX model over a network connection. Two applications were created, a Server and a Client. The Client was created as a Windows Form Application and the Server was created as a Windows Console Application

The decision to use Visual Studio 2008 was made because the NX Open API wizard does not work in the newer versions of Visual Studio. This was discovered after having problems installing the wizard and consulting Summit Systems.

#### Windows Form Application

Windows Forms is a graphical application programming interface that can be created in Visual Studios. It is a tool to design user interfaces for applications programmed in .NET languages such as Visual Basic, C++ and C# (C sharp).

#### Windows Console Application

A console application is a text-only computer interface. “Consoles provide high-level support for simple character-mode applications that interact with the user by using functions that read from standard input and write to standard output or standard error“ [12].



### **1.6.3 JT2GO**

JT2GO is a free viewer for JT files developed by Siemens PLM Software. It is the most robust 3D visualization format available today and is the world's most widely used 3D file format for PLM. It can be downloaded from Siemens' web pages [13, 6].

### **1.6.4 FTP Server**

File Transfer Protocol (FTP) is a network protocol that is used to transfer files between hosts over the internet [14]. For this thesis a FTP server was supplied by Summit AS. This means that files can be uploaded to this server by one host and downloaded by another host. This is used to transfer files between the Server application and Client application in this thesis.

### **1.6.5 Batch File**

Batch files are used to simplify routines or repetitive tasks. It is an unformatted text file containing commands to be executed when the system runs the file. It has a .bat or .cmd file name extension [15].

## Chapter 2

# Software Requirements Specification

### 2.1 Purpose

This software requirements specification describes an application that can update a NX 8.5 CAD model via the internet. The application can be used to update models on the go or to create modified versions of a product. The purpose of creating the system is to figure out how it is possible to remotely update CAD models with a focus on the method, not the complexity of the end result.

### 2.2 Scope

The system that is specified, “Web Based Customizable Design”, should be a program with a graphical user interface that allows the user to change the values of parameters in a CAD model. A server will be running on a system that has access to NX 8.5 and can update the CAD model. The user should be able to do the changes on a remote system and connecting to the system running the server using internet connection. A CAD model updated to the selected parameters should be downloaded and opened on the users system.

The system can be viewed as a way to remote control a CAD system and can give benefits by removing possible errors in modeling, reducing design time, setting design rules and make it possible to make changes on the go. By making

a user interface where the user can change the values of parameters the need for CAD experience will be gone.

## 2.3 Definition of Terms

Abbreviation	Meaning
WBCD	Web Based Customizable Design
GUI	Graphical User Interface
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CAE	Computer Aided Engineering
NX 8.5	CAD/CAM/CAE program by Siemens
NX	Referring to NX 8,5 or previous versions
JT	3D data format by Siemens
FTP	File Transfer Protocol
FEM	Finite Element Method
FE	Finite Element
CBD	Custom Board Design
3D PDF	3D visualization format
EPS	Expanded Polystyrene
PU	Polyurethane
CNC	Computer Numerical Control
PTS	Product Template Studio
PLM	Product Lifecycle Management
API	Application Programming Interface
ASP.NET	Server-side Web application framework
VB	Visual Basic - .NET programming language
C#	C sharp, Programming language
C++	Programming language
ISO	International Standard Organization
JT2GO	JT file viewer by Siemens
STEP	STandardized Exchange and Product model data
TCP	Transmission Control Protocol
VBScript	Visual Basic Scripting (for web pages)

Table 2.1: List of Abbreviations

## 2.4 Intended Use and Users

Web Based Customizable Design is intended to be used by sales personnel in meetings to get an updated model to the customer's specifications on the go, by designers that can use different specifications on products in a product family or any other user that want a customized version of the part. The users of Web Based Customizable Design are not expected to have CAD experience. It is important that the GUI is simple and intuitive.

## 2.5 Operating Environment

The software will operate on two or more different systems that are connected to the internet so they are able to communicate with each other. The server is an executable file (.exe) that runs on newer Windows operative systems on a computer that has NX 8.5 installed. The client is an executable file (.exe) and can be run on newer Windows operative systems and only need to have a program that can view JT files. A JT viewer such as JT2GO or any major CAD system can be used to view JT files.

The server needs to use batch files, visual basic text files and command prompt. The client needs to know the IP address of the server. Both the user system and the server system need to have the address and credentials of the FTP Server.

## 2.6 Overall System Requirements

The overall system requirements for the application are defined by the scope, requests from adviser and the needs discovered while working on the thesis. In Chapter 8 this table will be revisited to see the status of the requirements in Table 2.2.

#	Requirement
1	The system should showcase possibilities for different types of input in the GUI
2	The system can update CAD models
3	The system should be generic and can be used on any model
4	The system should be able to update different kinds of parameters
5	The system can use pre-defined standard dimensions on models (ISO standards)
6	The system doesn't require CAD experience to use
7	The system can export updated CAD models in selected format
8	The NX window is never opened in the process
9	The system should store the result file on the user's system
10	The server can handle multiple requests from users without being restarted
11	The system should be faster than modeling the part from scratch
12	The client solution should be made as a web site
13	The system does not require NX installed or licensed on the client

Table 2.2: Overall system requirements

# Chapter 3

## File Formats for Visualization

### 3.1 Overview

This chapter will discuss some of the different file formats NX 8.5 can export to figure out what format is optimal for visualizing NX-models.

Below is a list of the file formats that NX can export [16]. The interesting formats for this thesis are JT, Part files and STEP.

DWG
DXF
IGES
JT
Parasolid
PROE
SolidWorks
STEP
STL
Part file

Table 3.1: File formats that NX can export

In addition to the formats NX can export, it is possible to use the 3D PDF format to visualize 3D models by converting JT files. The table below shows the four formats that were studied for this thesis and their uses [17].

File format	Use
JT	<b>3D Visualization Format</b>
	Used for collaboration
	No need to undergo multiple translation cycles between different systems
	Once exported it can be used by other CAD systems
	Translators available for all major CAD systems
STEP	<b>Data Exchanging Format</b>
	Used for communication and file transfer
	Transfer all data from one CAD system to another CAD system
3D PDF	Turned into a proprietary format before being used again
	<b>Document Format</b>
NX Part File	Used to enable 3D content in PDF documents
	<b>NX File Format</b>
	Used for modeling

Table 3.2: Use of file formats

Firewire’s Custom Board Design uses the JT file format in collaboration with Anark Core Server to create a customized 3D PDF file of the custom board that includes the 3D solid model of the base design and all the exact dimensions. In this model it is also possible to compare the customized board to the stock board that they changed. To visualize this they use selective translucency so the user can get a feeling of what has been changed compared to the stock model[2].

## 3.2 JT

JT is the world’s most widely used 3D file format for Product Lifecycle Management (PLM) [13]. It is the first 3D visualization format to get an ISO International Standard. It is a compact and accurate format that is used to communicate the very important design information that typically only can be found in a CAD file.

Siemens PLM software customers rely on JT to be the most robust 3D visualization format available today [13]. It can be used to show both CAD files and FEM analyzes.

The JT files are compact and small in size compared to other formats which makes them suitable for viewing files on laptops and to be uploaded and down-

loaded from online servers. “JT format is an industry focused, high-performance, lightweight, flexible file format for capturing and re purposing 3D Product Definition data that enables collaboration, validation and visualization throughout the extended enterprise” [13].

JT requires a viewer such as JT2GO or a CAD program to be viewed. JT2GO is free and can easily be downloaded from “Siemens JT Open” [6] and there are no need to undergo any translation cycles when the files are viewed in different systems [17]. Once the JT file is exported by NX it can be viewed in all major CAD systems.

### **3.3 STEP**

STEP stands for STandardized Exchange of Product model data. “STEP is intended to provide industry wide descriptions of engineering systems” [18]. STEP files are used when CAD files needs to be transferred from one CAD system to another CAD system, e.g. from NX 8.5 to Autodesk Inventor. To be able to be viewed in several different CAD systems the STEP files needs to communicate as much information as possible which leads to large files that are not suitable for simple visualization purposes. As all CAD systems have different methods of using the different file types, the STEP files are transformed to a proprietary format before it is ready to be used again.

### **3.4 3D PDF**

3D PDF is a way to share 3D models so that they can be viewed on any computer that has Adobe Acrobat Reader installed. The document-centric outputs are interactive, meaning you can zoom and rotate the model inside the document [19].

Creating 3D PDF files are not as easy as creating the other file formats. Anark Core Servers delivers a package that makes this possible and after an email exchange with the Norwegian Supplier “Infinity Innovations” the price of the package was estimated to be 6500 NOK for the package of Anark Core Servers and Adobe Acrobat Pro XI. The email can be found in Appendix E.



## 3.5 NX Part Files

A NX part file can contain information such as a 3D model, a 2D drawing, or NX CAM mode programming.

It contains all the information needed for the model to be viewed and edited in NX but it can not be transferred to any other CAD system. Other programs can be used to view the NX Part files but this is not the focus of this thesis [20].

## 3.6 Chapter Discussion

As described in the previous sections each of the four studied file formats have different uses. JT is for visualization, STEP is for exchanging data, 3D PDF is for documents and NX Part files are used for modeling, 2D documentation, simulations and CAM programming. All the formats have the possibility to show a 3D model and 3D images exported by NX but for the purpose of this thesis the JT format should be used for visualization. This was decided because both the literature study and conversations with employees at Summit showed that the JT file format is best suited for visualizing NX models. 3D PDF files can be easier to read but the document format makes the model unable to be used again by another program while the JT files can both be viewed and later used in other systems.

With JT files users can create a mock up of a model. This means that engineers can validate that products don't interfere with each other before a prototype is made. This is often used in large assemblies such as oil rigs where there can be several thousand parts in an assembly. In these assemblies the file size of each file is important because of the computing power needed to view all parts. The JT files created from the Web Based Customizable Design program will be able to be put directly into such an assembly.

## Chapter 4

# Importing, Updating and Exporting in NX

### 4.1 NX Open API

“Open API is a collection of routines that allows programs to access and affect the NX Object Model” [21].

To be able to do changes to a model in NX without manually pressing the buttons in the interface changes must be made through the NX Open API (Application Programming Interface). “Open API is a collection of routines that allows programs to access and affect the NX Object Model” [21]. An API can be viewed at as the “backdoor” to NX and is the interface that users need to access to make custom features for NX. The NX Open API has great support for .NET programming languages such as VB (Visual Basic) and C# (C sharp). In this thesis the programming was done in VB.

There are different ways to manipulate NX through the NX Open API. A program can be coded through a NX Open API Wizard in Visual Studio 2008 (Not supported in newer versions of Visual Studio) or compiled as a standalone program. Macro files can be used to manipulate models in NX and do FEM (Finite Element Method) simulations but this was not the focus of this thesis and will not be studied. Another option is to make a Journal file in NX and modify it so that NX compiles the file internally.

## 4.2 Importing, Updating and Exporting Results in NX

Manipulating data in NX can be done in several different ways. This section will discuss some of these ways.

### 4.2.1 NX Open API Wizard

In Visual Studio 2005 and 2008 there is an application wizard that developers can use to program extension addons or plugins. This doesn't work for the newer versions of Visual Studio. This wizard helps with generating much of the code that is standard in every extension to NX. An application wizard can be described as a guide through a set of steps that the user has to go through. This is usually set up as a sequence of dialog boxes where the user gets several options, or yes/no options of what code he wants generated. Using a wizard is great for getting some of the standard code that every program needs that either is trivial or infrequently used and it is great for linking the program to the NX Open libraries that are included in the NX install folder. The NX Open libraries are what make the compiler able to understand NX-specific code.

Extensions or plugins created by the NX Open API Wizard can either be compiled as a stand alone program and run as an .exe file or compiled directly in NX by creating a custom button that runs a script.

Appendix D goes through some of the most important commands created by journal files for this thesis and describes what their function is. This is the same type of code that is used in applications coded through the NX Open API Wizard.

### 4.2.2 Macro

When a user sets up a macro it sets up a sequence of commands that will be executed every time someone runs the macro file [22]. This could also be used for this purpose, but macro files generate source code that is very difficult to understand and records every mouse-click and keyboard button clicked during the recording of the macro. This makes it a bad candidate for this thesis because it is hard to find what part of the code is relevant for the developer. Figure 4.1 shows a screen shot of some of the code generated by macros just to give an example of how hard macros are to work with.

```

FOCUS CHANGE IN 1
CURSOR_EVENT 1001 3,1,100,0 ! single_pt, mb1/0+0, , nn
CPOS 18.4286726007631,26.9848420225459,0
ASK_ITEM 5832704 (1 OPTI 0) = 3 0 ! End Point
ASK_ITEM 5832704 (1 OPTI 0) = 10 1 ! Fixed
ASK_ITEM 5832704 (1 OPTI 0) = 10 1 ! Fixed
ASK_ITEM 5832704 (1 OPTI 0) = 10 0 ! Fixed
ASK_ITEM 6422528 (1 OPTM 0) = 3 ! End Point
ASK_ITEM 6422528 (1 OPTM 0) = 12 ! Show Shortcuts
ASK_ITEM 6422530 (1 TOOL 0) = -1 !
OK 0 0 ! OK Callback
ASK_ITEM 3080192 (1 BOOL 0) = 1 ! Select Planar Face or Plane
ASK_ITEM 3080192 (1 BOOL 0) = 0 ! Select Planar Face or Plane
END_ITEM 262144 (1 OPTM 0) = 0 ! On Plane
END_ITEM 262146 (1 TOOL 0) = 0 ! On Plane
END_ITEM 1245184 (1 OPTM 0) = 0 ! Inferred

```

Figure 4.1: Macro code example

### 4.2.3 Journal

Journal is a tool in NX that allows a user to automate routines and is described in Chapter 5. Section 5.4 “Journal” goes through the steps a user have to go through to record and edit a journal while Section 5.3 “Model” goes through the functions used in journaling for this thesis. Figure 4.2 shows what journal code in Visual Basic looks like and it is easier to work with and understand than macro code because it the Visual Basic programming language as output. Journaling does not cover as many functions in NX as macros but covers all the necessary functions needed for this thesis.

```

'-----
' Menu: Tools->Expression...
'-----
Dim markId3 As Session.UndoMarkId
markId3 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Expression")

Dim expression1 As Expression = CType(workPart.Expressions.FindObject("flange_thickness"), Expression)

Dim unit1 As Unit = CType(workPart.UnitCollection.FindObject("MilliMeter"), Unit)

workPart.Expressions.EditWithUnits(expression1, unit1, "5")

```

Figure 4.2: Journal code example

### 4.3 Chapter Discussion

It was decided to use journaling because it was a good way to learn how to access the NX Open API and it was compatible with the needs of this thesis to create an application to customize a 3D model. It generates all the code needed for this application and only small parts of the code needs to be changed and those parts are easy to find in the file. The choice to not use macro files was made because they create a lot of unnecessary overhead information (such as mouse clicks) which makes it hard to find the relevant information. This is especially a big hindrance if the developer is new to programming in NX. Actions performed in PTS are not recorded by journal files and PTS was therefore not used for this thesis.

For a developer to understand the NX Open API he can simply perform an action while recording a journal file and review the journal file later to see what was added. This would be much harder to do using macro files.

# Chapter 5

## Process Description

### 5.1 Overall Process Description

This chapter will describe the process a model goes through. The flow charts in this thesis are made with a top-down structure which means that the first flow charts shows the major steps from a birds-eye view while the following flow charts goes more into detail about the different programs and environments.

The process starts with opening NX to create and design the model, modifying it in the journal file and finally customizing it in the Web Based Customizable Design application. Figure 5.1 is a flow chart with an overview of the process of a model. The different colors describe what is going on in different programs or environments.

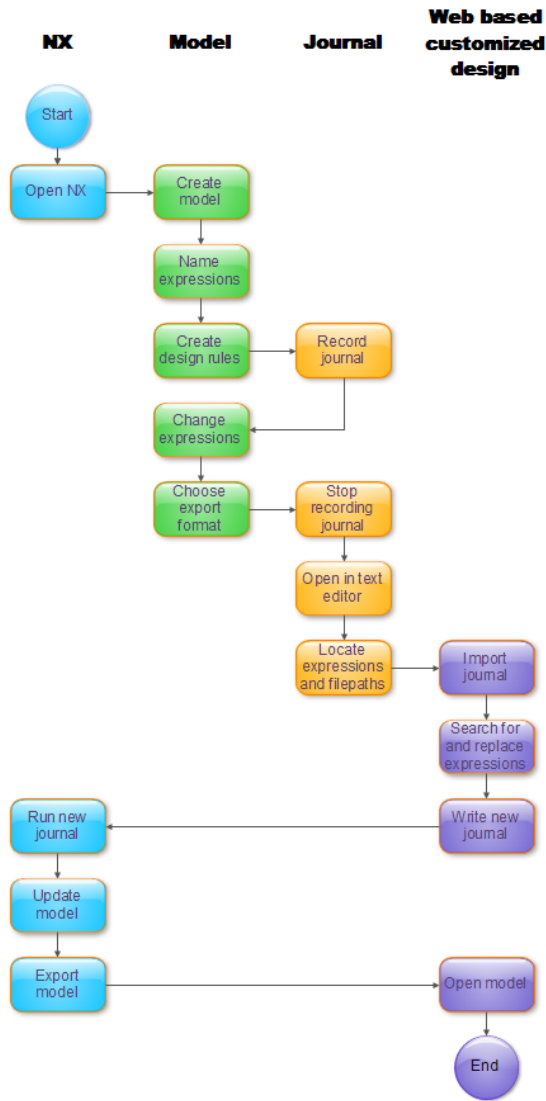


Figure 5.1: Flow chart of the process description

Some of the tasks in the process description are there to show what is done by the developer of the software while some are there to show what the application does. Figure 5.2 shows this by framing the actions performed by the developer and the actions performed by the application.

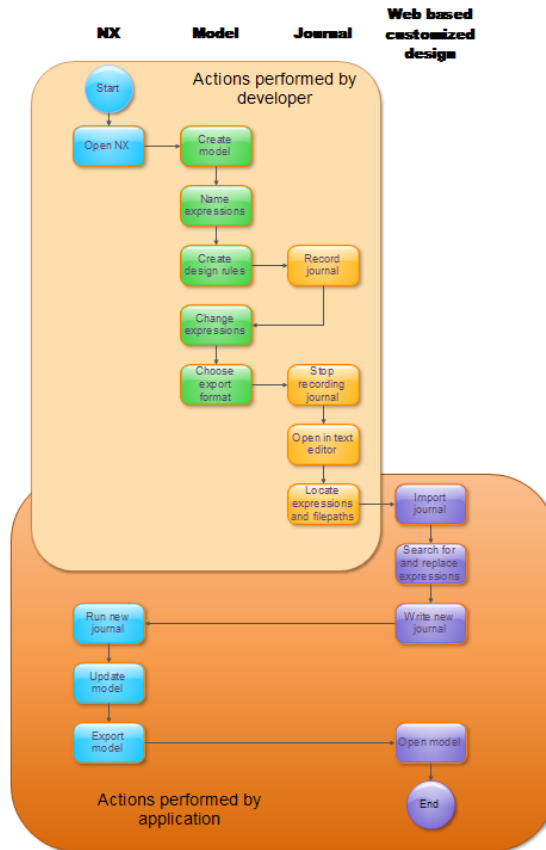


Figure 5.2: Actions performed by developers and actions performed by application



## **5.2 NX**

### **5.2.1 Overview**

This section will describe what is done by the developer and what is later done by the application in NX. Figure 5.3 is a flow chart with an overview of the process in NX. The different sections describe what is done by the developer and what is done by the application.

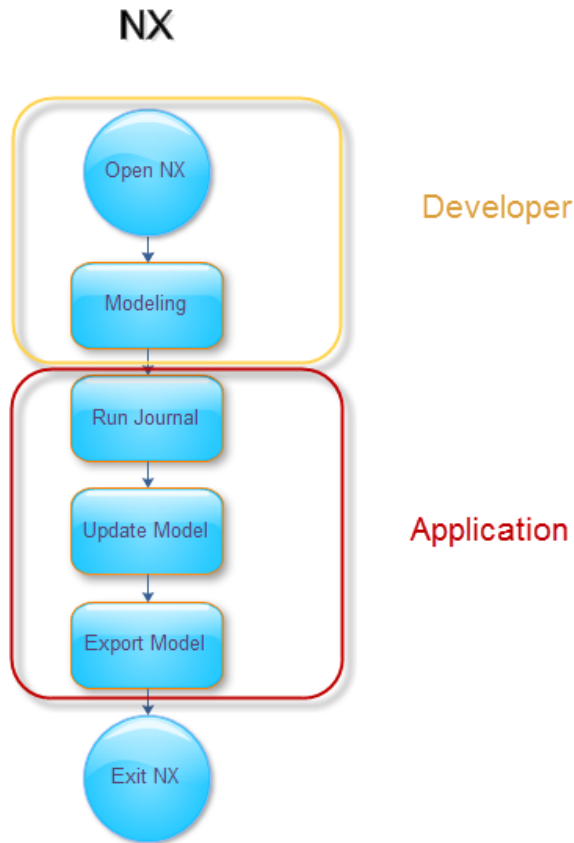


Figure 5.3: Flow chart of the process in NX

### 5.2.2 Modeling

At the start of the process NX is opened and a model is created and modeled by the developer as described in Section 5.3 called "Model".

### **5.2.3 Run Journal**

Running the new journal is done at a later stage in the process as can be seen in the overview flow chart in Figure 5.1. Running the journal is done by calling a cmd prompt command window and using the run\_journal function to run the new journal created by WBCD. A more detailed description can be found in Section 5.4.6.

### **5.2.4 Update Model**

When the user runs the journal the model is automatically updated by NX to the new values selected by the user in the Web Based Customizable Design application.

### **5.2.5 Export Model**

The user selects the wanted format to export the model in and chooses what location to export the model to. It is exported in the selected format to the selected location by NX. In this thesis the JT file format is used as export format because this was figured out to be the best visualization format in Chapter 3.

## **5.3 Model**

### **5.3.1 Overview**

This section describes the actions done by the developer in the modeling and sketching environment in NX. Figure 5.4 shows a flow chart of the tasks done by the developer when modeling and recording the journal file.

# Model

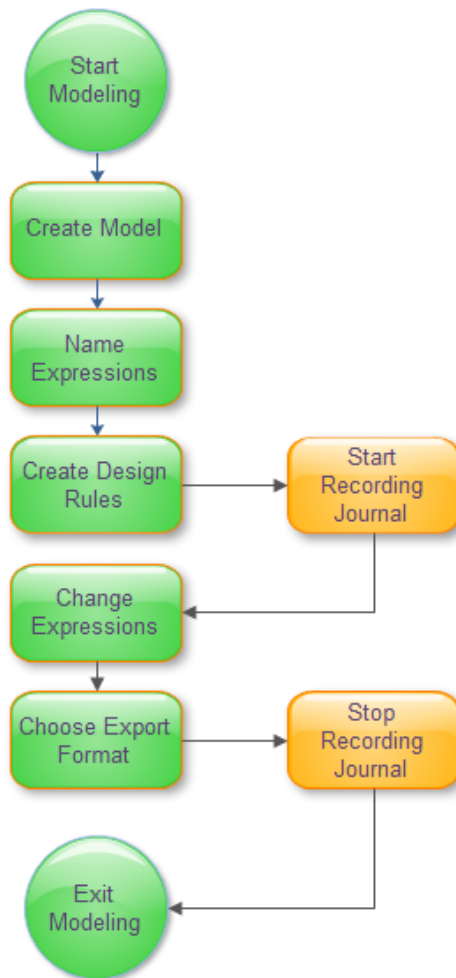


Figure 5.4: Flow chart of the modeling in NX

### 5.3.2 Create Model

The first step is to create the model you want to be able to customize.

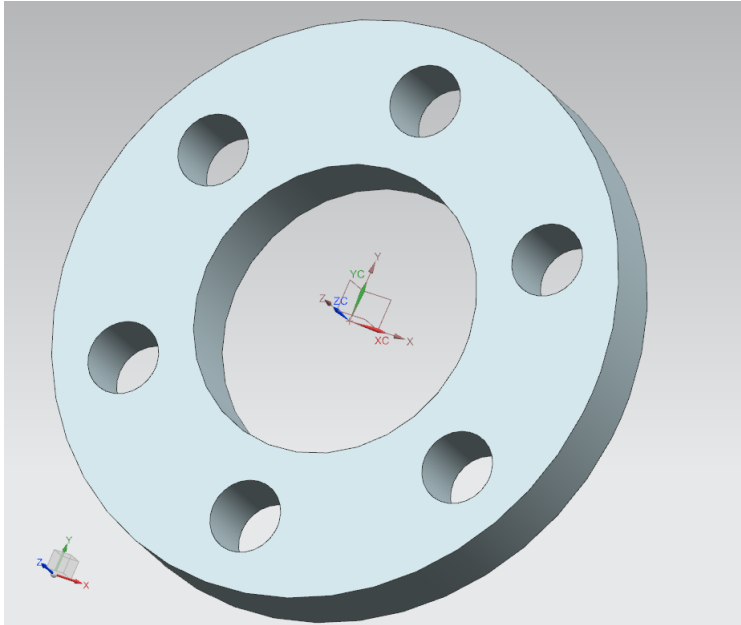


Figure 5.5: Flange model from NX

For this thesis a simple model of a flange that can be used to seal connection between pipes was created. It was chosen to get a model that was easy to work with and that was not too complex while still having several different parameters that can be changed.

### 5.3.3 Name Expressions

To identify the important parameters and easily be able to understand what they control they should be named according to their function. For this thesis 5 expressions were named and these control the 5 parameters it is possible to customize using the Web Based Customizable Design application.

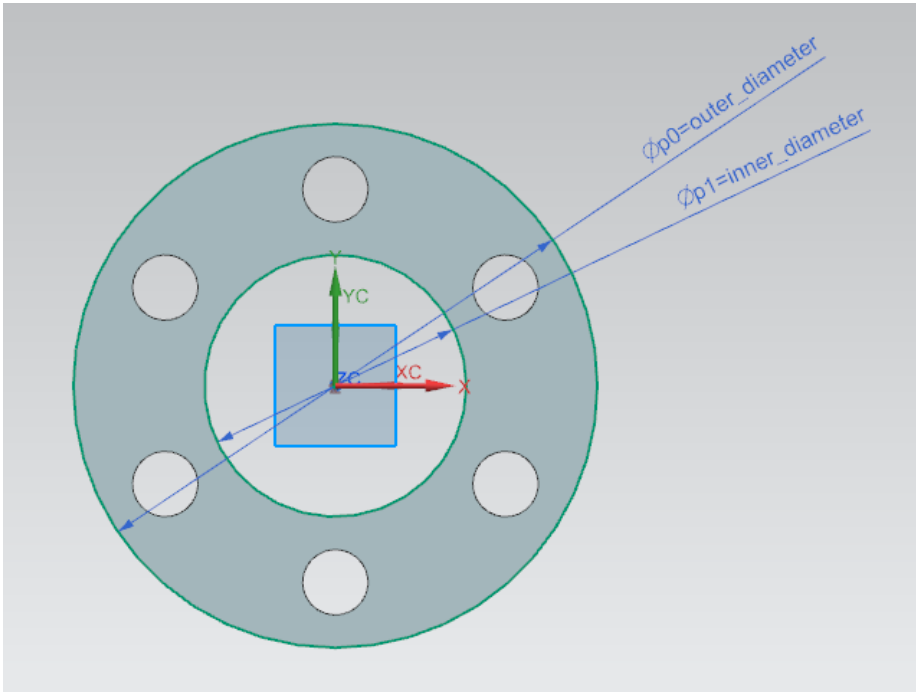


Figure 5.6: Named expressions in sketch

Figure 5.6 shows how it is possible to name the expressions with names that describe the function. In this figure expression  $p0=outer\_diameter$  describes the outer diameter while  $p1=inner\_diameter$  describes the inner diameter. The  $\emptyset$  in front of the expression name is to show that the dimension is a diameter.

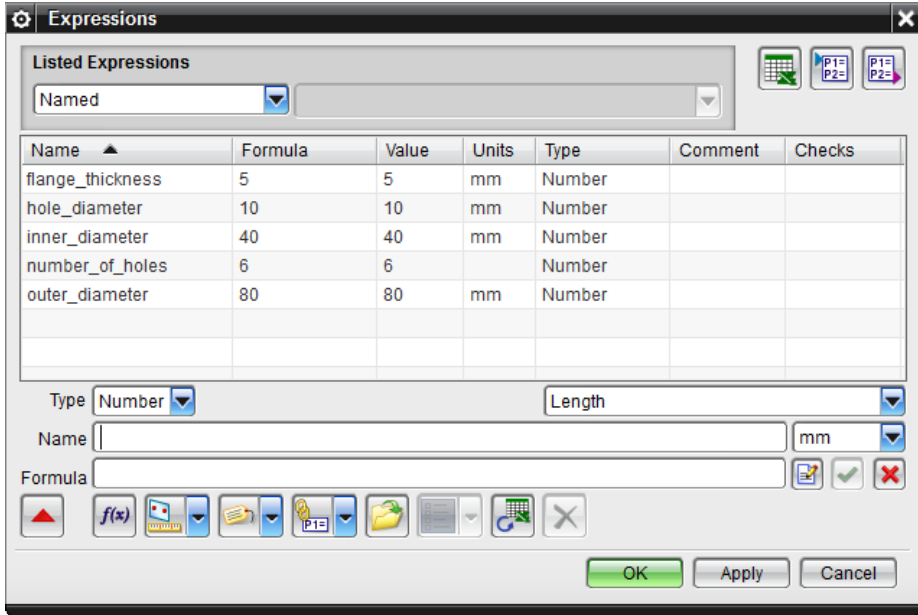


Figure 5.7: Named expressions in a list

Figure 5.7 shows a list of all the named expressions used in this thesis.

### 5.3.4 Create Design Rules

The model needs to have a set of rules as to not break the design intent. These rules can for example be that inner diameter has to be smaller than outer diameter and that the hole diameter has to be smaller than the difference between outer diameter and inner diameter.

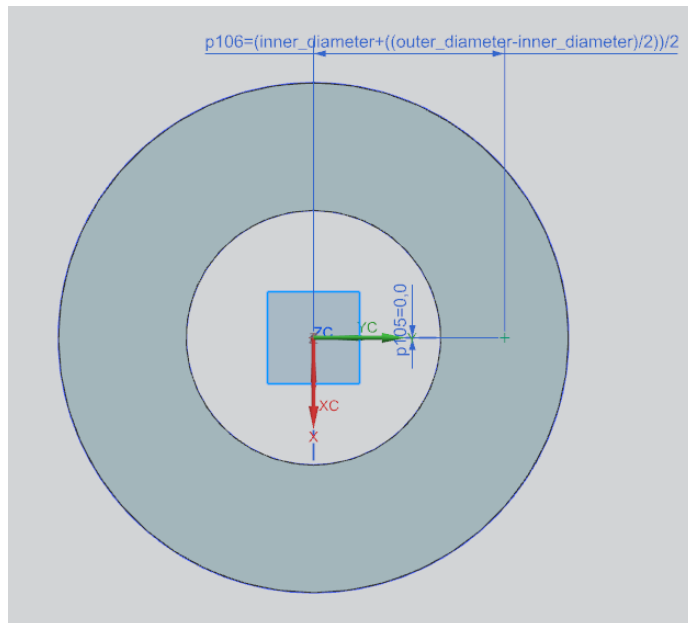


Figure 5.8: Design rules for hole placement

An example used in this model is that the holes should be placed centered between the inner- and outer diameter. By using the named expressions for these parameters we ensure that when the diameters are changed the hole placement is updated accordingly. The formula used to decide hole placement is shown below where  $D_{HolePlacement}$  =the diameter for hole placement,  $D_{inner}$  =the inner diameter and  $D_{outer}$  =the outer diameter.



$$D_{HolePlacement} = \frac{(D_{inner}) + \left(\frac{D_{outer} - D_{inner}}{2}\right)}{2}$$

Figure 5.9: Hole placement formula

At this point in the process the use of Product Template Studio could be implemented, but testing showed that changes made in PTS wasn't recorded in the journal files and for that reason it was decided not to use PTS to create rules for the model in this thesis.

### 5.3.5 Change Expressions

Before doing this step it is important to start recording the journal as described in Section 5.4.

To make it easy to find the parameters you want to be able to change you should name them with describing names that are easy to find. In the table below one can see the name and units of the expressions in the model.

Expression name	Unit
flange_thickness	mm
hole_diameter	mm
inner_diameter	mm
number_of_holes	
outer_diameter	mm

Table 5.1: List of important expressions in model

To change the parameters in the important expressions you can access them by going to Tools -> Expressions (Ctrl + E). This will bring up the list below where you can see the expression names, values and units [23]. For the expressions to be recorded in the journal file you need to change the value of the expression. It doesn't have to be changed to the correct value as this can be fixed in the journal file later but it has to be changed into another number. The units are important because NX creates code in the journal that specifies the unit. The unit "mm" is millimeters and the blank field is a constant with no unit.

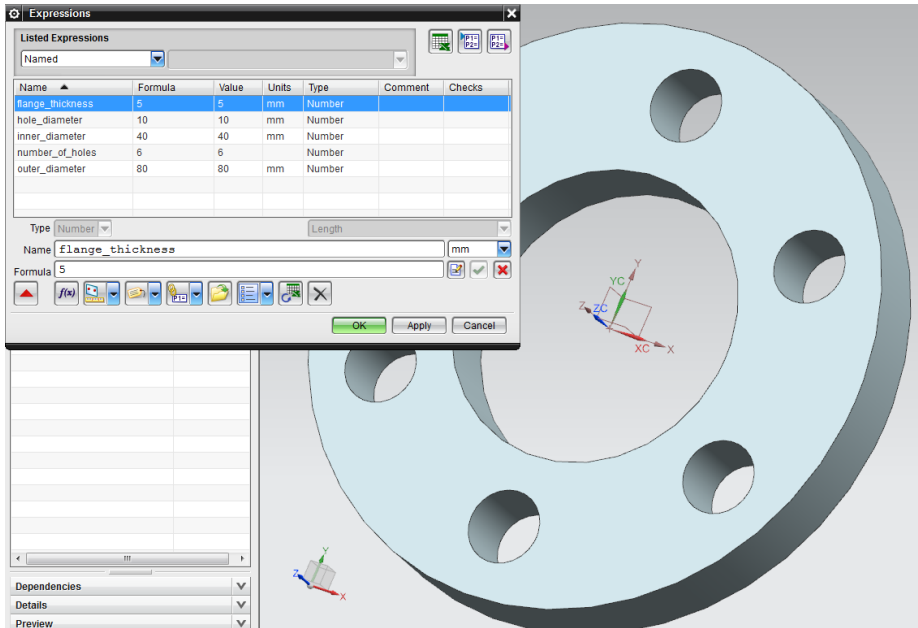


Figure 5.10: Changing the value of an expression

### 5.3.6 Choose Export Format

Save the file and export it by going to (File -> Export -> Select format) in the wanted format. For this thesis the JT file format is selected.

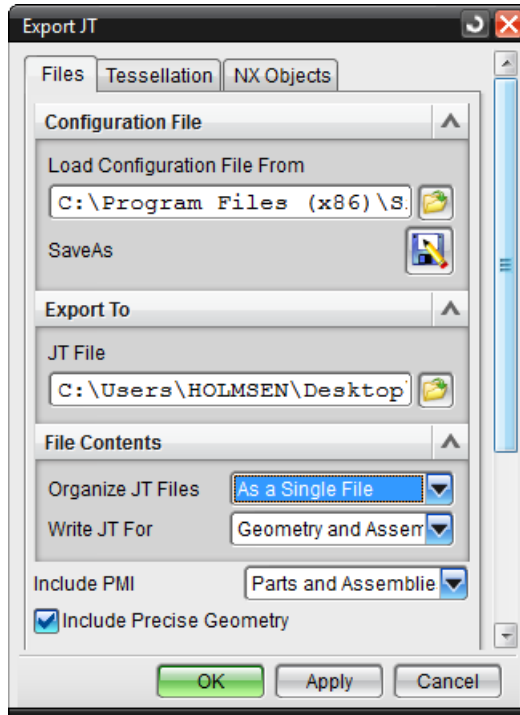


Figure 5.11: Export JT files

It is important to select “As a Single File” under File Contents in the Organize JT Files field. This ensures that the geometry is saved in the JT file without depending on the .prt (part file) of the model. This makes the file monolithic so that it can still be easily viewed after being downloaded from a FTP server.

Exportable file types
JT
Part
Assembly
STEP
FEM
SIM
Image (.png, .jpeg)
PDF
Draftings

Table 5.2: List of available export formats from NX

Table 5.2 shows some of the file formats that NX can export. There are other file formats that are possible to export but these are not mentioned as they are not in the scope of this thesis.

## 5.4 Journal

### 5.4.1 Overview

This section describes the steps needed to create a journal file with NX and edit it.

# Journal



Figure 5.12: Flow chart of tasks for Journal

## 5.4.2 Record Journal

A journal file is created when you click “Record Journal” in NX. A journal is like a log of every tool or function you use in NX and is stored in a text file. You can run the journal file at a later time and NX will perform the same actions again. It is important to start the journal recording before you open the file you are using so that NX knows what file you want to work on. The journal menu isn’t available by default in the NX start up window but by right clicking the banner and selecting “Journal” you can make it available. The “Record Journal” button is located at (Tools -> Journal -> Record).

After the model is opened the user has to go through all the steps he would do every time he want to change something in the model. Change all expressions that you need to be able to change because this creates a reference to them in the journal file. This is what makes it possible to change the parameters of the model at later stages in the process.

Journal file can be exported in Visual Basic and C# but in this thesis Visual Basic is used (.vb file extension) as this is the same programming language as the Web Based Customizable Design application is created in.

### Further possibilities when recording journal files

It is also possible to do more advanced operations via the journal files such as simulations in NX Nastran and create CAM production files so that the paths for a CNC machine can be created automatically for production [2].

In the programming for this thesis the names of the expressions are hard-coded, meaning that the names are physically coded and if e.g. an expression name is changed the program will not work anymore. It is possible to research the use of the export expressions tool from “Tools -> Import and Export Expressions” to figure out ways to avoid this Generalproblem.

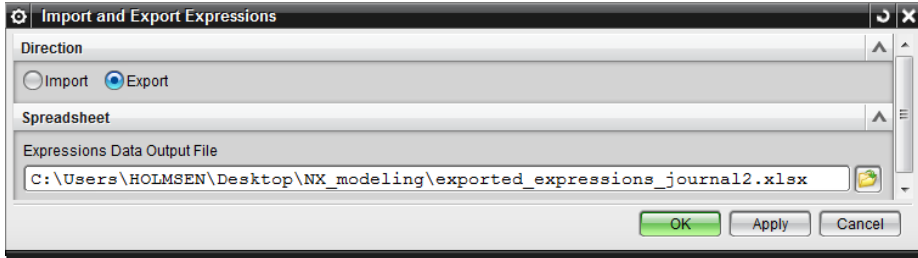


Figure 5.13: Importing and exporting expressions in NX

When exporting expressions an Excel file is created at the selected output location. This Excel file can be opened and variable names, values and units can be extracted as viewed below.

	A	B	C	D	E	F	G
1	NAME	FORMULA	VALUE	UNITS	TYPE	COMMENT	
2	flange_thickness	5	5	mm	Number		
3	hole_diameter	10	10	mm	Number		
4	inner_diameter	40	40	mm	Number		
5	number_of_holes	6	6		Number		
6	outer_diameter	80	80	mm	Number		
7							
8							

Figure 5.14: Exported expressions in Excel

Only the named expressions will be exported in the Excel file.

Custom Board Design (discussed in Section 1.2) exports information such as the volume of surfboards which is hard to calculate manually. The same method can also be used to export information on volume and weight of models which can be useful to e.g. calculate price and amount of materials needed.

These operations are not in the scope of this thesis but are mentioned because it showcases some additional possible uses of the application.

### 5.4.3 Stop Recording Journal

After going through all the necessary steps in NX press the stop record button to stop recording the NX session save the journal text file.

Operations performed before stopping the journal:

- Record Journal
- Open part file
- Change expressions
- Save file
- Export JT file

### 5.4.4 Open in Text Editor

By editing parameters in the journal file the Web Based Customizable Design application is able to change the parameters of a model. To be able to find the location of the expressions and file paths of the model the journal file must be opened in a text editor such as e.g. “Notepad”, “Notepad++” or Visual Studio. In this file we can see the code generated by NX when recording the journal file, and the language it is coded in is the language selected when recording the journal. Much of the code is not interesting when the only thing to be changed is the values of some expressions but the rest should not be removed as NX needs it to repeat the process. A description of the most important commands found in the journal file can be found in Appendix D. For more detailed information on the classes check the “NX85 NXOpen .Net API Reference” [24].

### 5.4.5 Locate Expressions and File Paths

When the journal file has been opened in the text editor the expressions that were changed when recording the journal can be found by either searching for the expression names or by scrolling through the document.

```
Dim expression1 As Expression = CType(workPart.Expressions.FindObject("flange_thickness"), Expression)
Dim unit1 As Unit = CType(workPart.UnitCollection.FindObject("MilliMeter"), Unit)
workPart.Expressions.EditWithUnits(expression1, unit1, "8")
```

Figure 5.15: Changing the value of an expression example code



Figure 5.15 shows an example of what the code looks like when the expression for “flange\_thickness” is changed. It is worth noting that also the unit of the expression is declared. Similar code is created for the other expressions that were changed during the recording of the journal file.

```
jtCreator1.OutputJtFile = "C:\Users\HOLMSEN\Desktop\NX_modeling\Flange_expressions.jt"
```

Figure 5.16: Code showing file path of exported JT file

Figure 5.16 shows the line of code where the location of the output file in the JT format is saved. This is useful because this location is important for handling the file after it is exported.

#### 5.4.6 Running a Journal File

There are two ways to run a journal file. The most common is to use the user interface in NX and play it by going to Tools -> Journal -> Play and selecting the journal file. The other option is to use a command prompt to start NX by the command “run\_journal”.

Run\_journal is a command that is used to start a journal file without starting the NX user interface but perform the action in windowless mode. To execute the command the command prompt must change directory to the install folder of NX to be able to reach the UGII.bat file which starts NX. Run\_journal is followed by the file path of the journal file.

To use the run\_journal command a batch file was created. This batch file first changes directory to the NX install folder and then running the journal. Figure 5.17 shows the command prompt that is started by executing the batch file.

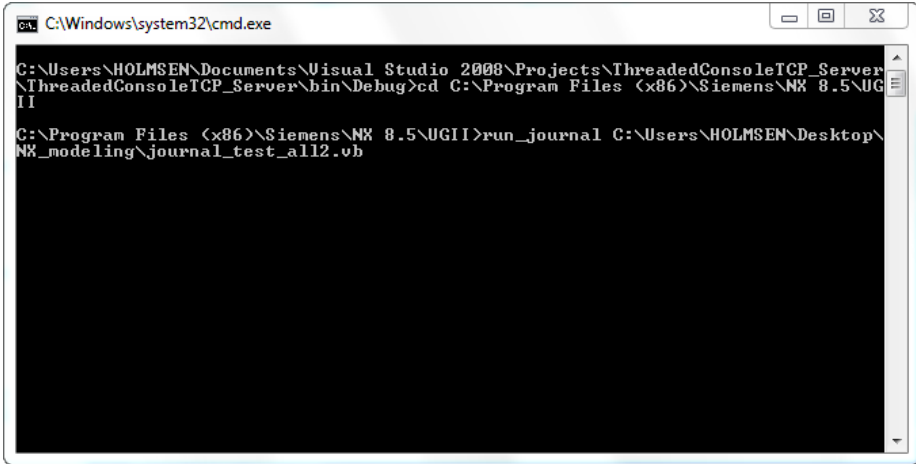


Figure 5.17: Run journal in command prompt window

Figure 5.18 shows the batch file used to run the journal file.

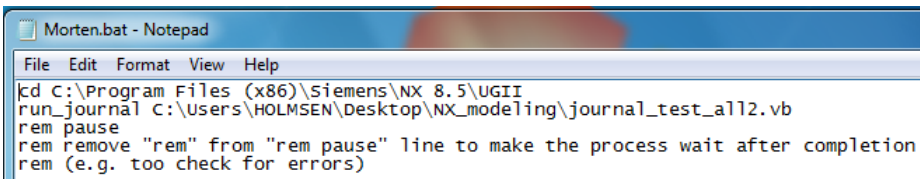


Figure 5.18: Run journal by batch file

## **5.5 Web Based Customizable Design (WBCD)**

### **5.5.1 Overview**

This section will only give a short overview over what the WBCD does without going into detail about how it does it. WBCD uses socket programming to be able to send information between two computers. Chapter 6 will go into detail of how the WBCD works.

## Web Based Customizable Design Overview

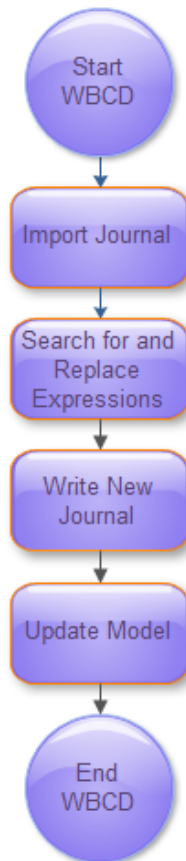


Figure 5.19: Flow chart of tasks for Journal

### **5.5.2 Import Journal**

WBCD imports the journal file that was recorded earlier to be able to make changes for it.

### **5.5.3 Search for and Replace Expressions**

WBCD uses the expressions the user located in the journal file to find and change the values to what the user has set as inputs in the application.

### **5.5.4 Write New Journal**

After the values of all expressions have been replaced the application creates a new journal file by writing to an empty text file and saving it as a .vb file. You can run the new journal file in NX to update and export the model

### **5.5.5 Open Model**

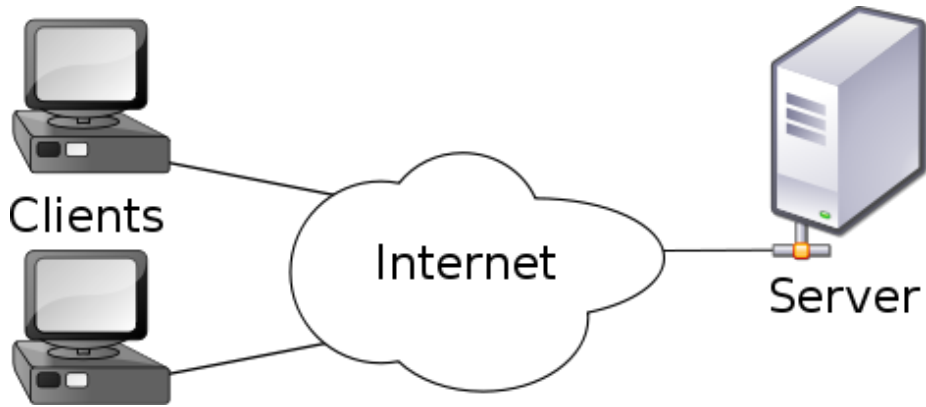
After the new journal file is sent to NX and the model is updated, a JT file is exported and WBCD opens it so the user can see the result.

## Chapter 6

# Web Based Customizable Design Application

### 6.1 Client-Server Architecture

The Web Based Customizable Design (WBCD) application is based on the client-server architecture . A client is the computer or application that requests a service from the server [25]. A server is a program that can operate as a socket listener and clients can send requests to the server via IP (Internet Protocol) [26]. When a connection request is sent from a client program the server program creates a TCP (Transmission Control Protocol) socket connection. TCP is a secure way to deliver ordered and error-checked packets between computers connected to each other via the internet[27]. This connection makes the programs able to send bytes back and forth to communicate with each other. Each time a client connects to the server it is assigned a number to identify the session.



Source: [25]

Figure 6.1: Client-server architecture illustration

The server application was made as a Windows Console Application so that it can be listening for connections on a specified port. It can be visible for the user and programmed so that the status of the process is shown in the console window or running in the background as a hidden process. The server application is located on a machine that can run NX in windowless mode (hidden for the user) when the program is running. As long as the server is running it is listening for connections from client programs on a specified port number. For this program the port number is 8888. The server uses multi-threading to be able to handle requests from multiple clients. “Multi-threading is a widespread programming and execution model that allows multiple threads to exist within the context of a single process” [28].

The client application was made using the Windows Form Application in Visual Studios 2008 which makes it possible to easily make a user interface where the parameters of the customization can be entered.

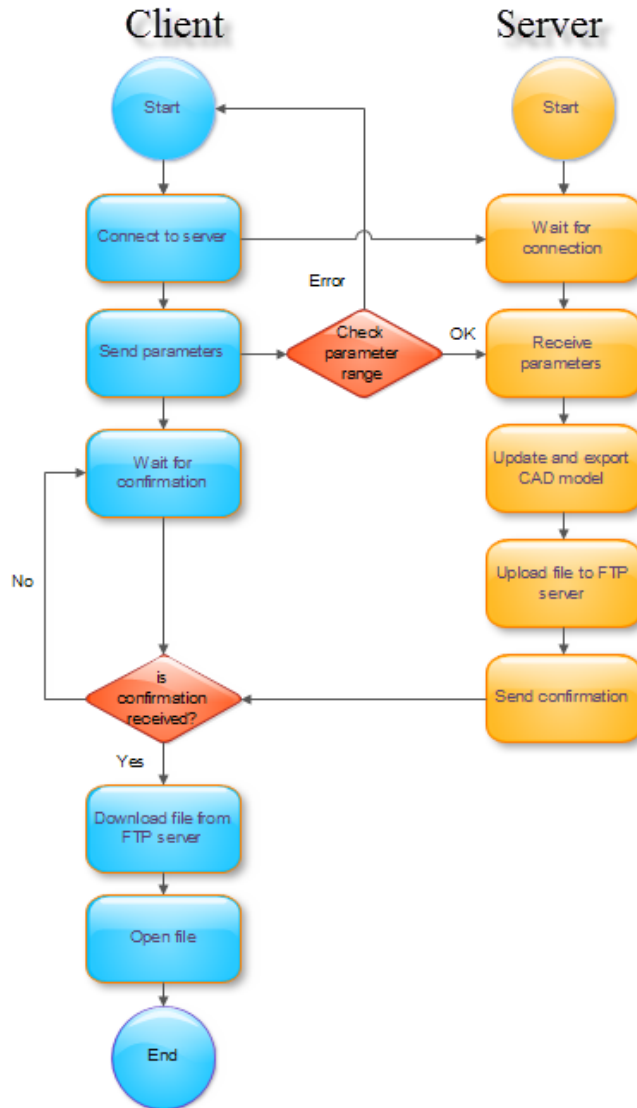


Figure 6.2: Flow chart of tasks for Client and Server



## 6.2 Client

### 6.2.1 Graphical User Interface

The client application is the part of Web Based Customizable Design the user sees. A GUI (Graphical User Interface) was created to make the customer able to easily customize the parameters. The GUI includes an image of the flange that can be customized to give the user an idea of what the model looks like. It was designed in Visual Studio 2008 using a Windows Form Application template. Figure 6.3 shows a snapshot of the Web Based Customizable Design client application. The source code for the client application can be found in Appendix B.1 and the source code for the design can be found in Appendix B.2.

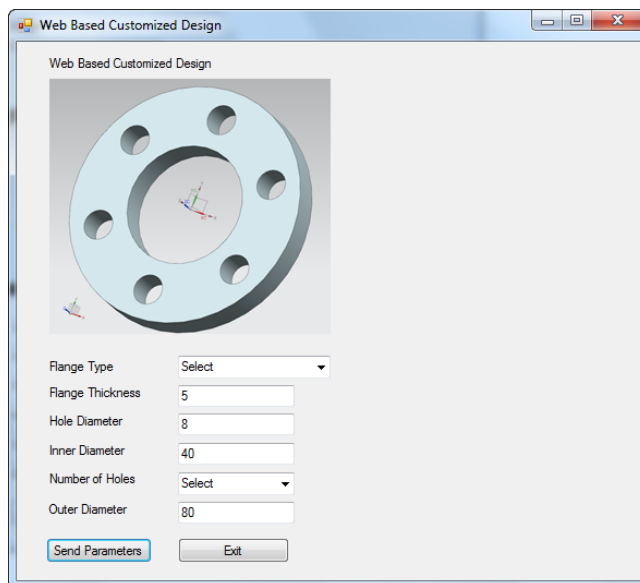


Figure 6.3: Graphical User Interface of Web Based Customizable Design

The GUI has several text boxes or drop-down lists where the user can input selected parameters. In the text boxes the user manually types the numbers and in the drop-down lists the user selects the numbers or other alternatives from the list. The different methods of inputs were added to the GUI to show some of the possibilities for limiting inputs. Figure6.4 shows the options for the “Flange

Type” drop-down list where the user either can select “Custom flange” or some per-made flanges with ISO standards. When selecting the ISO standards the numbers in the text boxes will automatically update to the numbers specified in the ISO standard so the user can see the exact numbers. The ISO standards used are discussed in Section 6.2.2.

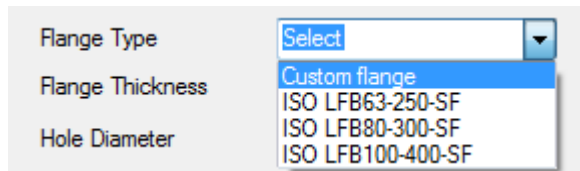


Figure 6.4: Selecting flange type in Web Based Customizable Design

The regular text boxes have built in rules for the numbers that are entered into them. When numbers that break the rule is entered the application will display a message box that says what rule is broken. Examples of these rules are that e.g. “Outer diameter has to be larger than Inner Diameter!” such as shown in Figure 6.5.

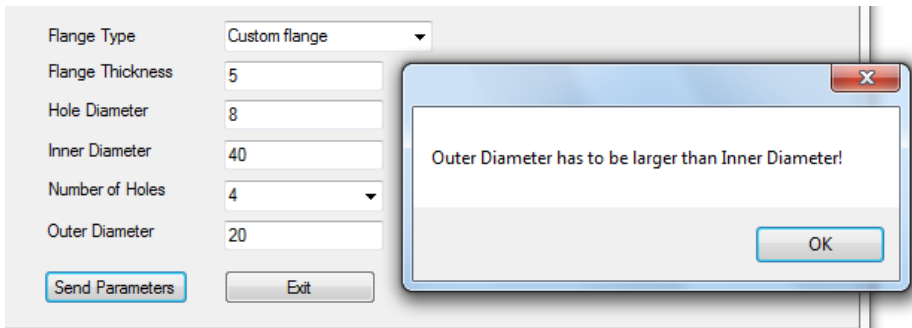


Figure 6.5: Design rules in Web Based Customizable Design example

Other rules included in the model will give the following error messages:

- Number of Holes cannot be larger than 12
- Number of Holes cannot be smaller than 1

- Hole Diameter must be smaller than 16 mm
- Hole Diameter must be larger than 1 mm
- Select Number of Holes (if nothing is selected in the “Number of Holes” drop-down list)
- Select Flange Type (if nothing is selected in the “Flange Type” drop-down list)

Other rules can be added, but the rules above were selected to show some of the possibilities to limit inputs. To show the range of the inputs simple labels can be put next to the text boxes and drop-down lists so the user easily can see what his possibilities are. This has not been implemented in the solution but Figure 6.6 shows an example of how it could be done.

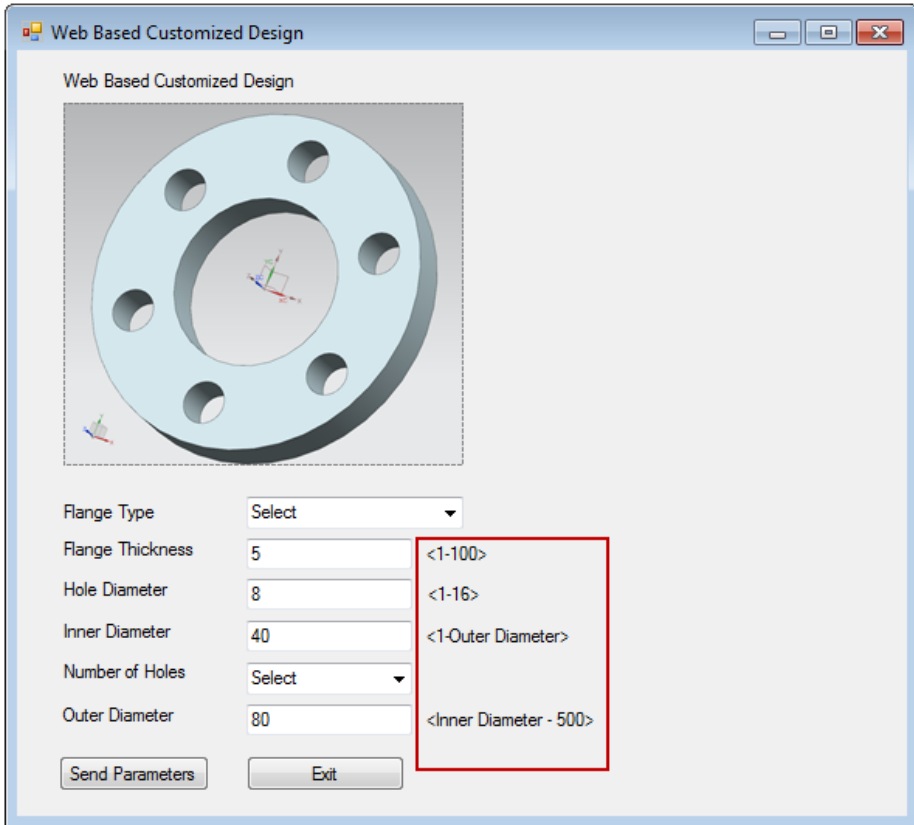



Figure 6.6: Adding range labels in GUI

The “Send Parameters” button sends the selected parameters to the server and waits for a conformation before downloading and opening the JT file. The “Exit” button shuts down the client.

### 6.2.2 ISO Standards for Example Flanges

To show that it is possible for the application to have pre-made examples of ISO standard flanges available in the client application three examples were made. The examples can be selected from the “Flange Type” drop-down list.

The examples used in this program are some of the ISO Socket Weld Flanges from “A&N Corporation ISO flanges and fittings” [29]



### ISO-LFB Socket Weld Flange

**Materials**

- Standard
  - 304 Stainless Steel
  - 316 Stainless Steel
  - Aluminum
- Alternative: Call for price and delivery
  - 316 Stainless Steel
  - Aluminum
- Options: (Call for price and delivery)

**Product Notes**

- Tapped bolt holes
- Internal Weld
- Self-Aligning
- Designed to fit standard inch tubing O.D.'s

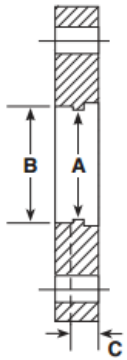


Figure 6.7: A&N Corporation ISO-LFB Socket Weld Flange

The following pre-made flanges were implemented in the model as examples [29].

ISO LFB Socket Weld Flange	LFB63-250-SF		LFB80-300-SF		LFB100-400-SF	
	inches	mm	inches	mm	inches	mm
Flange Thickness	0,5	12,7	0,5	12,7	0,5	12,7
Hole Diameter	5/8	16	5/8	16	5/8	16
Inner Diameter	2,5	63,5	3	76,2	4	101,6
Number of Holes	4		4		4	
Outer Diameter	3,74	95	4,33	110	5,12	130

Table 6.1: Pre-made LFB Socket Weld Flanges

When implementing the ISO standard flanges the model was already finished and wasn't complex enough to accurately depict the ISO flanges so the flanges found in the application are as close approximations as possible with the models complexity. Even though they are not exact models it shows that it is possible to have pre-parametrized options the user can select. Figure 6.8 shows what the output from the ISO standard flanges looks like. Note that the scale of the first and second flange is different than what it appears in the figure because scale is poorly visualized in JT2GO. However the user can take simple measurements in JT2GO like length and diameters to control the scaling.

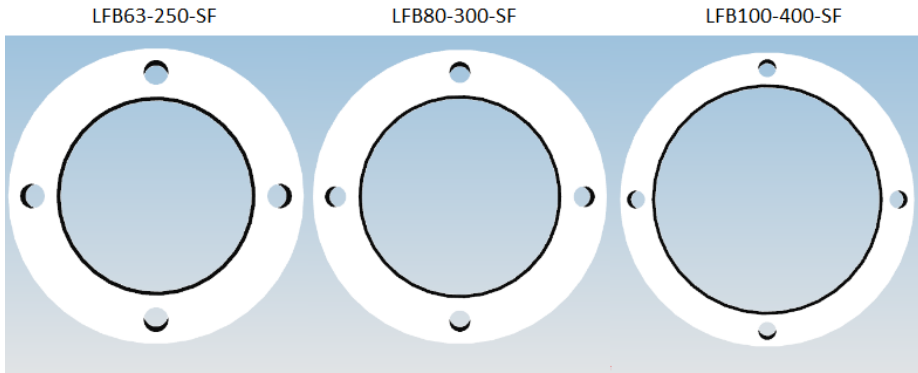


Figure 6.8: Results from WBCD using ISO standards

6.2.3 Sequence List of Client Application

### Sequence List of Client Application

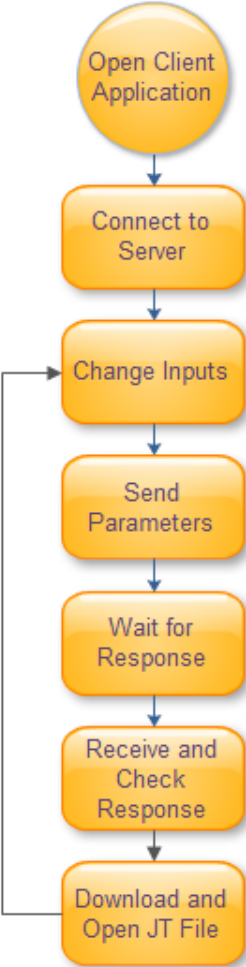


Figure 6.9: Sequence list of Client Application

## **Open Client GUI**

When the client is opened it sends a connection request to the server on a IP address and port number specified in the program. If accepted a socket connection is created between the two applications. This connection allows the client to declare a network stream so it can send bytes to the server and receive bytes from the server.

## **Change Inputs in GUI**

When selecting any of the pre-made ISO standard options from the “Flange Type” drop-down list the rest of the values in the text boxes below will change to the values of the ISO standard. This gives the user the possibility to see the values of the ISO standards.

## **Click “Send Parameters” Button**

This button controls the main function of the application. When this button is pressed the application executes its main functions. The session number is increased to match the request count in the Server Application. The parameters in the text boxes and drop-down lists are read and the values are checked. The parameters are finally sent as bytes to the Server Application.

## **Wait for Response from Server Application**

The Client Application listens on port 8888 for a response from the Server Application.

## **Receive and Check Response from Server Application**

When a response is received from the Server Application the message is checked to see if the request number at the end of the message is equal to the session number that was set when the user clicks the “Send Parameters” button. If the numbers match, the application knows that the Server Application is finished.

## **Download and Open JT File from FTP Server**

When the confirmation is received and checked the Client starts to download the JT file from the FTP server where the Server Application uploaded it. When the file is finished downloading it is stored and opened on the user system. The



file is opened in the default reader for JT files on the user system. Figure 6.10 displays examples of the results possible with the WBCD application.

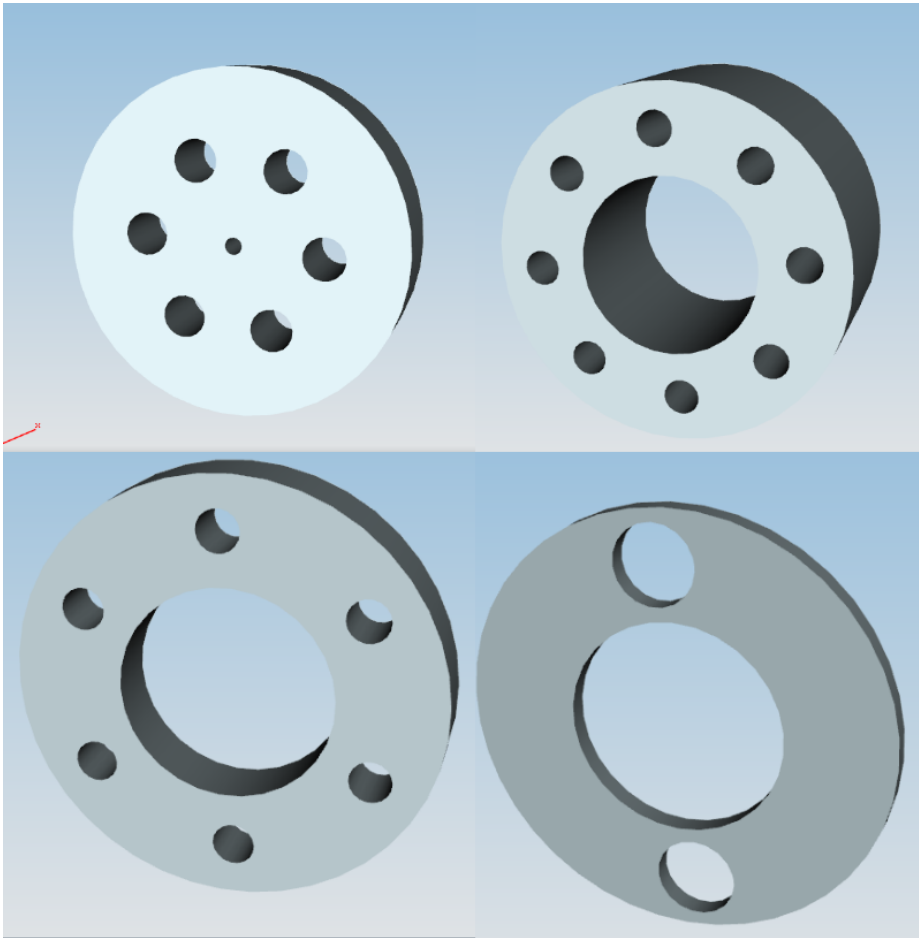


Figure 6.10: Customized flange examples

## 6.3 Server

### 6.3.1 Server User Interface

The server application is programmed as a Windows Console Application in Visual Studio 2008. This means that it has a text based user interface which runs in a console window on the server. When the server is running the user interface will look like Figure 6.11. The text lines displayed in the console window are not important for the application, but are there to give the user administrating the server application the possibility to see what requests are made. If an error occurs he can find the reason for the error by seeing what the latest displayed text line was.

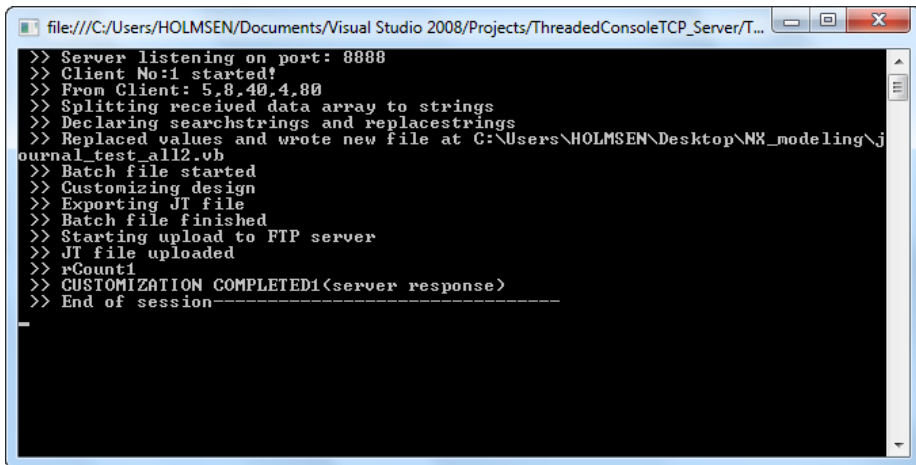
A screenshot of a Windows console window titled "file:///C:/Users/HOLMSEN/Documents/Visual Studio 2008/Projects/ThreadedConsoleTCP\_Server/T...". The window contains a list of log messages in a monospaced font. The messages are: ">> Server listening on port: 8888", ">> Client No:1 started!", ">> From Client: 5,8,40,4,80", ">> Splitting received data array to strings", ">> Declaring searchstrings and replacestrings", ">> Replaced values and wrote new file at C:\Users\HOLMSEN\Desktop\NX\_modeling\journal\_test\_all2.vb", ">> Batch file started", ">> Customizing design", ">> Exporting JT file", ">> Batch file finished", ">> Starting upload to FTP server", ">> JT file uploaded", ">> rCount1", ">> CUSTOMIZATION COMPLETED(server response)", and ">> End of session-----". The window has a standard Windows title bar with minimize, maximize, and close buttons.

Figure 6.11: Server Console Window

When the Server Application is running it is possible for different Client Applications to connect to it and the same Client Application can connect to the server several times to get several different models. The source code for the WBCD server application can be found in Appendix B.3.

### 6.3.2 Sequence List of Server Application

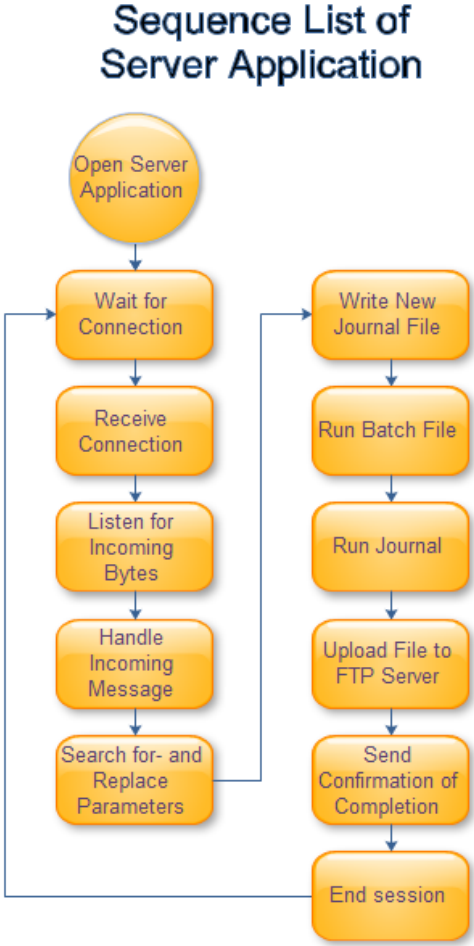


Figure 6.12: Sequence List of Server Application

## **Opening Server Application**

When opening the server application it starts to listen on the specified port for incoming connections from clients on the local IP address (meaning the IP address of the computer the server is running on). The user interface displays that the server is listening on port 8888 so all connections to the local IP address on port 8888 will be addressed by the Server Application. As long as the application is open it will listen for incoming connections.

## **Receive Connection from Client Application**

When a Client Application sends a connection request to the Server Application and the request is accepted it allows a TCP session to be created. The connection request is sent from the Client Application when it is opened. The server then assigns a number to this session to be able to identify it from other sessions. The message “>>Client No:1 started!” is displayed in the console window.

## **Listen for Incoming Bytes**

When a connection is made the server starts a Network Stream which is listening for messages sent from the Client Application. When a message is received it is in byte format and needs to be encoded to ASCII format to be readable as a text string.

## **Handle Incoming Message**

The text string is displayed in the console window as “>>From Client: 5,8,40,4,80” where the numbers 5, 8, 40, and 80 is the values that were selected in the Client Application GUI. These values are sent in a specified order so the Server Application knows which number is for which parameter. The values are separated by a comma in the text string and are separated by parsing the string between commas.

Position	Parameter name	Example value
1	Flange Thickness	5
2	Hole Diameter	8
3	Inner Diameter	40
4	Number of Holes	4
5	Outer Diameter	80

Table 6.2: Parameters received from client

Table 6.2 shows the ordering of the values that are sent from the Client Application to the Server Application and the values that are used in the example. Values will vary as the input values vary.

After the values are connected to each parameter they are added to an array (almost like a list) and the message “>>Splitting received data array to strings” is displayed.

### Search for- and Replace Parameter Values

Search strings are created for the server to know what to replace. Search strings are found in the journal file and are the code generated when changing the values of expressions. The string that replaces the search string is called a replace string. A message saying “>>Declaring search strings and replace strings” is displayed in the console window.

```
Dim searchStringFlangeThickness As String = _
"(expression1, unit1, ""\d+"")"

Dim replaceStringFlangeThickness As String = _
"expression1, unit1, "" & strFlangeThickness & """
```

Figure 6.13: Search string and replace string examples

Figure 6.13 shows an example of a search string can be seen, named searchStringFlangeThickness, and a replace string, named replaceStringFlangeThickness. The text “\d” makes the search string independent of a specific number (integer) in the journal text string so that any number can be used. This will ensure that a change in the journal file will not cause errors. The “& str-

FlangeThickness&” text takes the value for Flange Thickness that was received from the client and puts it in the right place.

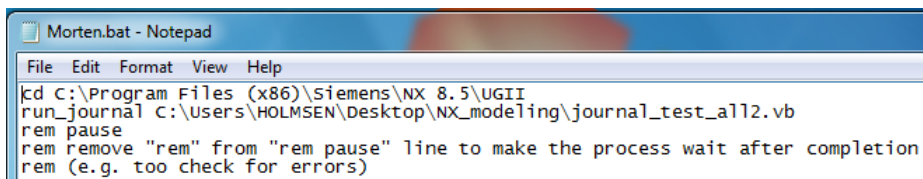
When the incoming message is handled and strings are declared the journal text file is imported to the application. It reads the journal text file to a string. The new values received from the client are inserted where the old values were in the journal file.

### Write New Journal File

After all values are replaced a new journal text file is created. This file is separate from the old journal file to avoid overwriting it. A message saying “>>Replaced values and wrote new file at newJournalFilePath” is displayed in the console window where “newJournalFilePath” is the file path of the new journal file.

### Run Batch File

The next step is to run a batch file. Using a batch file the server can start a command prompt (cmd prompt) window where it can change directory to the NX install folder and use the run\_journal command to execute the updated journal file. The text in the batch file can be seen in Figure 6.14. The text after “rem” is remarks and will not be executed.



```
cd C:\Program Files (x86)\Siemens\NX 8.5\UGII
run_journal C:\Users\HOLMSEN\Desktop\NX_modeling\journal_test_all2.vb
rem pause
rem remove "rem" from "rem pause" line to make the process wait after completion
rem (e.g. too check for errors)
```

Figure 6.14: Batch file used to run Journal file

After the Batch file is started the server displays the following messages

">>Batch file started"

">>Customizing design"

">>Exporting JT file"

After this the Server Application waits until the run\_journal process passes back an exit code which lets it know that the process is finished. After this message is received the command window displays ">>Batch file finished".

## **Run Journal**

The batch file executes the `run_journal` command in the NX install folder. This command starts a NX 8.5 session where it runs the journal file described in Section 5.4 “Journal”. When using this command the command prompt runs NX in a windowless mode which makes the process save time. The output of the journal file is exported as a JT file for visualization of the new model. When the journal finishes the batch file is stopped and sends the exit code needed for the Server Application to continue.

## **Upload File to FTP Server**

Before starting to upload the exported file the console window displays “>>Starting upload to FTP server”. The server connects to the FTP server by sending a FTP Web Request and connecting with specified credentials. The FTP upload method is based on a tutorial by [HowToStartProgramming.com](http://HowToStartProgramming.com) [30]. For this thesis a FTP server was set up by Summit Systems and its address was “ftp://ftp2.summit.no/NTNU/MBM”. After the upload is finished the console window displays “>>JT file uploaded”.

## **Send Confirmation of Completion to Client**

The console window displays the request count as “>>rCount1” as this is request number 1 in the example. For each request from the same application this count will increase. This count is important because it makes the client able to see which confirmation is from which session. The Server Application sends the message “CUSTOMIZATION COMPLETED1” because the request count is 1.

## **End Session**

Before the session ends the console window displays “>>End of session——  
———” to make a line so an admin can be able to easily separate the different sessions. After this message is sent the session with the client is terminated and the server goes back to listening for incoming connections.

## 6.4 Communication With Other Software

The sequence diagram in Figure 6.15 shows how the WBCD application interacts and communicates with other software. A sequence diagram is often used in computer science to describe how groups of objects collaborate [31]. Each participant (Client, Server, Batch file, NX 8.5 and FTP server) has a lifeline (vertical line) with an activation bar (vertical orange box) that indicates when it is active. An arrow from one participant to another is a interaction between the participants. An arrow that returns to its starting activation bar is a self-call (step that doesn't involve other participants).

Figure 6.15 shows a scenario where a user uses the WBCD application and describes what happens before he can open the file on his system.

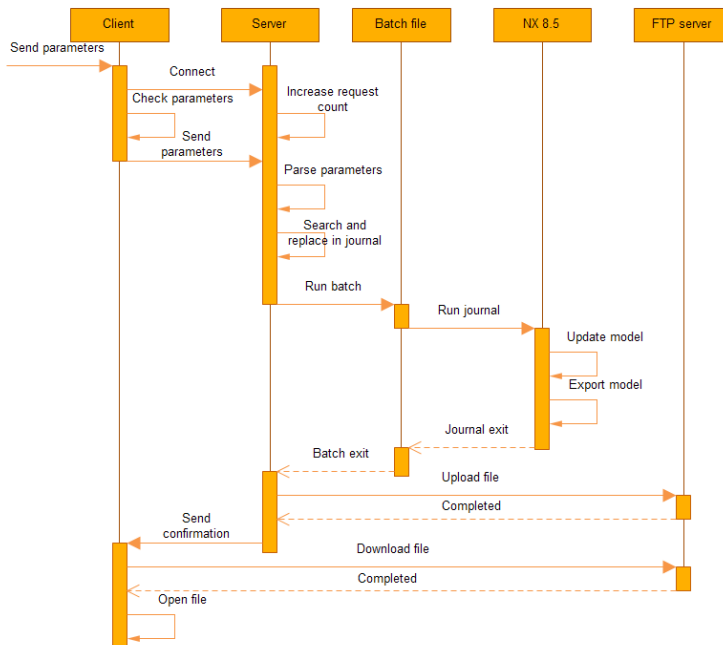


Figure 6.15: Sequence diagram of communication between software

The sequence is initiated when a user clicks the “Send Parameters” button in the WBCD client application. The client connects to the server and sends the



parameters after checking them. The connection between the server and client has been described in Section 6.1.

After the server has done its self-calls it starts a process of running the batch file. By starting a process that opens the batch file the server application can wait for an exit message to know when the batch file is finished. This means that the server is idle while the batch file is open.

The content of the batch file can be seen in Figure 6.14. The batch file runs the journal file in NX 8.5 as described in Figure 5.17. After the journal is finished the model has been updated and the result exported. The command window and batch file now closes and the exit message is sent to the WBCD server application. The server now knows that the file is ready to be uploaded to the FTP server.

The server application connects to the FTP server and uploads the updated model, and when completed sends a confirmation back to the client application. The client application then connects to the FTP server and downloads and opens the file to the users system.

## Chapter 7

# Using Web Based Customizable Design

This chapter will discuss some of the benefits a company can get by using the WBCD application, some of the limitations in the application and the work that needs to be done to implement the application on a different system.

### 7.1 Benefits

**Explicit Inputs Maintains Design Intent.** Using a simple graphical user interface with limited options available will makes the most important inputs explicit[7]. This will remove many possibilities for errors made by the users of CAD systems and will help maintain the creator of the model's design intent.

**Improves Consistency in Designs.** When different employees needs to use a part that is similar to an already existing part they might start from scratch and make one in their own way. This will in the end give a large amount of models that are supposed to be the same but are built in completely different ways. WBCD will keep the design in CAD models consistent.

**Hides Complexity of Internal Model.** When working with CAD models they can easily be very complex and hard to understand, especially when working on a model created by someone else which is often the case in large

companies. The need to understand the design knowledge of the creator of the model is removed. The complexity of the internal model is hidden for the user who can only see the simple UI in the WBCD application [7].

**Simple UI Accelerates Process.** When only presented 5 customizable parameters the process of changing the values is as quick and efficient as possible [7].

Design time can be reduced for products in a product family by quickly selecting requested features and dimensions.

**Productivity Savings From Using Parametric Models.** Firewire surfboards can by using CBD complete CAD and CAM models that are 98% completed including CNC tools that only needs some small finishing touches [2, 1].

**Remote Access.** As the client application uses the internet to communicate the server application gives the solution the ability to access a CAD model on the go which is great for making quick changes. Changes can be performed on the go by anyone, for example in a meeting and it doesn't require an installed CAD system.

**No CAD Experience Needed.** The application can be used by anyone who knows how to use a computer. No CAD experience is needed which makes the application perfect for e.g. sales personnel.

**Multiple Possible Usage Areas.** The application can either be used towards customers, internally in a company or both.

**Hides Secret Design Knowledge** A company wants customers to be able to order customized products that are limited by the rules for maintaining design intent, and they want the results of the design choices to be sent back to the customers. To avoid sending out a complete, heavy parametric model which contains the firms secret knowledge on design intent and details in how the parts are constructed they can use the WBCD application.

## 7.2 Limitations

As the application is programmed to establish a solution to the problem set in the thesis, focus has been on completing the process and not on making the solution robust in every possible way. This has led to some limitations in the program that will be discussed in this section.

The server application will only work when running on the system it was developed on. This is because the client uses a static IP address to connect to the server. The file paths that link to part files, JT files, journal files and batch files are also hard coded for the developing system. These variables can easily be changed in the developing code by changing the strings for file paths and IP addresses.

The user needs to have a CAD system or a JT viewer such as JT2GO [13] installed to view the result files.

The application only works for the specific model used in this thesis. When creating the search and replace algorithm the developer needs to figure out the expression names used and change the search strings to be able to update the model.

Only the parameters that are specified in the development of the model can be customized. Further customization of the model needs to be done in a CAD system.

The files are downloaded to a folder specified by the source code and it is not possible for the user to change the location of the file. This should be changed in the implementation of a generic solution.

## 7.3 Implementation

When implementing the solution on another system or for another model the entire solution would have to be tailored. This section discusses what needs to be done if this solution should be implemented for a new 3D model on a different system, not what should be done when developing a new, generic solution. Figure 7.1 shows what needs to be changed during implementation of the solution on a different system. This includes creating new 3D models and changing the source code for the application to adapt to the new system.

# Implementation

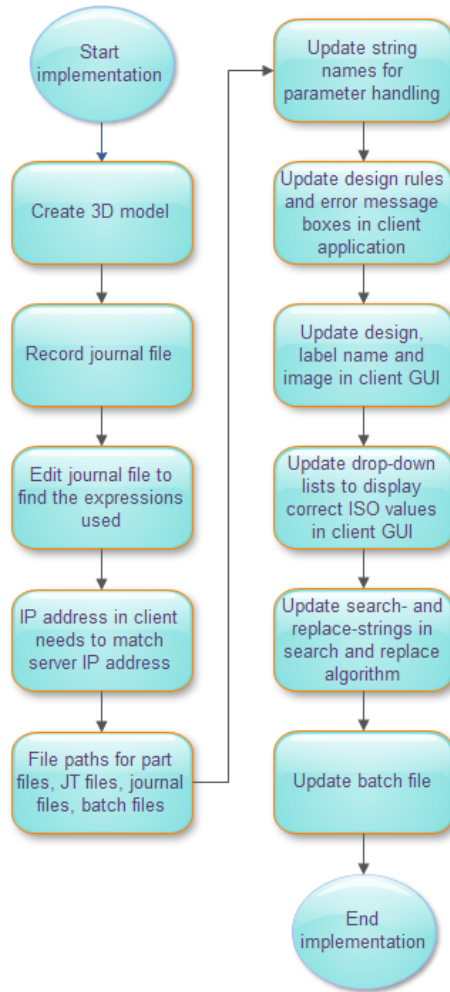


Figure 7.1: Implementation changes

# Chapter 8

## Discussion

### General

The method needed to develop a generic web based application to remotely customize a CAD model have been researched in this thesis. Different visualization formats had to be studied, along with the programming interface of NX to understand how to import data, update models, and export the results. A 3D model with customizable parameters had to be created, designed and used in PTS. A web based solution to configure the parameters had to be programmed.

The result was that a method for remotely customizing a CAD model was established. The problem stated that a generic solution was the goal, but this proved to be difficult to complete in the duration of a master thesis. An example solution has been created but it is limited to the system it is created on and the model it is based on. The solution is called Web Based Customizable Design (WBCD). To be able to use the method on other systems and for other products further development and customization is needed. Included in the solution is all of the requested functions for remotely customizing a 3D model. The most fitting visualization format for this thesis is JT but there are possibilities for including other export formats when customizing the application. Importing parameters, updating models and exporting results is possible by accessing NX through journaling. A guide for creating a model that can be used for the application is included in the thesis but the use of Product Template Studio (PTS) turned out to be excessive because changes made in PTS were not recorded in the journal file. The solution programmed consists of a server application run-

ning on a machine with NX 8.5 installed and a client application on a remote device. The user can choose parameters in the GUI and send them to the server which updates a 3D CAD model via journal files and uploads the result to a FTP server. The client downloads the result and opens the file for the user.

#	System Requirement	Status
1	The system should showcase possibilities for different types of input in the GUI	✓
2	The system can update CAD models	✓
3	The system should be generic and can be used on any model	✗
4	The system should be able to update different kinds of parameters	✓
5	The system can use pre-defined standard dimensions on models (ISO standards)	✓
6	The system doesn't require CAD experience to use	✓
7	The system can export updated CAD models in selected format	✓
8	The NX window is never opened in the process	✓
9	The system should store the result file on the user's system	✓
10	The server can handle multiple requests from users without being restarted	●
11	The system should be faster than modeling the part from scratch	✓
12	The client solution should be made as a web site	✗
13	The system does not require NX installed or licensed on the client	✓

✓ = completed

● = partially completed

✗ = not completed

Table 8.1: Overall system requirements status

### Generic

Creating a generic solution for this method proved to be difficult to complete given the limited time of a masters thesis. To be able to make a generic solution



there first needs to be developed a solution that later can be improved to be able to include functions so that it can be used for other models as well. By the time the development of the method and solution was finished there was not enough time to focus on improving the application and therefore the status of System Requirement number 3 in Table 2.2 is set as “not completed”. Creating a generic solution using the method described in this thesis is the next step in the list of further work.

### **Comparison with CBD**

Because the thesis is based on a previous solution by Firewire Surfboards, Custom Board Design (CBD) it is natural to compare the two solutions. The main difference is that CBD is a web page programmed in ASP.NET while WBCD is an application programmed in VB which gives the two solutions different functions and appearances. CBD is also more complex as the user can select and customize several different stock surfboards where as WBCD only uses one model.

When deciding how to structure the solution the options were to create either a web page or a client application to communicate with the server application that can run and update a NX model. Choosing a web site would mean that half of the solution would have to be programmed in VBScript or ASP.NET while the server application would be programmed in VB. Learning to code in a new programming language is a highly time consuming process. With almost no experience in creating web sites and the time it would take to learn a new programming language it was decided to not create a web site but create an application that would have the same function as a web site. This means that there is an extra installation process for the client application that would have been avoided if the website option was selected. This is why System Requirement number 12 in Table 2.2 is set as “not completed”.

When designing the WBCD client application the focus was on keeping it simple and intuitive. The design was greatly inspired by the CBD web site [5] where there is an image of the model to give an impression of what the part will look like and the customizable parameters are listed next to the image. It was important to give the parameters disambiguous names like the CBD web site does so the user easily can understand what parameter controls what function. The need to understand the design knowledge of the creator of the model is removed. It will be important to continue in the same style when implementing the solution for other models by keeping the design simple and

the parameter names disambiguous. A possibility can be to have a clickable link behind the different parameters that links to an image that describes or circles the parameter.

When designing the server application it was important that it could be running without needing any interaction and the easiest way to do this was to create a console application. If the system that is running the server is used for other purposes (not a dedicated Web Based Customizable Design server) the console application can be hidden so the server runs in the background. No information was available about how CBD's server worked and emails sent to Firewire and ShapeLogic requesting details were never answered.

The server can handle multiple requests from the same client in a row without being restarted but it crashes if several clients try to connect to the server at the same time. System Requirement number 10 in Table 2.2 is yellow because the requirement status is set as "partially completed"

After requests from Summit Systems the possibility for different type of input methods were added to the client design to showcase the possibility to both have text boxes (where the user has to type in the values) and drop-down lists (where the user selects a value from a list). This meant that additional checks had to be added to the application to check that the user had selected values in all of the drop-down lists. Summit Systems also requested that some pre-made examples were implemented in the solution to showcase the possibility to have a list of different available designs in a product family in the same model. This lead to the implementation of the ISO standards of the "A&N Corporation ISO flanges and fittings" [29] discussed in Section 6.2.2.

The possibility to choose different starting models in CBD is something that can be studied further and would benefit WBCD. This can be beneficial if a company has a product family that requires different CAD models to be used (such as the different stock models of surfboards) but they are so similar that having both options in the same application is logical. This works in CBD because all of the surfboards can be customized in the same ways (length, width, thickness). For this to work for WBCD the selected starting model would have to decide what customizable parameters should be displayed in the GUI.

The output format from the two solutions are also different. CBD uses 3D PDF files because their application is aimed at customers that don't normally use CAD programs. WBCD uses the JT format because the users are expected to have the possibility to view these files either with a free JT viewer such as JT2GO or a CAD system. The 3D PDF output was researched for the WBCD but after consulting Infinity Innovations it was decided to be too expensive and not deemed necessary to complete the application as it has different intended

user base. Summit Systems were clear on their recommendation to use JT as the visualization format for the thesis.

### **3D Model**

At the start of the project there were no requirements to the 3D model that would be used in the application. The factors that were important were that the model should be simple, that some of basic design rules were included, and that there were several customizable parameters. Choosing a flange was done to get a model with low complexity that everyone easily could understand and changes could be seen easily while still having several changeable parameters. The number of customizable parameters was chosen because it was the number of parameters needed to make the simple flange model and it meets the requirement to use different types of expressions (millimeters and constants).

### **Design Rules**

The design rules are one of the major benefits of using a product configurator such as WBCD. It allows the user to customize the model the way he wants within the limits of the model without being able to break the design intent of the model. Implementing design rules can be done in two different ways and both kinds should be used to obtain the best result. Rules can either be implemented in the model file in NX or in the GUI of the client application. The rules that are implemented in the model file in NX should be rules that makes the design unable to break the design intent (e.g. hole size larger than the flange diameter) and rules for placing components (e.g. hole placement between outer and inner diameter) while rules that control for example size of the component or if all features have been selected can be implemented in the GUI.

The difference between the two types of implementation is the point in the process where they are detected and their consequences. If the rules of the GUI are broken a message box immediately pops up to notify the user. These errors are easy to fix by not allowing the process to continue and make the user select new values and therefore has no large consequences. As many as possible of the rules needed should be applied in the GUI because these rules are there to make sure that the design rules in the model file in NX are not broken. If rules are broken in the model file, NX will fail when the application attempts to run the journal file. This will cause the application to stop and no result will be created

and will give major consequences for the user.

## **NX Programming Interface**

The programming interface in NX is described in the NX 7.5 documentation [21] but there is no clear guide available on how to start using it. Understanding how to access the programming interface of NX was done with a lot of trial and error. After consulting Summit Systems for how to access NX it was decided to use journaling because it is a good way to generate code by performing actions in NX without getting all the unnecessary overhead information a macro file creates. This allows the user to record a journal, and perform a specific action and then look at the code generated to see what code is connected to the actions. The reason for not using PTS when designing the model is that it is developed by an external company and the actions performed are not recorded by the journal used to customize the model and it is therefore incompatible with the WBCD solution. Appendix D discusses the code needed for this thesis. One of the problems encountered when testing the journaling was to be able to export JT files and use them in different locations on the system than where it was generated by NX. By default NX creates a JT file that references a sub-jt file in a folder where the original JT file is created. If only the JT file is moved without the associated sub-jt file it cannot be opened. By selecting a monolithic output the JT file stores all the geometry information and is able to visualize the model when moved to different systems.

## **Further Possibilities with Journaling**

When recording journals some further possibilities were discussed in Section 5.4. These were mentioned to inform readers that there are many more possibilities for making generic solutions by exporting lists from NX and creating methods that can import e.g parameter names automatically. The choice to not focus on these methods was made because this would branch out from the goal of establishing a method for customization of parameters, so even though it would bring interesting aspects into the thesis it was not prioritized.

## **Flow Charts**

When describing the method using flow charts it is difficult to explain everything in a single figure. Therefore the top-down approach has been used to focus on the current working environment to give more detailed flow charts in the later chapters.

## **Creating a Generic Solution**

A generic solution should be able to use different 3D models and change different parameters. To be able to create this the developer needs to find a way to create a dynamic user interface that adapts to the model selected by updating image and the available customizable parameters. A set of parameterized 3D models with associated journal files must be created for the different products in the generic solution. It is important to include information on which model the user selected in the client in the message sent to the server so the server knows what model to update and what journal to use.

Journaling works great for updating simple 3D models but when they start to get very complex there might be a need to use other methods of updating the models. The developer should research the possibility to use macro files instead of journaling so Product Template Studio could be included if that is required.

The developer of a generic solution would need knowledge on parametrization of 3D models in NX and programming skills in Visual Basic (including socket programming). Developing a new solution can either be based on the WBCD application source code or started from scratch and only consulting the source code to see what methods was used. It would be recommended to start from scratch if the goal is a generic solution but using the WBCD method.

## Chapter 9

# Conclusion and Further Work

### 9.1 Conclusion

This thesis explains the method for making a web based application to remotely customize a CAD model. A method for creating a 3D model and recording a journal to be able to change parameters has been described. An application has been created that is able to remotely customize 3D models to show that the method works. The application works on the system it was developed on and the model it was developed for but further work is needed if it is to be implemented on other systems. The solution consists of a server that runs on a system with NX installed that can update the model with the requested parameters sent from a user through a client application.

Different file formats for visualizing CAD models and how to access the programming interface of NX have been studied. Methods for importing data, updating models and exporting the results have been developed using expressions and journaling.

Benefits from the application includes maintained design intent in CAD models, reduced modeling time, gives remote access and removes the need for CAD experience and training for the end user.

The purpose of the thesis has been achieved by developing a method and creating a solution. Contacting companies after finishing the solution was not completed because of limited time.

## 9.2 Further Work

The focus of this thesis was to find a method for the problem, not to create a robust solution. Therefore the next step is to create a robust solution using the methods described in the thesis with a focus on being able to customize it for different CAD models with different parameters, making it a more generic solution. This includes researching the possibility to record functions used in Product Template Studio with macro files for more complex models.

Creating a web site instead of the client application is the next step to make the solution easily accessible for the user and remove the installation process. This also means that there is a possibility to use the solution in a web shop where customers can customize products themselves.

A real implementation for a specific customer requirement would possibly add some complexity to the solution, as other requirements could be introduced. Specific naming of files, feedback to the company who and when a customer downloaded a part, or other functionality that customers would look for in such a solution.

# Bibliography

- [1] Bruce Pettibone. Siemens looks at firewire's cbd. Available at: <http://vimeo.com/42175880>, 2011.
- [2] Bruce Pettibone. Serving up engineer-to-order surfboards. Available at: <http://www.knovelblogs.com/2011/10/04/ec-serving-up-engineer-to-order-surfboards/>, 2011.
- [3] SiemensPLM. Case study - nx cad technology drives custom surfboard design. Available at: [http://www.plm.automation.siemens.com/no\\_no/about\\_us/success/case\\_study.cfm?Component=123472&ComponentTemplate=1481](http://www.plm.automation.siemens.com/no_no/about_us/success/case_study.cfm?Component=123472&ComponentTemplate=1481), 2011.
- [4] Nev Hyman. Nev discussing the custom board design (cbd) platform. Available at: <http://vimeo.com/42239207>, 2012.
- [5] Firewire. Custom board design webpage. Available at: <http://custom.firewiresurfboards.com/>, 2013.
- [6] SiemensPLM. Jt open. Available at: [http://www.plm.automation.siemens.com/en\\_us/products/open/jtopen/](http://www.plm.automation.siemens.com/en_us/products/open/jtopen/), 2013.
- [7] Summit Systems. Introduction to product template studio (pts) author (nx7.5). Course catalog, April 2010.
- [8] Issac. Multi threaded server socket programming in vb.net. Available at: <http://www.java-samples.com/showtutorial.php?tutorialid=1064>, 2013.
- [9] SiemensPLM. What is plm software? Available at: [http://www.plm.automation.siemens.com/en\\_us/plm/index.shtml](http://www.plm.automation.siemens.com/en_us/plm/index.shtml), 2013.



- [10] Wikipedia. Visual studio. Available at: [http://en.wikipedia.org/wiki/Visual\\_Studio](http://en.wikipedia.org/wiki/Visual_Studio), May 2013.
- [11] Microsoft Developer Network. Visual studio resources. Available at: <http://msdn.microsoft.com/en-us/vstudio/cc136611.aspx>, May 2013.
- [12] Microsoft Developer Network. Consoles. Available at: [http://msdn.microsoft.com/en-us/library/ms682010\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ms682010(v=vs.85).aspx), Oct 2012.
- [13] SiemensPLM. Jt2go. Available at: [http://www.plm.automation.siemens.com/en\\_us/products/teamcenter/lifecycle-visualization/jt2go/index.shtml](http://www.plm.automation.siemens.com/en_us/products/teamcenter/lifecycle-visualization/jt2go/index.shtml), 2013.
- [14] Wikipedia. File transfer protocol. Available at: [http://en.wikipedia.org/wiki/File\\_Transfer\\_Protocol](http://en.wikipedia.org/wiki/File_Transfer_Protocol), May 2013.
- [15] Microsoft Windows Documentation. Using batch files. Available at: <http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/batch.msp?mfr=true>, May 2013.
- [16] Archdaily. Nx 8 from siemens plm software cad systems. Available at: <http://cad-systems.archdaily.com/1/20/NX-8>, Feb 2013.
- [17] PJ Jakovljevic. Siemens jt data format gets a nod from iso. Available at: <http://blog.technologyevaluation.com/blog/2013/01/29/siemens%E2%80%99-jt-data-format-gets-a-nod-from-iso/>, Jan 2013.
- [18] Anthony N. Godwin Richard M. Botting. Analysis of the step standard data access interface using formal methods. *Computer Standards & Interfaces*, 10:437–455, 1995.
- [19] Adobe. 3d solutions. Available at: [http://www.adobe.com/manufacturing/solutions/3d\\_solutions/](http://www.adobe.com/manufacturing/solutions/3d_solutions/), Feb 2013.
- [20] Productionmachining.com. Software enhanced to read nx part files. Available at: <http://www.productionmachining.com/products/software-enhanced-to-read-nx-part-files>, Aug 2010.
- [21] SiemensPLM. Nx 7.5 documentation, 2009.
- [22] Wikipedia. Macro (computer science). Available at: [http://en.wikipedia.org/wiki/Macro\\_\(computer\\_science\)](http://en.wikipedia.org/wiki/Macro_(computer_science)), May 2013.

- [23] Sham Tickoo. *NX 8.5 for Designers*. Cadcam Technologies, 2013.
- [24] SiemensPLM. Nx85 nxopen .net api reference, 2013.
- [25] Wikipedia. Client-server model. Available at: [https://en.wikipedia.org/wiki/Client%E2%80%93server\\_model](https://en.wikipedia.org/wiki/Client%E2%80%93server_model), May 2013.
- [26] Comer and Stevens. *Vol III: Client-Server Programming and Applications. Internetworking with TCP/IP*. Department of Computer Sciences, Purdue University,, 1993.
- [27] W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley Professional, 1993.
- [28] Wikipedia. Thread (computing). Available at: [http://en.wikipedia.org/wiki/Multithreading\\_\(software\)](http://en.wikipedia.org/wiki/Multithreading_(software)), May 2013.
- [29] A&N Corporation. A&n corporation iso flanges and fittings. Available at: <http://www.ancorp.com/familyPDFs/file725200881303AM63.pdf>, 2013.
- [30] HowToStartProgramming. Ftp upload. Available at: <http://howtostartprogramming.com/vb-net/vb-net-tutorial-26-ftp-upload/>, 2010.
- [31] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modelling Language*. Addison-Wesley, third edition edition, 2004.

# Appendix A

## Journal File

```

' NX 8.5.0.23
' Journal created by HOLMSEN on Mon Mar 11 18:01:43 2013 W. Europe Standard
Time
'
Option Strict Off
Imports System
Imports NXOpen

Module NXJournal
Sub Main (ByVal args() As String)

Dim theSession As Session = Session.GetSession()
' -----
'   Menu: File->Open...
' -----
Dim basePart1 As BasePart
Dim partLoadStatus1 As PartLoadStatus
basePart1 =
theSession.Parts.OpenBaseDisplay("C:\Users\HOLMSEN\Desktop\NX_modeling\Flan
ge_expressions.prt", partLoadStatus1)

Dim workPart As Part = theSession.Parts.Work

Dim displayPart As Part = theSession.Parts.Display

partLoadStatus1.Dispose()
Dim markId1 As Session.UndoMarkId
markId1 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Enter
Gateway")

Dim markId2 As Session.UndoMarkId
markId2 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Enter
Modeling")

' -----
'   Menu: Tools->Expression...
' -----
Dim markId3 As Session.UndoMarkId
markId3 = theSession.SetUndoMark(Session.MarkVisibility.Visible,
"Expression")

Dim expression1 As Expression =
CType(workPart.Expressions.FindObject("flange_thickness"), Expression)

Dim unit1 As Unit = CType(workPart.UnitCollection.FindObject("MilliMeter"),
Unit)

workPart.Expressions.EditWithUnits(expression1, unit1, "6")

Dim expression2 As Expression =
CType(workPart.Expressions.FindObject("hole_diameter"), Expression)

workPart.Expressions.EditWithUnits(expression2, unit1, "8")

Dim expression3 As Expression =
CType(workPart.Expressions.FindObject("inner_diameter"), Expression)

workPart.Expressions.EditWithUnits(expression3, unit1, "40")

Dim expression4 As Expression =
CType(workPart.Expressions.FindObject("number_of_holes"), Expression)

```

```

Dim nullUnit As Unit = Nothing

workPart.Expressions.EditWithUnits(expression4, nullUnit, "6")

Dim expression5 As Expression =
CType(workPart.Expressions.FindObject("outer_diameter"), Expression)

workPart.Expressions.EditWithUnits(expression5, unit1, "80
")

Dim nErrs1 As Integer
nErrs1 = theSession.UpdateManager.DoUpdate(markId3)

' -----
'   Menu: File->Export->JT...
' -----
Dim markId4 As Session.UndoMarkId
markId4 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Start")

Dim jtCreator1 As JtCreator
jtCreator1 = theSession.PvtransManager.CreateJtCreator()

jtCreator1.IncludePmi = JtCreator.PmiOption.PartAndAsm

jtCreator1.ConfigFile = "C:\Program Files (x86)\Siemens\NX
8.5\pvtrans\tessUG.config"

jtCreator1.JtfileStructure = JtCreator.FileStructure.Monolithic

jtCreator1.AutolowLod = True

jtCreator1.PreciseGeom = True

theSession.SetUndoMarkName(markId4, "Export JT Dialog")

Dim listCreator1 As ListCreator
listCreator1 = jtCreator1.NewLevel()

listCreator1.Chordal = 0.001

listCreator1.Angular = 20.0

listCreator1.TessOption = ListCreator.TessellationOption.Defined

jtCreator1.LodList.Append(listCreator1)

Dim listCreator2 As ListCreator
listCreator2 = jtCreator1.NewLevel()

listCreator2.Chordal = 0.001

listCreator2.Angular = 20.0

listCreator2.TessOption = ListCreator.TessellationOption.Defined

jtCreator1.LodList.Append(listCreator2)

Dim listCreator3 As ListCreator
listCreator3 = jtCreator1.NewLevel()

```

```

listCreator3.Chordal = 0.001

listCreator3.Angular = 20.0

listCreator3.TessOption = ListCreator.TessellationOption.Defined

jtCreator1.LodList.Append(listCreator3)

listCreator2.Chordal = 0.0035

listCreator2.Angular = 0.0

listCreator2.Simplify = 0.4

listCreator2.AdvCompression = 0.5

listCreator3.Chordal = 0.01

listCreator3.Angular = 0.0

listCreator3.Simplify = 0.1

listCreator3.AdvCompression = 1.0

Dim markId5 As Session.UndoMarkId
markId5 = theSession.SetUndoMark(Session.MarkVisibility.Invisible, "Export
JT")

theSession.DeleteUndoMark(markId5, Nothing)

Dim markId6 As Session.UndoMarkId
markId6 = theSession.SetUndoMark(Session.MarkVisibility.Invisible, "Export
JT")

jtCreator1.OutputJtFile =
"C:\Users\HOLMSEN\Desktop\NX_modeling\Flange_expressions.jt"
'jtCreator1.OutputJtFile =
"C:\Users\HOLMSEN\Dropbox\MASTER\Flange_expressions.jt"

Dim nXObject1 As NXObject
nXObject1 = jtCreator1.Commit()

theSession.DeleteUndoMark(markId6, Nothing)

theSession.SetUndoMarkName(markId4, "Export JT")

jtCreator1.Destroy()

' -----
'   Menu: Tools->Journal->Stop Recording
' -----

End Sub
End Module

```

## Appendix B

# Web Based Customizable Design Application Source Code

### B.1 Client Application Source Code

This is the code that controls the functions used by the WBCD client application as discussed in Section 6.2.1. The source code is also attached in the .zip file delivered on DAIM.





```

        MessageBox.Show("Outer Diameter has to be larger
                        than Inner Diameter!")
        button1Clicked = False
        Exit While
    End If

    If numberOfHoles > 12 Then
        MessageBox.Show("Number of holes cannot be larger
                        than 12!")
        button1Clicked = False
        Exit While
    ElseIf numberOfHoles < 1 Then
        MessageBox.Show("Number of holes cannot be smaller
                        than 1!")
        button1Clicked = False
        Exit While
    End If

    If numHoleDiameter > 16 Then
        MessageBox.Show("Hole Diameter must be smaller
                        than 16 mm!")
        button1Clicked = False
        Exit While
    ElseIf numHoleDiameter < 1 Then
        MessageBox.Show("Hole Diameter must be larger than
                        1 mm!")
        button1Clicked = False
        Exit While
    End If

    If numberOfHoles = "" Then
        MessageBox.Show("Select number of holes!")
        button1Clicked = False
        Exit While
    Else
        'MessageBox.Show("number of holes = " &
        'numberOfHoles)
    End If

    Dim flangeType As String = ComboBox2.SelectedItem
    If flangeType = "" Then
        MessageBox.Show("Select Flange type")
        button1Clicked = False
        Exit While
    ElseIf flangeType = "Custom flange" Then
        'do nothing
    ElseIf flangeType = "ISO LFB63-250-SF" Then
        ComboBox1.SelectedItem = "4"
        ComboBox1.Text = "4"
        FlangeThickness = "6"
        HoleDiameter = "8"
        InnerDiameter = "64"
        numberOfHoles = "4"
        OuterDiameter = "95"
    ElseIf flangeType = "ISO LFB80-300-SF" Then

```

```

        ComboBox1.SelectedItem = "4"
        ComboBox1.Text = "4"
        FlangeThickness = "6"
        HoleDiameter = "8"
        InnerDiameter = "76"
        numberOfHoles = "4"
        OuterDiameter = "110"
    ElseIf flangeType = "ISO LFB100-400-SF" Then
        ComboBox1.SelectedItem = "4"
        ComboBox1.Text = "4"
        FlangeThickness = "6"
        HoleDiameter = "8"
        InnerDiameter = "102"
        numberOfHoles = "4"
        OuterDiameter = "130"
        ComboBox1.Update()

    Else
        MessageBox.Show("Select Flange type")
        button1Clicked = False
        Exit While
        'MessageBox.Show("flange type: " & flangeType)
    End If
    'MessageBox.Show("flange type: " & flangeType)
    'Send message to the server
    Dim SendBytes As Byte() = Nothing

SendBytes = Encoding.ASCII.GetBytes(FlangeThickness +
    "," + HoleDiameter + "," + InnerDiameter + "," +
    numberOfHoles + "," + OuterDiameter + " ")
serverStream.Write(SendBytes, 0, SendBytes.Length)
'Read the NetworkStream (serverStream) into a byte
buffer
txtReceivedText.Clear()
Dim bytes(clientSocket.ReceiveBufferSize) As Byte
serverStream.Read(bytes, 0,
    CInt(clientSocket.ReceiveBufferSize))
'Output the data received from the host to the console
Dim returnData As String =
    Encoding.ASCII.GetString(bytes)
txtReceivedText.Text = txtReceivedText.Text +
    Environment.NewLine + _
"[Client]: " & returnData

'check if received text is "customization completed"
If InStr(txtReceivedText.Text, "CUSTOMIZATION
    COMPLETED" & parameterCounter) > 0 Then
    'MessageBox.Show("inside instr if")
    txtReceivedText.Clear()
    Dim jtFilePath As String =
"C:\Users\HOLMSEN\Desktop\NX_modeling\flange_expressions.jt"
    Dim ftpFilePath As String =
"ftp://ftp2.summit.no/NTNU/MBM/Flange_expressions.jt"
    Dim ftpDownloadedFilePath = "C:\New
Folder\flange_expressions_downloaded.jt"

```

```

        'This method downloads the file from the FTP
        server and opens it.
        My.Computer.Network.DownloadFile(ftpFilePath,
            ftpDownloadedFilePath, "ntnu", "master", False,
            100000, True)

        Process.Start(ftpDownloadedFilePath)
        txtReceivedText.Text = txtReceivedText.Text +
            Environment.NewLine + "[Client]: Opened JT file"
        txtReceivedText.Clear()
        'clientSocket.Close()
        button1Clicked = False
        Exit While

    Else

    End If

    Catch exc As Exception
        MessageBox.Show(exc.ToString)
    End Try
Else
    If Not serverStream.CanWrite Then
        txtReceivedText.Text = txtReceivedText.Text +
            Environment.NewLine + "[Client]: can not write
            data to this stream"
        serverStream.Close()
    Else

        If Not serverStream.CanRead Then
            txtReceivedText.Text = txtReceivedText.Text +
                Environment.NewLine + " [Client]: can not read
                data from this stream"
            serverStream.Close()
        End If
    End If
End While
Catch ex As Exception
    MessageBox.Show(ex.ToString)
End Try

End Sub

Sub connect(ByVal connectValue As String)
    Try
        msg("Client Started")

        'Declares IPAddress and port and tries to connect to server
        Dim localAddr As IPAddress = IPAddress.Parse("129.241.62.200")
        Dim intPort As Integer = "8888"
        clientSocket.Connect(localAddr, intPort)
        'Changes labell to ...
        'Labell.Text = "Web Basec Customized Design - Server Connected
    ..."

```

```

    Catch ex As Exception
        MessageBox.Show("Connection Failed." & Environment.NewLine & "No
            response from server", "Error")
        MessageBox.Show(ex.ToString)
        Exit Try
    End Try
End Sub

Sub msg(ByVal msg As String)
    'TextBox1.Text = TextBox1.Text + Environment.NewLine + " >> " + msg
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button2.Click
    clientSocket.Close()
    Me.Close()

End Sub

Private Sub ListBox1_SelectedIndexChanged(ByVal sender As System.Object,
    ByVal e As System.EventArgs)

End Sub

Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles TextBox1.TextChanged

End Sub

Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged

End Sub

Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Label1.Click

End Sub

Private Sub ComboBox2_SelectedIndexChanged(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles ComboBox2.SelectedIndexChanged
    Dim flangeType As String = ComboBox2.SelectedItem
    If flangeType = "ISO LFB63-250-SF" Then
        TextBox1.Text = "6"
        TextBox2.Text = "8"
        TextBox3.Text = "64"
        ComboBox1.SelectedItem = "4"
        TextBox5.Text = "95"
        Me.Update()
    ElseIf flangeType = "ISO LFB80-300-SF" Then
        TextBox1.Text = "6"
        TextBox2.Text = "8"
        TextBox3.Text = "76"
        ComboBox1.SelectedItem = "4"
        TextBox5.Text = "110"
    End If
End Sub

```

```
Me.Update()  
ElseIf flangeType = "ISO LFB100-400-SF" Then  
    TextBox1.Text = "6"  
    TextBox2.Text = "8"  
    TextBox3.Text = "102"  
    ComboBox1.SelectedItem = "4"  
    TextBox5.Text = "130"  
Me.Update()  
End If  
  
End Sub  
End Class
```

## **B.2 Client Application Designer Source Code**

This is the code that decides the design of the user interface of the WBCD client application as discussed in Section 6.2.1. The source code is also attached in the .zip file delivered on DAIM.

```

<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class Form1
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Dim resources As System.ComponentModel.ComponentResourceManager = New
            System.ComponentModel.ComponentResourceManager(GetType(Form1))
        Me.Button1 = New System.Windows.Forms.Button
        Me.TextBox1 = New System.Windows.Forms.TextBox
        Me.Label1 = New System.Windows.Forms.Label
        Me.Button2 = New System.Windows.Forms.Button
        Me.TextBox2 = New System.Windows.Forms.TextBox
        Me.TextBox3 = New System.Windows.Forms.TextBox
        Me.TextBox5 = New System.Windows.Forms.TextBox
        Me.Label2 = New System.Windows.Forms.Label
        Me.Label3 = New System.Windows.Forms.Label
        Me.Label4 = New System.Windows.Forms.Label
        Me.Label5 = New System.Windows.Forms.Label
        Me.Label6 = New System.Windows.Forms.Label
        Me.txtReceivedText = New System.Windows.Forms.TextBox
        Me.PictureBox1 = New System.Windows.Forms.PictureBox
        Me.ComboBox1 = New System.Windows.Forms.ComboBox
        Me.ComboBox2 = New System.Windows.Forms.ComboBox
        Me.Label7 = New System.Windows.Forms.Label
        CType(Me.PictureBox1,
            System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
        '
        'Button1
        '
        Me.Button1.Location = New System.Drawing.Point(27, 455)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New System.Drawing.Size(97, 23)
        Me.Button1.TabIndex = 0
        Me.Button1.Text = "Send Parameters"
        Me.Button1.UseVisualStyleBackColor = True
        '
        'TextBox1

```

```

|
Me.TextBox1.Location = New System.Drawing.Point(148, 315)
Me.TextBox1.Name = "TextBox1"
Me.TextBox1.Size = New System.Drawing.Size(107, 20)
Me.TextBox1.TabIndex = 1
Me.TextBox1.Text = "5"
|
|'Label1
|
Me.Label1.AutoSize = True
Me.Label1.Location = New System.Drawing.Point(27, 13)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(156, 13)
Me.Label1.TabIndex = 2
Me.Label1.Text = "Web Based Customized Design"
|
|'Button2
|
Me.Button2.Location = New System.Drawing.Point(148, 455)
Me.Button2.Name = "Button2"
Me.Button2.Size = New System.Drawing.Size(102, 23)
Me.Button2.TabIndex = 3
Me.Button2.Text = "Exit"
Me.Button2.UseVisualStyleBackColor = True
|
|'TextBox2
|
Me.TextBox2.Location = New System.Drawing.Point(148, 341)
Me.TextBox2.Name = "TextBox2"
Me.TextBox2.ShortcutsEnabled = False
Me.TextBox2.Size = New System.Drawing.Size(107, 20)
Me.TextBox2.TabIndex = 6
Me.TextBox2.Text = "8"
|
|'TextBox3
|
Me.TextBox3.Location = New System.Drawing.Point(148, 368)
Me.TextBox3.Name = "TextBox3"
Me.TextBox3.Size = New System.Drawing.Size(107, 20)
Me.TextBox3.TabIndex = 8
Me.TextBox3.Text = "40"
|
|'TextBox5
|
Me.TextBox5.Location = New System.Drawing.Point(148, 422)
Me.TextBox5.Name = "TextBox5"
Me.TextBox5.Size = New System.Drawing.Size(107, 20)
Me.TextBox5.TabIndex = 10
Me.TextBox5.Text = "80"
|
|'Label2
|
Me.Label2.AutoSize = True
Me.Label2.Location = New System.Drawing.Point(27, 315)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(91, 13)
Me.Label2.TabIndex = 11

```



```

Me.Label2.Text = "Flange Thickness"
'
'Label3
'
Me.Label3.AutoSize = True
Me.Label3.Location = New System.Drawing.Point(27, 341)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(74, 13)
Me.Label3.TabIndex = 12
Me.Label3.Text = "Hole Diameter"
'
'Label4
'
Me.Label4.AutoSize = True
Me.Label4.Location = New System.Drawing.Point(27, 394)
Me.Label4.Name = "Label4"
Me.Label4.Size = New System.Drawing.Size(86, 13)
Me.Label4.TabIndex = 14
Me.Label4.Text = "Number of Holes"
'
'Label5
'
Me.Label5.AutoSize = True
Me.Label5.Location = New System.Drawing.Point(27, 368)
Me.Label5.Name = "Label5"
Me.Label5.Size = New System.Drawing.Size(76, 13)
Me.Label5.TabIndex = 13
Me.Label5.Text = "Inner Diameter"
'
'Label6
'
Me.Label6.AutoSize = True
Me.Label6.Location = New System.Drawing.Point(27, 422)
Me.Label6.Name = "Label6"
Me.Label6.Size = New System.Drawing.Size(78, 13)
Me.Label6.TabIndex = 15
Me.Label6.Text = "Outer Diameter"
'
'txtReceivedText
'
Me.txtReceivedText.Location = New System.Drawing.Point(30, 43)
Me.txtReceivedText.Multiline = True
Me.txtReceivedText.Name = "txtReceivedText"
Me.txtReceivedText.Size = New System.Drawing.Size(228, 126)
Me.txtReceivedText.TabIndex = 7
'
'PictureBox1
'
Me.PictureBox1.Image = CType(resources.GetObject("PictureBox1.Image"),
    System.Drawing.Image)
Me.PictureBox1.Location = New System.Drawing.Point(30, 34)
Me.PictureBox1.Name = "PictureBox1"
Me.PictureBox1.Size = New System.Drawing.Size(258, 234)
Me.PictureBox1.TabIndex = 16
Me.PictureBox1.TabStop = False
'
'ComboBox1

```

```

'
Me.ComboBox1.FormattingEnabled = True
Me.ComboBox1.Items.AddRange(New Object() {"2", "4", "6", "8"})
Me.ComboBox1.Location = New System.Drawing.Point(148, 395)
Me.ComboBox1.Name = "ComboBox1"
Me.ComboBox1.Size = New System.Drawing.Size(107, 21)
Me.ComboBox1.TabIndex = 17
Me.ComboBox1.Text = "Select"
'
'ComboBox2
'
Me.ComboBox2.FormattingEnabled = True
Me.ComboBox2.Items.AddRange(New Object() {"Custom flange", "ISO LFB63-
    250-SF", "ISO LFB80-300-SF", "ISO LFB100-400-SF"})
Me.ComboBox2.Location = New System.Drawing.Point(148, 288)
Me.ComboBox2.Name = "ComboBox2"
Me.ComboBox2.Size = New System.Drawing.Size(140, 21)
Me.ComboBox2.TabIndex = 18
Me.ComboBox2.Text = "Select"
'
'Label7
'
Me.Label7.AutoSize = True
Me.Label7.Location = New System.Drawing.Point(27, 291)
Me.Label7.Name = "Label7"
Me.Label7.Size = New System.Drawing.Size(66, 13)
Me.Label7.TabIndex = 19
Me.Label7.Text = "Flange Type"
'
'Form1
'
Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
Me.ClientSize = New System.Drawing.Size(574, 494)
Me.Controls.Add(Me.Label7)
Me.Controls.Add(Me.ComboBox2)
Me.Controls.Add(Me.ComboBox1)
Me.Controls.Add(Me.PictureBox1)
Me.Controls.Add(Me.Label6)
Me.Controls.Add(Me.Label4)
Me.Controls.Add(Me.Label5)
Me.Controls.Add(Me.Label3)
Me.Controls.Add(Me.Label2)
Me.Controls.Add(Me.TextBox5)
Me.Controls.Add(Me.TextBox3)
Me.Controls.Add(Me.txtReceivedText)
Me.Controls.Add(Me.TextBox2)
Me.Controls.Add(Me.Button2)
Me.Controls.Add(Me.Label1)
Me.Controls.Add(Me.TextBox1)
Me.Controls.Add(Me.Button1)
Me.Name = "Form1"
Me.Text = "Web Based Customized Design"
CType(Me.PictureBox1,
    System.ComponentModel.ISupportInitialize).EndInit()
Me.ResumeLayout(False)
Me.PerformLayout()

```

```
End Sub
Friend WithEvents Button1 As System.Windows.Forms.Button
Friend WithEvents TextBox1 As System.Windows.Forms.TextBox
Friend WithEvents Label1 As System.Windows.Forms.Label
Friend WithEvents Button2 As System.Windows.Forms.Button
Friend WithEvents TextBox2 As System.Windows.Forms.TextBox
Friend WithEvents TextBox3 As System.Windows.Forms.TextBox
Friend WithEvents TextBox5 As System.Windows.Forms.TextBox
Friend WithEvents Label2 As System.Windows.Forms.Label
Friend WithEvents Label3 As System.Windows.Forms.Label
Friend WithEvents Label4 As System.Windows.Forms.Label
Friend WithEvents Label5 As System.Windows.Forms.Label
Friend WithEvents Label6 As System.Windows.Forms.Label
Friend WithEvents txtReceivedText As System.Windows.Forms.TextBox
Friend WithEvents PictureBox1 As System.Windows.Forms.PictureBox
Friend WithEvents ComboBox1 As System.Windows.Forms.ComboBox
Friend WithEvents ComboBox2 As System.Windows.Forms.ComboBox
Friend WithEvents Label7 As System.Windows.Forms.Label
```

```
End Class
```

## **B.3 Server Application Source Code**

This is the code that controls the functions and user interface of the WBCD server application as discussed in Section 6.3.1. The source code is also attached in the .zip file delivered on DAIM.

```

Imports System.Net
Imports System.Net.Sockets
Imports System.Text
Imports System.IO
Imports System.Text.RegularExpressions

'code from http://www.java-samples.com/showtutorial.php?tutorialid=1064
'for the threaded server

Module WBCD_Server
    Sub Main()
        Dim localIpAddress As IPAddress = IPAddress.Parse("129.241.62.200")
        Dim localPort As Integer = "8888"
        Dim serverSocket As New TcpListener(localIpAddress, localPort)
        Dim clientSocket As TcpClient = Nothing
        Dim counter As Integer
        Dim parameterCounter As Integer = 0

        serverSocket.Start()
        msg("Server listening on port: " & localPort)
        counter = 0
        While (True)
            counter += 1
            clientSocket = serverSocket.AcceptTcpClient()
            msg("Client No:" + Convert.ToString(counter) + " started!")
            Dim client As New handleCline
            client.startClient(clientSocket, Convert.ToString(counter))
        End While

        clientSocket.Close()
        serverSocket.Stop()
        msg("exit")
        'Console.ReadLine()
    End Sub

    Sub msg(ByVal mesg As String)
        mesg.Trim()
        Console.WriteLine(" >> " + mesg)
    End Sub

    Public Class handleCline
        Dim clientSocket As TcpClient
        Dim clNo As String
        Public Sub startClient(ByVal inClientSocket As TcpClient, _
            ByVal clineNo As String)
            Me.clientSocket = inClientSocket
            Me.clNo = clineNo
            Dim ctThread As Threading.Thread = New Threading.Thread(AddressOf
                doChat)
            ctThread.Start()
        End Sub
        Private Sub doChat()
            Dim requestCount As Integer
            Dim bytesFrom(10024) As Byte
            Dim dataFromClient As String
            Dim sendBytes As [Byte]()
            Dim serverResponse As String

```

```

Dim rCount As String
requestCount = 0

While (True)
    Try
        requestCount = requestCount + 1
        Dim networkStream As NetworkStream = _
            clientSocket.GetStream()
        networkStream.Read(bytesFrom, 0,
            CInt(clientSocket.ReceiveBufferSize))
        dataFromClient =
            System.Text.Encoding.ASCII.GetString(bytesFrom)
        'remove null characters from string
        dataFromClient = dataFromClient.Replace(vbNullChar, "")
        msg("From Client: " & dataFromClient)

        'declare array
        Dim arrayExpressions() As String

        'split array at ","
        arrayExpressions = dataFromClient.Split(",")
        Dim strFlangeThickness As String = arrayExpressions(0)
        Dim strHoleDiameter As String = arrayExpressions(1)
        Dim strInnerDiameter As String = arrayExpressions(2)
        Dim strNumberOfHoles As String = arrayExpressions(3)
        Dim strOuterDiameter As String = arrayExpressions(4)

        msg("Splitting received data array to strings")

        Dim strJournalFilePath As String =
            "C:\Users\HOLMSEN\Desktop\NX_modeling\journal_test_all.vb"
        Dim strNewJournalFilePath As String =
            "C:\Users\HOLMSEN\Desktop\NX_modeling\journal_test_all2.vb"

        'Dim batchFilePath As String = "C:\Program Files
            (x86)\Siemens\NX 8.5\UGII\morten.bat"
        Dim batchFilePath As String =
            "C:\Users\HOLMSEN\Desktop\NX_modeling\morten.bat"
        Dim jtFilePath As String =
            "C:\Users\HOLMSEN\Desktop\NX_modeling\flange_expressions.jt"
        Dim ftpFilePath As String =
            "ftp://ftp2.summit.no/NTNU/MBM/Flange_expressions.jt"

        '-----SEARCH AND REPLACE-----

        'Declare all searchStrings and replaceStrings

        Dim searchStringFlangeThickness As String = "(expression1,
            unit1, ""\d+"")"
        Dim replaceStringFlangeThickness As String = "expression1,
            unit1, "" & strFlangeThickness & """"

        Dim searchStringHoleDiameter As String = "(expression2,

```

```

        unit1, "\d+")"
Dim replaceStringHoleDiameter As String = "expression2,
    unit1, "" & strHoleDiameter & """"

Dim searchStringInnerDiameter As String = "(expression3,
    unit1, "\d+")"
Dim replaceStringInnerDiameter As String = "expression3,
    unit1, "" & strInnerDiameter & """"

Dim searchStringNumberOfHoles As String = "(expression4,
    nullUnit, "\d+")"
Dim replaceStringNumberOfHoles As String = "expression4,
    nullUnit, "" & strNumberOfHoles & """"

Dim searchStringOuterDiameter As String = "(expression5,
    unit1, "\d+")"
Dim replaceStringOuterDiameter As String = "expression5,
    unit1, "" & strOuterDiameter & """"
msg("Declaring searchstrings and replacestrings")

Dim objectReader As New StreamReader(strJournalFilePath)
Dim textboxJournal As String = objectReader.ReadToEnd

    textboxJournal = Regex.Replace(textboxJournal,
searchStringFlangeThickness, replaceStringFlangeThickness)
    textboxJournal = Regex.Replace(textboxJournal,
searchStringHoleDiameter, replaceStringHoleDiameter)
    textboxJournal = Regex.Replace(textboxJournal,
searchStringInnerDiameter, replaceStringInnerDiameter)
    textboxJournal = Regex.Replace(textboxJournal,
searchStringNumberOfHoles, replaceStringNumberOfHoles)
    textboxJournal = Regex.Replace(textboxJournal,
searchStringOuterDiameter, replaceStringOuterDiameter)

Dim strWrite As New StreamWriter(strNewJournalFilePath)
strWrite.Write(textboxJournal)
strWrite.Close()
objectReader.Close()

msg("Replaced values and wrote new file at " &
    strNewJournalFilePath)

'-----RUN BATCH WHICH RUNS JOURNAL
Dim objProcess As System.Diagnostics.Process
objProcess = New System.Diagnostics.Process()
objProcess.StartInfo.FileName = batchFilePath
objProcess.StartInfo.WindowStyle =
    ProcessWindowStyle.Normal

objProcess.Start()
msg("Batch file started")
msg("Customizing design")
msg("Exporting JT file")
'Wait until the process passes back an exit code
objProcess.WaitForExit()
'Free resources associated with this process

```

```

objProcess.Close()
'objProcess.Dispose()
msg("Batch file finished")

'-----UPLOAD JT FILE TO FTP SERVER
'this code uploads the JT file to the FTP server so it can
be downloaded by the client

'FTP Upload method based on this tutorial by How To Start
Programming
'http://howtostartprogramming.com/vb-net/vb-net-tutorial-
26-ftp-upload/

msg("Starting upload to FTP server")
Dim UploadRequest As System.Net.FtpWebRequest =
DirectCast(System.Net.WebRequest.Create(ftpFilePath),
System.Net.FtpWebRequest)
UploadRequest.Credentials = New
System.Net.NetworkCredential("ntnu", "master")
UploadRequest.Method =
System.Net.WebRequestMethods.Ftp.UploadFile

Dim file() As Byte =
System.IO.File.ReadAllBytes(jtFilePath)

Dim stream As System.IO.Stream =
UploadRequest.GetRequestStream()
stream.Write(file, 0, file.Length)

stream.Close()
stream.Dispose()

msg("JT file uploaded")

'-----send CUSTOMIZATION COMPLETED to client-----
'msg("From client" + clNo + dataFromClient)
rCount = Convert.ToString(requestCount)
msg("rCount" & rCount)
serverResponse = "CUSTOMIZATION COMPLETED" & rCount
sendBytes = Encoding.ASCII.GetBytes(serverResponse)

networkStream.Write(sendBytes, 0, sendBytes.Length)
networkStream.Flush()
'msg(serverResponse & "(server response)")

msg("End of session-----")

Catch ex As Exception
'MsgBox(ex.ToString)
End Try

End While

End Sub
End Class
End Module

```



## Appendix C

# Batch File to Run Journal

Listing C.1: Journal file

```
cd C:\Program Files (x86)\Siemens\NX 8.5\UGII
run_journal C:\Users\HOLMSEN\Desktop\NX_modeling\
    journal_test_all2.vb
rem pause
```

# Appendix D

## NX Open API

### D.1 NX Open API

This appendix will go through some of the NX Open API commands that are needed in the journal files by describing what some of the important commands do.

### D.2 NX Open API commands

This section will explain some of the commands that are used in the NX Open API (Application Programming Interface).

#### D.2.1 Imports

```
Imports System
Imports NXOpen
```

To be able to understand the commands that NX uses the journal file needs to import the System.dll and NXOpen.dll libraries.

## D.2.2 UndoMarkId “SetUndoMark”

```
Dim markId1 As Session.UndoMarkId markId1 = theSession
    .SetUndoMark (Session.MarkVisibility.Visible, "
    Expression")
```

This command creates a markId from the expression command which can be found in the “recent commands list” and also in the “undo” function

## D.2.3 Unit “FindObject”

```
Dim unit1 As Unit =
    CType(workPart.UnitCollection.FindObject("MilliMeter")
    , Unit)
```

This command creates an object for what kind of unit is used, in this case “MilliMeter”

## D.2.4 Unit “nullUnit”

```
Dim nullUnit As Unit = Nothing
```

This command declares a unit which is a constant that has no dimension.

## D.2.5 Expressions “CreateWithUnit”

```

Dim expression1 As Expression expression1 =workPart.
    Expressions.CreateWithUnits ("
    Created_Expression_by_MORTENMOI=999999999", unit1)

```

This command creates an expression and a value. “Created\_Expression\_by\_MORTENMOI” is the expression name and “999999999” is the value. CreateWithUnits specifies that an expression is created and states what unit is used. In this case it is unit1 which was declared to be “millimeters” in D.2.3.

## D.2.6 Expressions “EditWithUnit”

```

Dim expression1 As Expression =
    CType(workPart.Expressions.FindObject ("
    Created_Expression_by_MORTENMOI"), Expression)
Dim unit1 As Unit = CType(workPart.UnitCollection.
    FindObject("MilliMeter"), Unit)
workPart.Expressions.EditWithUnits(expression1, unit1,
    "888888888")

```

The first lines declares the expression by searching through the expression family class to find “Created\_Expression\_by\_MORTENMOI”.

The second paragraph declares the unit of the expression and the third paragraph edits the expression by expression name (expression1), unit (unit1=millimeter) and value (88888888)

## D.2.7 Session “GetSession”

```

Module NXJournal
Sub Main (ByVal args() As String)

```

```

Dim theSession As Session = Session.GetSession()
,
'INSERT CODE TO EXECUTE HERE
,
End Sub
End Module

```

When NX is running a session it needs to declare it and uses the command GetSession to connect to the session currently running.

### D.2.8 Open File

This is an example of code generated when a file is opened in NX 8.5

```

Dim basePart1 As BasePart Dim partLoadStatus1 As
PartLoadStatus basePart1 =

theSession.Parts.OpenBaseDisplay("C:\Users\HOLMSEN\
Desktop\NX_modeling\Flange_expressions.prt",
partLoadStatus1)

Dim workPart As Part = theSession.Parts.Work
Dim displayPart As Part = theSession.Parts.Display
partLoadStatus1.Dispose()

Dim markId1 As Session.UndoMarkId markId1 = theSession
.SetUndoMark(Session.MarkVisibility.Visible, "Enter
_Gateway")

Dim markId2 As Session.UndoMarkId markId2 = theSession
.SetUndoMark(Session.MarkVisibility.Visible, "Enter
_Modeling")

```

It starts of by loading the BasePart and Base Display from the file path of the opened file. After this it sets the current part to “work part” and “displayed part” and it updates the MarkId’s for entering Gateway mode and Modeling mode.

### D.2.9 Save File

```
Dim partSaveStatus1 As PartSaveStatus

partSaveStatus1 = workPart.SaveAs("C:\Users\HOLMSEN\
    Desktop\NX_modeling\Flange_expressions_save_test")

partSaveStatus1.Dispose()
```

This code is used to save the part. A new partname can be selected in the file path to save a new file that doesn’t overwrite an old file.

### D.2.10 Close File

```
Dim anyPartsModified1 As Boolean
Dim partSaveStatus1 As PartSaveStatus

theSession.Parts.SaveAll(anyPartsModified1 ,
    partSaveStatus1)

partSaveStatus1.Dispose()

theSession.Parts.CloseAll(BasePart.CloseModified ,
    CloseModified , Nothing)

workPart = Nothing
displayPart = Nothing
```

This code is used to save and close all open parts in NX.

## D.2.11 Export JT file

```
Dim markId1 As Session.UndoMarkId markId1 = theSession
    .SetUndoMark(Session.MarkVisibility.Visible, "Start
")

Dim jtCreator1 As JtCreator jtCreator1 = theSession.
    PvtransManager.CreateJtCreator()

jtCreator1.IncludePmi = JtCreator.PmiOption.PartAndAsm

jtCreator1.ConfigFile = "C:\Program_Files_(x86)\
    Siemens\NX_8.5\pvtrans\tessUG.config"

jtCreator1.AutolowLod = True

jtCreator1.PreciseGeom = True

theSession.SetUndoMarkName(markId1, "Export_JT_Dialog"
)

Dim listCreator1 As ListCreator listCreator1 =
    jtCreator1.NewLevel()

listCreator1.Chordal = 0.001

listCreator1.Angular = 20.0

listCreator1.TessOption = ListCreator.
    TessellationOption.Defined

jtCreator1.LodList.Append(listCreator1)

Dim listCreator2 As ListCreator listCreator2 =
    jtCreator1.NewLevel()

listCreator2.Chordal = 0.001
```

```
listCreator2.Angular = 20.0

listCreator2.TessOption = ListCreator.
    TessellationOption.Defined

jtCreator1.LodList.Append(listCreator2)

Dim listCreator3 As ListCreator listCreator3 =
    jtCreator1.NewLevel()

listCreator3.Chordal = 0.001

listCreator3.Angular = 20.0

listCreator3.TessOption = ListCreator.
    TessellationOption.Defined

jtCreator1.LodList.Append(listCreator3)

listCreator2.Chordal = 0.0035

listCreator2.Angular = 0.0

listCreator2.Simplify = 0.4

listCreator2.AdvCompression = 0.5

listCreator3.Chordal = 0.01

listCreator3.Angular = 0.0

listCreator3.Simplify = 0.1

listCreator3.AdvCompression = 1.0
```



```

Dim markId2 As Session.UndoMarkId markId2 = theSession
    .SetUndoMark(Session.MarkVisibility.Invisible, "
    Export_JT")

theSession.DeleteUndoMark(markId2, Nothing)

Dim markId3 As Session.UndoMarkId markId3 = theSession
    .SetUndoMark(Session.MarkVisibility.Invisible, "
    Export_JT")

jtCreator1.OutputJtFile = "C:\Users\HOLMSEN\Desktop\
    NX_modeling\Flange_expressions.jt"

Dim nXObject1 As NXObject nXObject1 = jtCreator1.
    Commit()

theSession.DeleteUndoMark(markId3, Nothing)

theSession.SetUndoMarkName(markId1, "Export_JT")

jtCreator1.Destroy()

```

Exporting a JT file creates a lot of code that should not be tweaked but the line below is where you can specify the output file path of the JT file.

```

jtCreator1.OutputJtFile =
    "C:\Users\HOLMSEN\Desktop\NX_modeling\
    Flange_expressions.jt"

```

It is important to select the exported JT file as a monolithic structure when it is going to be viewed at a machine that does not have access to the original NX part files of the model. The following line makes the JT model “monolithic” which means that all information and shapes are stored in the JT files, not just references to the part files.

```
jtCreator1.JtfileStructure = JtCreator.FileStructure.  
    Monolithic
```



## Appendix E

# Email from Infinity Innovations

Hei igjen  
Takk for info.

Selve 3d pdf teknologien er utviklet for Adobe av en 3.part

Vi kan levere en frittstående konverter som tar inn bl.a. jt opptil v 9.5 med BREP, Tessellated, PMI egenskaper avhengig av valgt internt format. Deretter saves dette som 3d pdf – det kreves også Acrobat Pro til dette.

Konverteren kan leveres med Acrobat Pro XI i pakken, men kan bestilles separat hvis man allerede eier Acrobat – oppgradering til v XI er også en opsjon.

Priser standalone :

Importereren u/ Acrobat kr 3.250

Importereren m/ Acrobat Pro XI kr 6.500

Prisene er eks mva, men inkl 1 års oppgradering/vedlikehold.

Det finnes også mange tilleggsprodukter for distribusjon/samarbeide etc av 3d pdf.

Vi kan også levere en testversjon hvis ønsket.

Mvh/Regards

Paul Batt-Rawden

Infinity Innovations Norway

[www.infinity.no](http://www.infinity.no)