TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: BladeDesigner Beta Testing Process & Getting Started Guide
TITULACIÓ: Enginyeria Tècnica Aeronàutica, especialitat Aeronavegació
AUTOR: Antillach París, Jordi
DIRECTOR: Lang, Sebastian
DATA: 8 de juliol de 2011
BACHELORARBEIT

BladeDesigner Beta Testing
Process & Getting Started Guide

Author: Antillach París, Jordi

Tutor: Lang, Sebastian
Start: 01/05/2011
End: 15/07/2011
Títol: BladeDesigner Beta Testing Process & Getting Started Guide

Autor: Antillach París, Jordi

Director: Lang, Sebastian

Data: 15 de juliol de 2011

Resum

Aquest treball representa la culminació de la feina de molts altres estudiants de la TUM que des de fa ja més de 3 anys han iniciat la construcció d'un programa de CAD per a dissenyar compressors d'ús aeronàutic per a motors d'avió.

L'objectiu principal d'aquest treball era dur a terme el procès de testeig beta del software per tal d'elaborar un informe de viabilitat per a l'entrega al públic general. Aquest procès havia de tenir en compte totes les funcions compreses en el programa així com l'interfície creada per a comunicar-se amb l'usuari. A més també es va estimar necessària la creació d'un manual d'iniciació per a usuaris nous als programa com a complement de la documentació que acompanya el programa.

El mètode de testeig emprat en totes les fases de test han set test de caixa negra. Utilitzant exemples de models 3D coneguts i recreant-los amb el programa per a posteriori comparar-los i trobar en quins punts el programa introduïx errors i deformacions en la representació o simplement assegurar que les funcions cridades compleixen correctament la funció per a la que han estat dissenyades.

La creació de la guia ha estat quasi inherent al fet d'entrar en contacte amb el programa. A través del descobriment de les funcionalitats i basant-me en la documentació addicional proporcionada per la resta de participants en el projecte he anat redactant cada pas que prenia dins del programa així com una definició precisa de tots els elements de la interfície, resumint-los finalment en una guia en forma de manual pas a pas.

La conclusió final malauradament denota que el programa no està llest per a distribuir-lo al públic ja que les funcions relatives a la construcció de geometria estan incompletes en la seva integració amb la interfície i les funcions de suport o secundàries introduïxen errors i desestabilitzen l'operació del programa. No obstant era una conclusió esperada ja que aquest era el primer cop que es duia a terme un procès de testeig sobre el programa i l'expectativa era trobar els errors que posteriorment hauran de ser corretjits.
Overview

This essay represents the culminating point of the work carried on by students of TUM university that during the last 3 years have initiated a development process to create a CAD tool to design compressors for aeronautical compressors to use in aircraft engines.

The main objective of this project was to perform a Beta testing process on the software in order to elaborate a release to public viability report. This process had to take into account all functions integrated in the program as well as the interface created to communicate with the user. Additionally, it was deemed necessary to create a getting started guide for new users to complement the documentation attached to the program.

The testing method used was based on black box tests. Using already known 3D models and recreating them with BladeDesigner to compare them and then establish which functions of the program induced errors to the process or defects in the models generated.

The creation of the guide has almost been a result of the learning process needed to understand the program to test it. Through the acknowledgement of functionalities and supported by the documentation provided by other team members, I've documented every step taken into the program as well as a thorough definition of the interface elements summarizing them in a step by step guide.

Unfortunately, the final conclusion points out that the program is not ready for public release. The geometry building functions integrated in the interface are incomplete and the secondary functions introduce errors to the building process and destabilize the proper functionality of the program. Nevertheless, this conclusion was expected as this was the first time that BladeDesigner underwent a testing process and the expected result was to find these bugs which will have to be corrected later.
# Table of contents

1. Thesis introduction ..................................................................................................11
2. Introduction to the program .....................................................................................13
   2.1. Profile generation ..........................................................................................13
   2.2. Blade generation ............................................................................................15
3. Test process ..............................................................................................................19
   3.1. Introduction ...............................................................................................19
   3.2. v.2010 ........................................................................................................21
      3.2.1. Test run 1 ............................................................................................21
      3.2.2. Test run 2 ............................................................................................23
      3.2.3. Test run 3 ............................................................................................25
      3.2.4. Conclusions v2010 ...........................................................................27
   3.3. v.2011 ........................................................................................................29
      3.3.1. Test run 4 ............................................................................................29
      3.3.2. Test run 5 ............................................................................................34
      3.3.3. Conclusions v2011 .............................................................................42
4. Getting started manual .............................................................................................44
   4.1. The interface .............................................................................................44
      4.1.1. Turbo machine main window ............................................................47
      4.1.2. Blade row main window .......................................................................49
      4.1.3. Profile main window ...........................................................................53
      4.1.4. SCMMesher window ..........................................................................55
   4.2. My first turbo machine .............................................................................57
   4.3. Export your work ......................................................................................61
   4.4. SCMMesher tutorial ...............................................................................61
5. Final conclusions .....................................................................................................64
6. Annexes ...................................................................................................................67
   6.1. Unit test ..................................................................................................68
   6.2. Troubleshooting guide ..........................................................................70
Bibliography ...............................................................................................................73
Index of figures

3.1 - Flow chart with the testing procedure .................................................................20
3.2 - Shell error message after attempted save ............................................................33
3.3 - Shell error when importing incorrectly named files .............................................36
3.4 - Shell error when importing a incorrect type of file .............................................37
3.5 - Shell error for incorrect Degree values ...............................................................38
3.6 - Shell error when CpFt values are not entered ....................................................39
3.7 - Shell error when trying to execute SCMMesher ..................................................40
4.1 - BladeDesigner interface layout ........................................................................45
4.2 - Turbo machine Parameters window ....................................................................47
4.3 - Turbo machine Plot window ................................................................................48
4.4 - Turbo machine Special window ..........................................................................49
4.5 - Blade row Parameters window ...........................................................................50
4.6 - Blade row Plot window .......................................................................................51
4.7 - Blade row Parameters plot window .....................................................................52
4.8 - Blade row Special window ..................................................................................53
4.9 - Profile Parameters page .....................................................................................54
4.10 - Profile Plot window ...........................................................................................55
4.11 - SCMMesher window ..........................................................................................56
Index of Tables

4.1 - Data example for the hub ................................................................. 58
4.2 - Data example for the shroud ............................................................... 59
4.3 - Data example for the stacking line .................................................... 59
4.4 - Example polynomial coefficients ...................................................... 60
4.5 - Example for the span ................................................................. 60
4.6 - Example values for the options parameters ........................................ 62
4.7 - Example values for the BlockCellsAx .............................................. 62
4.8 - Example values for radial parameters .............................................. 62
4.9 - Example options for CGNS output choices ...................................... 63
4.10 - Example values for the CpFt coefficients ......................................... 63
4.11 - Example values for the additional Nurbs parameters .................. 63
Nomenclature

Abbreviations

NURBS  Non Rational Uniform Base-Spline
CAD    Computer Aided Design
CFD    Computational Fluid Dynamics
STEP   Standard for the Exchange of Product Model Data
IGES   Initial Graphics Exchange Specification
STL    Surface Tessellation Language
XML    Extensible Markup Language

Variables

γ     Stagger angle
β1    Angle of attack
β2    Exit angle
Δs    Edge length of two-dimensional network elements
Δφ    Edge length of the three-dimensional network elements in the
      circumferential direction
Δψ    Translation compensation in the circumferential direction
zyl   Vector in cylindrical coordinates
kart  Vector in cartesian coordinates
D (x) Thickness distribution
Dmax  Maximum thickness
K     Number of network elements / element edges in the meridional direction
L     Number of network elements / element edges in the circumferential
      direction
p0, ..., p3, coefficients for polynomial thickness distribution
b0, ..., b4 coefficients for polynomial thickness distribution
2D    Two-dimensional vector
3D    Three-dimensional vector
h     Running index for nodes in the circumferential direction
m, y' Two-dimensional space into rotation coordinates area-related
r, φ, z Three-dimensional space in cylindrical coordinates
x, y, z Three-dimensional space in cartesian coordinates
g     Running index for node in meridian direction
\bar{P}^i    Control points of a NURBS curve
Chapter 1:

Thesis introduction

It is widely known that one of the most critical parts in any plane is the engines. The engines are also the component of an aircraft that determine the operational costs of it. The more efficient the less fuel it's needed to cover same distances. So to build a good plane, good engines are needed and by good we mean safe and economically viable. As modern turbofan and turbojet engines rely on thermodynamic cycles to obtain propulsion energy from the power of the air mass flow going through them it's important to meet the performance parameters needed to get the maximum efficiency while ensuring they provide enough power to keep the plane flying. The adequate pressure at the entry of the combustion chamber it's the key to make all the engine work and for that compressors are needed. A compressor is usually formed by several fans spinning at different speeds and stacked in cascade their main function is to increase the air's pressure as it passes through their blades. So as important it is to have good airfoils to ensure aircraft's lift, the adequate shape of the blade in the compressor allows the adequate pressure to be met.

Nowadays there are many CAD programs that help engineers design the shapes of the future engines and compressors and these 3D models are used in CFD simulators to help engineers correct and perfect these blades. But so far all these CAD programs are generic and can serve multiple purposes. BladeDesigner3D is a program specifically oriented to design 3D models of turbo machines and compressors with aims to provide testing samples for CFD simulators. BladeDesigner is a free software tool developed in the TUM by and for students. BladeDesigner is still a work in progress started a few years ago which required the collaboration of many students and teachers but as a free software licensed program anyone can use and collaborate in improving it.

BladeDesigner is a program based on python scripts but as the most common of the people are not versed in programming and working with shells and command lines BladeDesigner also integrates a Graphic User Interface developed to adapt the program to widespread public use. The program used different calculating methods to build a 3D geometry from the input the user feeds so there's no actual need from the user to define the geometry the program interprets the data to build the geometry desired. Using any method the program work in tree scheme. The turbo machine is the final element desired and it's composed of the different blade rows and each blade rows is composed of multiple blade defined using airfoil profile that conform a single blade. This way the user controls each step of the process from the smallest element to the final geometry. As the final purpose of the program is to create valid geometry to use in CFD testing the program also has built in special functions to export these geometries into compatible file formats for the most of the actual simulators.

Currently the last version released from BladeDesigner is a Beta version which means it's intended to be fully functional in all aspects but that hasn't been tested. Testing is critical for the success of any software in any market because it ensures that the product
delivered to the public is free of errors and that people will be able to use as it is explained in the manual. Assuming that the users of the program don't need any skills in programming a bugged version of the program will cause the users to be unable to operate the program and thus rendering all the work useless.

Beta testing is an important part of the quality process of any software. It's intended to explore all functions of the program and check their correct response in all possible situations given by the user. It is in fact a simulation of what the final user may encounter when faced with the program.

Generally software developers have several people testing the same version of the program at the same time to ensure that all kinds of possibilities are covered but as BladeDesigner is a small project there's only one Beta tester. The following study is the Beta testing process followed with the two latest versions of BladeDesigner. In it all functions and processes of the BladeDesigner software have been tested in all situations possible with all the variations available. The conclusions are a compendium of all bugs encountered, under which circumstances and the probable agents that caused them. All this information is summarized to provide the developers with accuracy which functions have to be modified or redesigned, which parts of the program aren't working and an overall perspective of the readiness status of the program for public release.

Another important goal of this study as the first Beta testing process done to BladeDesigner program is to provide a guideline for future testing procedures and also a checkpoint of the actual state of the program. As new versions are released this document has to be updated to the new versions and results have to be compared to check if progress is made in the right directions removing old bugs and not creating new ones.

This was also the first time that a student not related to the creation process was involved in the project and so his knowledge of the tool was null. This provided the perfect opportunity to create a guide on how to operate the program properly. This manual included within the program will transform BladeDesigner into a complete package of software ready to be released to people without need to train them in the use of the program previously. This guide was created while learning how to use the program so it's orientated to first time users to understand all the aspects of the software and how to use them for their own purposes.

The software and all information about BladeDesigner as well as contact data with the developers and on line help can be found open to the public in: http://sourceforge.net/projects/bladedesigner/
Chapter 2:

Introduction to BladeDesigner

BladeDesigner 3D is a CAD program designer to build geometry of turbo machines. The process that lead to geometry generation is rather complex and involves a series of calculations and plane interpolations which are built inside the program to facilitate the users work. In order to fully understand the functionality of the program and the reasons for all the data that it asks it's necessary to know these internal processes.

The design process starts by calculating the thermal and aerodynamic parameters of the turbo machine we want to implement. These data will provide us the information we need to determine the entry and exit angles and cross sections that the air flow must go through to match the calculations done for the inlet and outlet of the turbo machine. These angles and cross sections are the data that we'll need to generate the 2D profiles that will form our blades.

2.1 – Profile generation

This is the actual point where BladeDesigner starts working. Once the user has the data on the leading an trailing edge and the chord the profiles have to be drawn. First step is to discretize the profiles and allocate the points that will form the skeleton line along the one dimensional chord. To that purpose BladeDesigner uses a finite difference method, the tridiagonal matrix algorithm (TDMA). These method provides the program with the control points where the thickness distribution is going to be developed along the chord line. These methods are fully explained in [2] of the Bibliography.

To ensure a proper thickness distribution the allocation of points of the TDMA is combined with weighting functions. The weighting functions control the matrix algorithm and thus have also the control of the final control points. Each profiles needs an adapted distribution to achieve a trustful representation. Thus the program has implemented several different types of weighting functions which the user can change depending on the needs of each profile.

The basic method uses the curvature of the chord contour as guideline for weighting. This method shows good results but for profiles with high curvature is not enough as this method procures an evenly distribution. For profiles with high curvature around the leading or trailing edge the resolution is not enough. That why the contour method can be combined with weighting functions to maximize it's effectiveness. These functions include different types of conditions that can be switched to better match the needs of the current profile. Arc sine, tangent, hyperbola, cubic parabola, parabola and straight line are the conditions available for the weighting functions. These weighting functions can be combined with the basic method or used to substitute it if it better suits the accuracy of the final construction.
The weighted skeleton line points distributed will be the coordinates where the thickness functions will be applied. There are three methods to apply the thickness distribution to a profile.

- Power function
- NACA 65
- Elliptic

The power function methods is formed by these two functions:

$$D(x) = \frac{p_0(x^{p_1} - x^{p_2})^{\frac{1}{p_3}}}{D_k(x)} + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0$$  \hspace{1cm} (2.1)

$$D_k(x) = p_0(x^{p_1} - x^{p_2})^{\frac{1}{p_3}}$$  \hspace{1cm} (2.2)

The first function once assigned the correspondent parameters $p$ and $b$ and given $x$ which is the vector of points in the camber line assigned previously it returns the list of values of the thickness distribution.

The second function generates the radius for the nose of the leading and trailing edges. The derivative for $x$ of this function:

$$D_p(x) = b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0$$  \hspace{1cm} (2.3)

Establishes the contour for the thickness distribution

The NACA 65 method uses a standard NACA thickness distribution approximation. The input data needed is the $x$-vector containing the point from the weighted distribution in the camber line and a value for maximum profile thickness. Using the standard defined function:

$$D(x) = D_{max}(1 - x) \frac{-2,8385 x^3 + 2,4478 x^2 - 0,2758 x + 1,0675 \sqrt{x}}{1 - 0,176 x}$$  \hspace{1cm} (2.4)

The returned value is a list of values for thickness distribution along the points defined
for the weighting methods.

The elliptic thickness distribution method uses the general form of the ellipse:

\[
\begin{pmatrix}
  x \\
  y
\end{pmatrix} = \begin{pmatrix}
  a \cos t \\
  b \sin t
\end{pmatrix}
\]  

(2.5)

and combines these conditions:

\[
\Rightarrow t = \arccos \left( \frac{x}{a} \right) \Rightarrow y = D(x) = b \sin \left( \arccos \left( \frac{x}{a} \right) \right)
\]  

(2.6)

to get this function:

\[
D(x) = \frac{D_{\text{max}}}{2} \sin \left( \arccos(2x - 1) \right)
\]  

(2.7)

Which once established a value for the maximum thickness Dmax and given the x-vector of point distribution, the output of the functions is a the list of values for thickness distribution along the camber line points.

### 2.2 – Blade Generation

Once the profiles have been created the next step is to create the blades that will form the blade row. For a more thorough explanations on the following look [1] in the Bibliography. The idea behind is that each blade is formed by profile sections aligned in space. The profiles exist as a set of coordinates in a 2D space that can be place inside a 3D space without need to interpolate them. Thanks to this feature the only errors introduced in the process are mapping errors and not interpolation errors.

Previous steps to map the 3D surfaces of the blade are necessary to adapt the 2D profiles for use:

- Scaling of the profiles (determine absolute size)
- Rotation/Translation of the profiles (placement in space)
- Determination of the juncture point between the blade and the central ring
- Geometry of the central ring

The scaling of the profiles is determined by the chord length specifies in each one. All the profiles forming a blade are linked between them by a line intersecting them in their
center of mass which is called the stacking line. This line joins all the profiles in a single element and establishes the translation parameter for correct placement in space. The stagger angle of each profile establishes their rotation around the stacking line giving them the correct orientation in space. The juncture point or stacking point is defined by the placement of the first profile in space. The central ring geometry is user defined in the program.

The algorithms used to define all these parameters above is different depending on which type of turbo machine we want to create. The methods for an axial compressor will be different from the ones for a radial compressor.

With the coordinates for the profiles placement calculated the next step is to perform the mapping of the 3D surfaces. When dealing with 3D surfaces mapping distortion of the geometry is a common problem. BladeDesigner integrates two mapping methods to avoid this problem: the angle preserving method or the length preserving method.

In the length preserving method each profile coordinate point is transformed during the rotation and translation method to the 3D coordinate space using the following function:

\[
\begin{align*}
\bar{p}_{3D,zyi}^i &= \begin{pmatrix}
    r(m) \\
    \frac{y'}{r(m)} \\
    z(m)
\end{pmatrix}^i
\end{align*}
\]  

(2.8)

In this case the cylindrical surface of revolution is an isometric figure which also preserves the angles so no distortion is introduced in the process.

For the angle preserving method the goal is to achieve an angle conforming mapping of the 2D profiles. The complexity of the problem comes from the nature of the profiles which exist as a cloud of points. The solution is based on a networked surface shaped like a mesh that will be mapped over. This process is accomplished in five steps.

First step is positioning the profile. The 2D cloud of points must be transformed into a 3D space using the function:

\[
\begin{align*}
\hat{p}_{2D,m}^i &= \hat{p}_{2D,m} - \begin{pmatrix}
    \min(p_{1,2D,m}) \\
    \min(p_{2,2D,m})
\end{pmatrix}^i  \\
\end{align*}
\]  

(2.9)

Second step is to generate a 2D network suitable for meshing. Using the profile points a
2D grid is defined. The number of cells is given by the function:

\[ L = \left[ \frac{\max(p_{1,2D,m})}{\Delta s} \right] \]  \hspace{1cm} (2.10)

and the size of the cells is given by the function:

\[ \Delta s = \frac{\max(p_{1,2D,m}) - \min(p_{1,2D,m})}{K} \] \hspace{1cm} (2.11)

Third step is generating a 3D mesh from the 2D grid. To meet on a double curved surface of revolution the cell size must determined by the function:

\[ m^g = m^{g-1} + r \Delta \varphi \] \hspace{1cm} (2.12)

The properties of the imaging network are essential for the quality of conformal mapping because depending on the values of the parameters Δφ more or less severely distorted lengths can be shown. The correct numerical solution for this parameter comes form the function:

\[ \int_{m_0}^{m_{G-1}} dm \approx \sum_{g=1}^{G-1} (m^g - m^{g-1}) = f(\Delta \varphi) = \max(p_{1,2D,m}) - \min(p_{1,2D,m}) \] \hspace{1cm} (2.013)

Now the 3D coordinates of the nodes are calculated using the following function:

\[ \tilde{\mathbf{n}}_{z=yl}^{g,h} = \begin{pmatrix} r(m^g) \\ h \Delta \varphi + \Delta \psi \\ z(m^g) \end{pmatrix}^{g,h} \] \hspace{1cm} (2.14)
Fourth step is to indexate the coordinates. For this task we need to determine 4 component of each point using the functions:

\[
\begin{pmatrix}
  k \\
  l
\end{pmatrix}^i = \begin{pmatrix}
  p_{2D,m}^i \\
  \Delta s
\end{pmatrix}
\]  \tag{2.15}

\[
\begin{pmatrix}
  f_k \\
  f_l
\end{pmatrix}^i = \frac{p_{2D,m}^i}{\Delta s} - \begin{pmatrix}
  k \\
  l
\end{pmatrix}^i
\]  \tag{2.16}

Fifth and last step is to map the 3D grid. The profile coordinates in \( r, \varphi, z \)-space need to be translated into a \( z, y, l \)-space with the following function:

\[
\tilde{p}_{3D,zyl}^i = \tilde{n}_{zyl}^{k_i,l_i} \begin{pmatrix}
  f_k (n_1^{k_1+1,l_1} - n_1^{k_1,l_1}) \\
  f_l (n_2^{k_2,l_1+1} - n_2^{k_2,l_1}) \\
  f_k (n_3^{k_3+1,l_1} - n_3^{k_3,l_1})
\end{pmatrix}^i
\]  \tag{2.17}

Now the profiles are in cylindrical coordinates but with the transformation matrix:

\[
\tilde{p}_{3D,kerl}^i = \tilde{p}_{3D,zyl}^i 
\begin{pmatrix}
  \cos(p_{2,3D,zyl}^i) & \sin(p_{2,3D,zyl}^i) & 0 \\
  -\sin(p_{2,3D,zyl}^i) & \cos(p_{2,3D,zyl}^i) & 0 \\
  0 & 0 & 1
\end{pmatrix}^i
\]  \tag{2.18}

can be transformed into cartesian coordinates which can be mapped in 3D.

To create a whole turbo machine the process is repeated as many times as blade rows the user has specified switching the coordinates following the users parameters to create a full turbo machine geometry.
Chapter 3:

Test process

3.1 – Introduction to the test process

Before starting any testing it was necessary to design the testing procedure as this was the first time this was done on BladeDesigner. Having a defined test procedure is important because it defines the strategy to follow during the test and more important even it gives an example on how to realize further testing with other versions of the program. Moreover by repeating the same procedure after introducing changes in the code the developers can be sure that the bugs previously encountered are being solved.

The test procedure had to include every aspect of the program to be thorough and complete. It was designed to test every aspect prioritizing the critical aspects of the program like the building functions that are considered the cornerstone of the program as without these functions the rest are useless and the whole program is left without purpose. But as important as the correct operation of the functions is the readiness of the GUI as even is the functions are working properly on code it’s integration with the buttons of the visual interface can introduce errors to the procedure. So with this goal the testing procedure was divided in two aspects: interface and functionalities.

During the test procedure some bugs indicated the probability of further errors combining program options which hadn't been included at first on the testing schedule so the actual version of the testing procedure it’s a revised version. You can see the latest version on Figure 3.1. Eventually the test procedure was perfected while during the tests as new glitches and other probabilities of error were discovered. All the testing procedure can be summarized in the following flow chart which explains the steps taken during the testing in descending order of criticality for the program.
BladeDesgineer Beta Testing & Manual

Functionalities

Geometry construction

- Describing, building & saving a complete turbomachine
  - Drawing of the profiles
  - Construction of the blade row
  - Generation of the turbomachine
  - Check correct storage of the xml file

- Describing, building & saving a turbomachine with copied/cut/pasted elements

Import/Export capabilities

- Export & Check a turbomachine
- Export & Check a blade row
- Export blade 3D profile points
- Building a turbomachine with imported profiles

SCMMesher functionality

Interface

Environment friendliness

- Language of labels and menus
- LineEdits security & Program safeguards
- Buttons functionality

Figure 3.1 – Flow chart with the testing procedure
3.2 – Test runs on the 2010 version

The following pages contain the test reports written as result of the testing process on the 2010 version of BladeDesigner. Each report summarizes the tests runs made.

3.2.1 - Test Report #1: Getting started & Interface Testing

This first test is oriented to explore the limits and safeguards of the bladedesigner program against a new user with no knowledge of the code and only with the help of the BladeDesigner pdf document and a turbo machine example already done. This test belongs to the interface testing branch of the testing procedure. The objects under testing will be:

- Environment friendliness
- Line edits safeguards (against mis-entries, overload numbers, etc...)
- Proper functionality of all buttons
- Accuracy of the guide provided.

Object 1 – Environment friendliness

When first opening the program the GUI welcomes you with an already set turbo machine with one blade row and and one profile with parameters set to 0. This helps the user get started right away without searching any menus. The left-hand-side cascade distribution helps the user organize all blade rows and profiles at any time, differentiating them by number and color. The labels are correctly linked to the line edits so the user knows at every time what inputs is he giving. The tool bar gives easy and quick access to most common operations. Menu bar is simple and uses common terminology like in common Linux/Windows applications thus making it easier for the user to interact. All in all the GUI is friendly for first time users and easy to learn and use.

For non-German users it's possible to switch the language setting to English in the Settings menu tab. The translation is accurate in general minding some minor spelling mistakes and all the aspects of the program have been translated without mixing languages in any page.

A minor issue detected is that when trying to save the project for further study the program doesn't send an OK message confirming the action has been performed correctly. It does not affect the proper functionality of the program but it's an easy procedure that assures the user in all the steps he's taking. It does check if you want to save after closing or opening a new turbo machine which is good.

Object 2 – Line edits safeguards

To this matter line edits have been tested by trying to enter letters or impossible numbers as input values. For this purpose we've tried all the line edits in all the program (turbo machine/blade row/profile pages) and with all the method options available. The
testing method is simple: type words or symbols and try to operate the program. The program autodetects the letters and symbols different from +/- and is not possible to write them. As for overloaded numbers it's possible to enter them but when trying to calculate the program sends a warning informing about overloaded inputs and in which field they're located. This is a good and sufficient safeguard protocol for this kind of program.

Object 3 – Button functionality

In this test we used the turbo machine example to test only the proper functionality of the buttons without looking into the functions performance. To this matter the buttons in the tool bar, the menu options and the Calculate buttons in the blade row and turbo machine were part of the test.

First testing the menu bar all the options available operated on command as expected. All the options are fully functional. The tool bar buttons also operate correctly performing all the operations commanded on click. Both calculate buttons are well linked to their respective functions and initiate it's intended operations when clicked. So far this test indicates that all buttons in the program are well integrated to their respective functions in the code.

Object 4 – Guide's accuracy

The provided pdf guide offers an insight on the physics which the program uses to do it's calculations and a step by step guide to create a turbo machine with a rotor and a stator rows. The guide offers guidance to introduce the parameters to the input labels to reach a result properly. A major problem detected in the guide is that is based on a polynomial function for the profiles but inside the program there's is no such option for polynomial function (just power function, circular arc and Nurbs). This confuses the user and forces him to improvise which will surely lead to many errors and frustration of the user. Some relevant input data (like chord thickness) isn't defined through the guide and it needs to be guessed based on the pictures of examples This adds more confusion to the user who cannot be sure if it's doing the correct thing.

Conclusions

This first report has shown that the program's interface is well implemented and translated. In general it suits it's purpose (help the user interact with the code behind it) and works correctly in every aspect. Although this test shows that all buttons and menu options perform correctly their respective operations they have been tested separately without combining their use. This combined testing falls under the testing branch of functionalities.
3.2.2 - Test Report #2: Working with examples

This second test is the first test of the functionalities branch of the testing procedure. This test is prepared to test the program when attempting to build geometry. For this purpose I will be using the turbo machine example given by the tutor. Objects under testing will be:

- Creating a simple turbo machine
- Copying/Cutting/Pasting profiles and blade rows

Object 1 – Changing data on a existing sample

The first attempt of testing was to create a turbo machine from scratch using the instructions in the pdf manual. This was the first time during the testing procedure to create an original turbo machine not taken from examples. First issue detected in this test was the lack of a calculate button in the profile's parameters page. As result of this profiles aren't drawn and the rest of the operation cannot continue. The blade row's Calculate button doesn't activate the process and appears to be deactivated while in previous testing it showed a normal operation. Turbo machine's calculate buttons show the same behavior No attempt of profile generation was successful and further testing proved impossible by the inability to generate any geometry.

With a set example the user is able to override the lack of a calculate button for the profiles as the program automatically reacts to the new entered numbers and changes the shape of the existing profiles. That way and after several tries the program presented a coherent functionality and showed good results. Geometry for the blade row and turbo machine was created as in normal operation and the files were stored correctly. Only problem detected after over 10 different tries is related to the error messaging.

When entered some incoherent data between profiles the program warns you with a message box: “Fehler list index out of range”. This message is unclear of the error source and deactivates the calculate button until data is changed. By changing data meaning that any kind of change is prone to reactivate the button and of course if the change does not involve the source of error giving again the same message box The uncertainty pinpointing the source of the error is not a program error itself but it's an issue when trying to correct the data as the user if forced to check all data entered and search visually for the mistake.

Another incoherency detected in the program is about construction levels. The problems comes out when changing levels of action. When using high value date (in the case provided 170° for angle of attack) in the profiles the blade row is built normally. The 3D presentation is coherent with the data and the program issues no warning. Then the entire turbo machine is built and although the progress bar acts as normally when it reaches 100% and the user checks the plot it realizes that the figure given is not the expected (in fact it's still the same figure from before changing the values). When trying to repeat the process without changing the data the blade row construction now fails issuing the Fehler warning. I'm not sure where the problem comes from as for all the
tries made the blade row is built with no problems initially but after the turbo machine construction gives buggy results the second attempt to build the same blade row fails in all cases tried.

Object 2 – Copying/Cutting/Pasting blades and profile rows

After a fast set of testing we've been able to identify copy/paste options a source for major glitches. First thing noticed was that when using the feature to copy/paste both profiles or blade rows is that it causes a major glitch in the turbo machine's “calculate” button. In fact whenever the user tries to build a turbo machines containing copied and pasted elements he finds that the button is deactivated and unable to use it. The effect then is terminating for the programs operation has it cancels it's last function. The program is able to build and present blade rows coherent with the data given and the new copied profiles but when trying to finish the construction by creating a turbo machine it fails in all conditions tried (single blade row with copied profiles, multiple copied blade rows, multiple original blade rows with copied profiles in them). The other fatal glitch caused by the copy/paste functions is related to saving. Whenever a turbo machines containing copied elements (profiles or blade rows) is saved, the save fails as when the user tries to open it after working on a different profile the program opens a new turbo machine When checking the storage folder looking for the file with copied elements the user realizes it has never been saved and therefore is lost. The effect is directly related because after saving a file with copied and then saved another one without them the first is lost but the second one is saved normally. Even a previously saved turbo machine when reopened and added some copied element is deleted from the folder where it was correctly stored.

Conclusions

After this second battery of testing one major glitch has been detected. The problems arisen by the copy/paste functionalities are a major setback for the program as they affect directly the normal functioning of itself. By disabling the construction process of the turbo machines and disabling the save function of the program they rend it useless.

The incoherences problems detected with the first object tests, although important because they affect the functionality of the program, are less critic as they appear only for some values very close to the program's margin of operation which aren't really used in actual turbo machines parameters Still the program's inability to pinpoint more accurately the source of the problem is an issue which directly affects the interaction of the user with the software.
3.2.3 - Test Report #3: Other analysis methods

So far all testing procedure have been performed using the “power function” option for the camber line and thickness distribution. This third test is oriented to work with the other options and verify if the bugs found so far are consistent with all the other methods provided by the program. This is an extension to the previous test and so it falls under the category of functionality testing inside the testing scheme. To this purpose we will use the turboexample [4] and change it to suit the test. The objects under test will be:

- Test runs without combining methods
- Test runs combining methods

Object 1 – Test runs without combining methods

This first test run was oriented to test aspects from already tested in the test series #2 but using alternate methods for the chord and thickness distribution. As seen on test report 1 the inability to calculate a profile plot forces the user to use a already done example (turboexample in our case) and work by changing that example. Problems arisen during this test phase relate to this issue. While changing the base method of chord and thickness distribution the program resets the plot leaving the user without the capability to build a new plot and thus preventing further testing. The only exception is the Nurbs method which accepts changes from an already built profile without erasing it. Tests under this option have been consistent with the issues presented in test reports 1 and 2. The same functionality problems as described previously were found when using the copy/cut/paste function. This can lead to the conclusion that the bug with these functions affects in general all the other options of the program with a lot of certainty. Even so until all the calculation methods can be tested there is no way to check if this statement is true.

As for the method use itself I have to point out that this is the first time using the program options without a step-by-step guide. The documentation provided is very thorough with the power function method but with the other methods it's very theoretical and without examples. The user feels the need for a more complete guide to acquire good results. The geometry built is quite complex and small disagreements within the profiles can easily cause the final geometry to be impossible to build. Without proper guidance and error managing the user can find himself trapped in an error loop.

Object 2 – Test runs combining methods

Given the issues presented above the only combined methods able for using were the Nurbs method combined with the power function. Using this two methods for creating combined profiles the user is able to successfully build a blade row and the consequent turbo machine Interaction between methods is well implemented and the program raises the proper warnings whenever the constructions is not possible due to incoherences between profiles.
Again the same error pattern was repeated when using the copy/cut/paste functions. Leaving aside these issue we can determine that at least the two methods tested combine each other correctly and the safest guess is that the integration between methods is well implemented and won't affect the proper construction of the geometry. Yet again the conclusion cannot be truly confirmed until there's a possibility to check all the methods available in the program.

Conclusions

Though more was expected from this test the inability to successfully build profiles with the NACA 65 and double circular arc methods proved to be a drag to this test runs. This problem was already found in test runs #2 but overrode by using the [4] as an initial point to introduce changes. The program self starting protocol when changing methods rules out this possibility for the two methods mentioned above but not for the Nurbs method. This exception allowed some amount of testing though it also points out that the code is not coherent for all methods. This differences don't affect the normal operation of the program and haven't arisen any bugs or errors though. The test that we were able to perform showed consistency with the previous tests and confirm the bugs discovered so far (mainly the inability to create profiles and the malfunction of the save and build options when working with copied and pasted elements). This test runs also was the first approach to the different methods contained within the program. Although this is not related to the program itself the tester has to point out that the documentation given to work with the program is very scarce related to this methods and proves the program to be very hard to work for non experienced users. It is strongly suggested to create new documentation including a thorough explanation of these methods in order to help the user interact with the program more easily.
3.2.4 - Conclusions on the 2010 version

Once concluded the testing process for the 2010 version of BladeDesigner 3D we have to summarize the results and report the actual state of the program regarding readiness for public use. Starting from the most to the least critical bugs and errors encountered in this version from it's functionality point of view the following need to be taken into account.

The most critical part of the program is the geometry buildup that's the whole purpose of the program. In this way the inability to draw new profiles inside the program supposes the most critical bug encountered in this version. The whole process is impossible to be completed without the smallest elements in the chain thus rendering the program useless. Using pre-made examples the program showed however a normal operation and full functionality. This suggests that this critical problem may be solved easily by placing a calculate button in the profile's page to trigger the function to draw the profiles just like in the blade row or the turbo machine.

The next bug encountered is the malfunction caused by the copy/cut/paste functions of the program. We've concluded that the use of these function has a corrupting effect only on the XML file containing the turbo machine by deleting it as in cases where the geometry had already been built prior to introducing copied elements the geometry folder and it's documents present in the same folder as the main XML file was not affected at all. The copy/cut/paste options also affect the proper functionality of the program by disabling the calculate buttons and thus making geometry construction impossible. This issue is less critical as although it affects the core of the program's purpose can be avoided by simply not using those functions.

The bug detected when using data values close the program's operating margins affects the construction of the turbo machine because it escapes the safeguards implemented for user data entries. Although they directly affect the operations of the program it's an issue that can be solved easily by recalculating the data entry margin for the parameters of blade rows and profiles without need of any code change.

As for the interface bug report we have to conclude that the GUI introduces no errors from itself into the proper functionality of the program and everything works as it is supposed to when tested separately. All the problems encountered so far are caused by functions mis-integration inside the overall program. The most accurate conclusion is that the GUI will function properly as long as the bugs encountered in the functions programming are fixed.

In overall shape we regrettably have to admit that the program as it is now it's not ready for public use. The first issue presented cuts half of the calculating methods out of operation and forces the user to work with preset examples making him unable to create original work of his own. If we add the bugs introduced by the functions supposed to ease the construction (copy/cut/paste) and force the user to spend more time over a project that could be done simply we find that the program as it is isn't much of a help but an obstacle to the actual work of the user. Also taking into account the bugs arisen from overloaded data values the conclusion is that only a user with accurate data or
expertise in the topic would be able to use the program. The intent of the program isn't that really but to have an easy and accessible to all publics tool to design turbo machines. The final conclusion is that this version of the program is still not ready for public release.
3.3 – Test runs on the 2011 version

The following reports correspond to the test series realized on the 2011 version of the program.

3.3.1 - Test Report #4: Testing updated version

In this test I've just installed the newest version of the program which includes some modifications in the visual aspect as well as some additions in representations of the plots. The functionalities remain the same except the new additions and it is our goal to determine if this new version has corrected the bugs encountered so far in the previous version. As now we already have a testing procedure established we'll use it with aims to determine which improvements have been made since the last versions in matter of bugs. The new functionalities also had to be tested and meant a modification of the testing procedure to adapt it to them. The objects under test will be:

- Environment friendliness
- Comparison of functionalities between versions
- Study of the known previous bugs

Object 1 – Environment friendliness

Just as done in test 1 the first approach to a program is to determine if a new user with no previous knowledge could get his way around the program. Although the symbols and buttons have been changed and have a new appearance it's functionality remains the same so the same conclusion as test 1 still applies. There are new features such as new plot options and the export options which will be dealt with in the next test report. The idea of including a Manual in the program though it's still under work I guess it's excellent and will prove the ultimate step in creating a product able to be distributed without any need for user preparation or creation of additional documentation to complement the program for user readiness.

As for error protection the line edits have been re-tested under the same conditions as in test 1. It has been tried to enter either letters or symbols in all of the line edits to test the possibility of allowing the user to make a mistake. All line edits allow only entering one symbol (+/-) per line and numbers. Expected as this was already shown in the previous version of the program the program prevents the user from entering data which could cause error when passing the variables through the code.

Down through the hierarchy we repeated the same test with the parameter's page for the profiles. All line edit were tested against user mis-entries and the results were coherent with those obtained for the blade row parameter page. Only + and – symbols are allowed to be entered and numbers. This blockade helps careless users avoid simple mis-entry mistakes which result in errors and loss of work time. Result were coherent with all the possible options of calculation methods.
Object 2 – Comparison of functionalities between versions

This new version has the same functionalities as the older version plus some additions that will be studied in the next test run. Starting with the tool bar the buttons for adding a new profile, adding a new blade row and deleting items work properly and give no problems. This was already confirmed in the older version and is coherent that results stay that way. Profiles and blade rows are created and deleted correctly in new documents and existent document reopened.

Copy, Paste and Cut are also present in this version. Again testing over new turbo machines and existent ones reopened profiles are copied and cut normally and pasted correctly without losing or corrupting the data stored in them. However when copying or cutting entire blade rows the pasted element is corrupted. Either by performing one of those actions the original element remains the same (except when cutting because it's deleted) but the pasted elements based on the original though they maintain the blade row's parameters data have their inside profiles corrupted. When clicking those profiles contained within copied or cut blade rows the page remains blank without showing any of the labels or line edits supposed to contain and this profiles disable the use of the tool bar buttons on them. It's not possible to delete them nor add new profiles to the affected blade row. It's also not possible to copy or cut them neither moving them up or down. This although it shouldn't affect the overall operation of the program means that when creating complex turbo machines with several blade rows which may be very similar between them the user is unable to work on copies of the original blade row forcing him to take more time to create new blade rows and having to enter the data on all of them.

Copy, Paste and Cut are also present in this version. Again testing over new turbo machines and existent ones reopened profiles are copied and cut normally and pasted correctly without losing or corrupting the data stored in them. However when copying or cutting entire blade rows the pasted element is corrupted. Either by performing one of those actions the original element remains the same (except when cutting because it's deleted) but the pasted elements based on the original though they maintain the blade row's parameters data have their inside profiles corrupted. When clicking those profiles contained within copied or cut blade rows the page remains blank without showing any of the labels or line edits supposed to contain and this profiles disable the use of the tool bar buttons on them. It's not possible to delete them nor add new profiles to the affected blade row. It's also not possible to copy or cut them neither moving them up or down. This although it shouldn't affect the overall operation of the program means that when creating complex turbo machines with several blade rows which may be very similar between them the user is unable to work on copies of the original blade row forcing him to take more time to create new blade rows and having to enter the data on all of them.

Final block of the tool bar are the up and down buttons. This buttons are used to change the order of either profiles or blade rows inside the turbo machines hierarchy allowing the user to change the disposition of the turbo machine without having to delete the existent elements and having to create them again. For blade rows the buttons work perfectly. This has been tested with combinations of up to 5 blade rows in the turbo machine hierarchy. In all the cases the user was able to move any of the elements in any direction wished. For profiles the same was tested with up to 5 profiles inside a blade row. In this case the performance of the Up button was perfect allowing to move any profile desired upwards the hierarchy but the Down button seemed disabled. For any position in the hierarchy tried with profiles the down button didn't performed it's function as if the button was disabled. This didn't cause any crash on the code and has no impact on the programs main functions and is merely a disturbance when working but shows that the inside code differs from it's use in blade rows and profiles when the action performed is the same (inside coding it just means to move an object of any kind) which means that either the code is not well implemented or that there are two different functions depending on the object desired to move and one of them is failing. This kind of redundancy may cause problems elsewhere.

Functions in the menu stay the same with the addition of the Manual. In the Help tab both options open the respective windows correctly. The Manual is empty but that's not a problem of coding it just means that it needs to be written. Language setting work properly and for every window down the hierarchy (turbo machine, blade row and
profile) and every tab in every window (parameters and special, plot do not contain labels which need to be translated) the language switches correctly. View tab allows the user to pop up or down the tool bar and the hierarchy perfectly and the python shell option also works properly. Edit commands are the same as in the tool bar and when tried to interact to see differences between both none was discovered. The buttons in the Edit menu respond the same as in the tool bar arising the same exceptions and bugs.

In the File tab commands New and Close perform its function correctly creating new turbo machines with one blade row containing one profile with blank parameters and closing the program as commanded correctly. Function Open, Save and Save as involved a major bug with the previous version and so are going to be thoroughly tested in the next test object.

Object 3 – Study of known previous bugs

In the test reports from the older version a couple of critical bugs were identified which affected severely the proper functionality of the program. We have to determine if this bugs have been corrected in the new version.

The first bug was related to the functions Copy, Cut and Paste. As seen in the older version working with copied or cut and pasted elements (both profiles and blade rows) the calculate buttons in both blade rows and turbo machines were disabled preventing the user to create the geometry. As the purpose of the program is to create this geometry this bug disables the whole program and is a critical glitch. In the new version we've several scenarios involving this functions. First only copied profiles we're tested. Using two original profiles with data from the documentation we copied one and pasted it and cut the other and pasted it. When trying to create the geometry result was the same as in the older version, both buttons acted as disabled and the program was unresponsive. We tried again only with one copied profile and again with only one cut element. In both cases result was the disabling of the calculate buttons. Then we tested the same but with blade rows. Using two original blade rows containing only one original profile each we used the same method as with profiles. First combining one copied and pasted and once cut and pasted blade rows. In this case blade rows were able to be created normally and the geometry generated was correct but when proceeding to create the turbo machine the button was disabled as in the other cases. We followed trying only with one copied blade row and then with only one cut blade row. Result was consistent with the first try, blade rows were created as in normal operation but the turbo machine's calculate button was disabled. Then we repeated all the process but working with copied elements from different files copying or cutting them from a known working turbo machine and pasting them in a new one for the purpose of test. Results again showed that the calculate buttons were disabled in all cases. This has led to the theory that the copy and cut functions move the bug up the hierarchy because when copying profiles both the blade row and turbo machine calculate button were disabled but when copying blade rows we were able to create the geometry but the turbo machine calculate button was disabled.

The second bug was also related to the Copy and Cut functions and the Save functions. It was discovered that when dealing with turbo machines containing pasted elements the
save function was corrupted. The result was not only that the turbo machines wasn't saved but that if already stored in the hard drive and then reopened to add a copied element when attempting to save the XML document was deleted. To test this in the new version we've used the turboexample given in the documentation [4] and saved it in another folder as backup version. With this version we've tried to save it again changing only data in the parameters without copying anything. This way the altered version of the example was saved without problems and was available to access correctly later on. The we reopened the backup version and added a copied profile. Using first only the Save function the result was that the XML file was deleted and when attempted to open the program automatically started a new turbo machine without data. We repeated this test this time using the Save as function and the result was the same, XML file was lost. We repeated both tests using copied elements from different turbo machines than the one we were trying to save and the result was the same. This behavior gave us an idea and we tried a different test. Using only original profiles we used the hot keys combinations Ctrl+c and Ctrl+v to copy and paste only data from the line edits. The results were ambiguous. When using this combination and attempting to save without building the geometry the result was the same as if we had copied elements inside the turbo machine thus deleting the XML file and losing the turbo machines But when using this hot keys combinations to copy data and then building the geometry prior to saving the function worked normally the XML file was saved correctly and was able to be accessed later on. In this case building the geometry prior to saving allowed us to avoid this bug and returned the program to it's nominal way of working. Unfortunately this hot keys don't work with elements from the hierarchy tree so this partial fix isn't useful the solve the prior bug. Looking at the shell we see the feedback from Figure 3.2 when a failed save occurs:
Conclusions

After extensively testing this new version we have to conclude that it hasn’t solved much problems. We have to consider that this version was released before the beta testing process began so the developers weren’t aware of the bugs present and the purpose of this new version was to add more functionalities to the program. The two major bugs spotted in the first test runs are still present in this new version and will have to be dealt with if the program is to reach it’s full potential. There are also some new minor glitches that don't affect the proper function of the program (Object 2) but were not present in the older version which suggests that the new functions added to the program might have created some incompatibilities with those functions. This suggests a revision of the new implementation in order to achieve at least the same level of program stability as in the older version.

Figure 3.2 – Shell error message after attempted save
3.3.2 - Test Report #5: New functions v.2011

This is the last test report for the BladeDesginer software. In this test the objective is to test the new functions added in the 2011 version of the program. These new functions include the export/import features and the SCMMesher. As the whole purpose of the program is to create geometry that will be used later in CFD simulators the program would not be complete without the capability to translate the objects created into a file format compatible with these programs. Some programs are able to create their own meshing schemes based on a geometry given that's why BladeDesigner has the export capability for geometry. Other programs need an already designed mesh to work on and this is why there's also the SCMMesher tool which creates this mesh in a compatible file format for CFD simulators. The tests were run over a previously done turbo machine saved and stored from other tests. The program used to verify the exports was SolidEdge which has import capability for IGES, STEP and STL file formats. This test will cover the two last items of the functionalities branch in the testing scheme which were added when this new version of the program was released. The objects under test in this report will be:

- Import/Export capabilities
- SCMMesher functionality

Object 1 – Import/Export capabilities

The import/export capabilities are stored in the tab named Special and only available for blade rows and turbo machines as they are the only objects built in 3D. When a geometry is built in BladeDesigner the resulting object is saved in a folder named geometry inside the folder containing the file and with .XML file format. Now the common file formats for geometry compatibility between CAD programs are IGES or STEP and BladeDesigner's exports are done in those formats. The first test was to determine the export capability for turbo machines. The only options for this purpose are to export in IGES and/or STEP and no import capability is implemented. The program allows the user to check all the options available to make all the exports at once. With the geometry built in program 3 tests were run. One for each export option and one with both at the same time. The export elements was a turbo machine consisting of one blade row with 5 profiles. The data used to define the turbo machine can be found in the Getting started guide.

In all the cases the program showed a successful export message and the files were correctly stored inside the geometry folder of the corresponding file. Both files were checked using CAD program SolidEdge and the geometry imported in both file extensions was correct and showed no signs of corruption during the process. The function to export turbo machines works correctly in all cases tested.

The blade row's Special tab houses the import/export functionalities for both the blade geometry and the 3D profile points. The blade geometry only has the export function implemented but several more options of export. As in the previous case each options was tested individually and then all at once. The file to export was once again the
simple row turbo machine from the Getting started guide. In all cases the program showed the successfully exported message and all the STEP, IGES and STL files were checked and approved using the SolidEdge import capabilities to compare the geometries. TurboGrid file also was also created successfully but unfortunately the testing tools available are not prepared for this kind of file format. So 5 out of 6 option work correctly without introducing any bugs to the geometry.

For 3D profile points the program offers the option to export and import them. Using the same example from the previous tests we first tested the export option. Choosing a value of 2 for both Degree and Degree U and Sharp as teFlag option we tried to do the export. The result is that nothing happened when the button was clicked as if the function was not properly linked to the button or the button was deactivated. The test was repeated with the same degree value but Round option for teFlag but result was the same. The test was repeated 5 times with degree values ranging from 2 to 10 with both teFlag options but no change in response was noticed. The export function couldn't be tested because the button to initiate it is not operational.

Next phase was the testing of the import function. For this test we tried to import in the test file a blade row from a given example by the developers. The import feature works over XML file format but the configuration of the file is different from the ones that BladeDesigner creates when storing the turbo machines elements. A proper importing file consists of three row of data elements: profile, hub and shroud coordinates. Each coordinate is defined in the x, y, z space in two sets of data between brackets. The first brackets contain the view in the radial direction (section) and the second brackets include the data points starting from the leading edge to the trailing edge and going back defining both suction and pressure side of the airfoil. The file has to be stored inside the geometry folder of the chosen project and with the proper name for the program to read the path correctly. If these are not correct the import progress bar gets stuck at 0% freezing the program. The shell gave us the error seen on Figure 3.3:
Figure 3.3 – Shell error when importing incorrectly named files

Once the file was stored correctly in the folder with the proper name the test was run. In order to run properly the blade row in which is imported must have the same number of active profiles as the file imported otherwise the progress bar is stuck at 50% when trying to interpolate the camber surface and the shell shows the message error from Figure 3.4:
Figure 3.4 – Shell error when importing a incorrect type of file

Also if the value of Degree or Degree U are not enough for the file being imported the program will stuck at 25% when interpolating the Nurbs surface and the shell will show the following error shown on Figure 3.5:
If instead the values are too much or the teFlag option is incorrect the same error message as in Fig. 3.4 will be displayed.

All things considered if the correct options are chosen the import is done correctly and in the Plot tab of the active blade row we can see the imported blade displayed. The fact that the error messages aren’t displayed in the progress bar message box indicates that in a standalone version of the program the user would have no shell to look for the error and he would just see that the program is stuck. This lack of information in the interface should be solved to help the user through the process.

The import feature doesn’t write any parameter data just show the 3D plot of the imported element. So in this case if we need to build the turbo machine with the imported file we have to manually enter the profiles and blade row data to continue the process. This is not a problem or error itself but it leaves without meaning the import functions as the user will have to get the numeric data and input it anyway.

Object 2 – SCMMesher

To test the SCMMesher we had to input some values to condition the resulting mesh. First test run though was to check the stability of the interface when the user neglectfully hits the calculate button without entering any data. The result of it was that the program crashed and showed a blank interface while ceased responding to any
command. The result was repeated whenever the CGNS options were active but the CpFt coefficients were not used. The shell showed that the function was caught in a loop probably expecting data which caused the program to crash. It was necessary to kill the process and initiate again the BladeDesigner to get back control of the software. This is shown in Figure 3.6:

Figure 3.6 – Shell error when CpFt values are not entered

To test the functionality we had to use the data inputs from the scm_tutorial from the BladeDesigner documentation tutorial folder. The objective was to use the turboexample [4] turbo machine combined with the SCMMesher data examples to create meshed file to be used in CFD simulators. But when the button was clicked nothing happened. The button acted as if deactivated and no error message was displayed on screen. We knew that the function is linked to the buttons because earlier it caused a reaction in the program. So we checked the shell and Figure 3.7 was the result:
Figure 3.7 – Shell error when trying to execute the SCMMesher function

We repeated the test with different turbo machines and changing the data on the SCMMesher and the result was the same. This indicates that there's a construction bug inside the mesher widget.

Conclusions

For this test runs we have to conclude that only half of the functionalities expected were operational. All aspects related to geometry exportation checked out fine in the testing and have been correctly integrated in the program. This options provide a full operational development of the program via geometry creation for CFD simulation use and show that the program can be used for it's final purpose.

On the other hand the 3D profile points exportation is deactivated or the button is unlinked to the function. Either way it's not a functional option of the program and the button should be removed until it's fully operational to avoid confusion for the users.

At the same time the import function also checked out fine and performed it's function. However the function actually only import the 3D geometry into the Plot tab of the blade row without importing the data parameters needed to continue the turbo machine's building. If the user is forced either way to enter manually these parameters the purpose of the function is lost as it's only use is to show. This may not be a bug itself but a bad
implementation of the function which developers didn't take into account and that should be corrected if full integration into the creation process must be achieved.

As for the interface issues, it can be concluded that these functions tested in this report were included in the software without taking much care of communication between user and program as in any of the cases tested the program showed any kind of error message box or any explanation of the errors encountered. Particularly the import function create some message error in the shell which could have been easily integrated as a message box in the program to warn the user about a wrong specification of teFlag option or incorrect degree values.
3.3.3 - Conclusion on the 2011 version

As previously done for the 2010 version, once the testing process on the 2011 version is complete a final state report must be written. This version is the first to fulfill the whole process of CFD geometry oriented construction initially intended for the BladeDesigner software. The main difference between this and the previous version is that the functions to export geometry and create meshes have been implemented. Also the task of this report is to check the progress on correction the bugs found in other version and report the actual overall state of the program.

In order of criticality first issue that's necessary to report is that this new version hasn't fixed the errors of the previous version. Drawing new profiles is still impossible and this brings the same consequences that in the previous version. The imposition to work with the preset examples still made it impossible to test all the calculating methods for profiles and to create new original turbo machines Added to this the bug introduced by the copy/cut/paste functions hasn't been corrected either. The user is still unable to use these functions without losing the saved project in which they were used and the geometry construction functions are deactivated whenever these functions are used. These were major critical bugs already detected in the previous version although it was expected for them no to be corrected as the beta testing process began when the 2011 version was already released.

From the previous version we also have to report the good news. The bug encountered when using close to margin values in profile data which caused the inability to create the turbo machine's geometry and then introducing errors to the blade row itself has been corrected. The program responds properly to all the values inside the margin of operation and when working with out of margin values it issues the proper warnings and error messages without crashing.

As for the new functions added in this version we have to admit only a 50% success. The geometry export functions are fully functional in both turbo machine and blade row for all the options available to export. The 3D model exported have been checked with an alternate CAD software proving that the geometry isn't corrupted in the process.

The export function for the 3D profile isn't ready and couldn't be tested. As for this moment it is uncertain if the function it's linked to the button or not implemented in the code. As for the import function the test result show it worked correctly and the 3D geometry could be displayed in the program. Related to this functions we have to point out that the fact that the import only involves the geometry elements and doesn't import the data related to them renders this function useless. If the user has to input manually the data related to the imported file the act of importing isn't necessary at all.

The last addition to this version was the SCMMesher function. Although this function proved to be implemented and linked to the trigger button it was impossible to test properly. The error displayed by the shell during the test run indicate that the construction of the widget containing this function is flawed and it's impossible to achieve a proper functionality in any turbo machine or set of values given. The ultimate result is that the function is not available to use.
In the interface part of the testing we have to report that a regression has been made. When 2010 version had all of it's buttons operating properly the changes in this version to define a new display have introduced bugs in the performance of some buttons. Like the up and down buttons which experience bugs in it's operation or the copy/cut/paste which although introducing a major bug in the previous version they worked properly in it's function. In this new version aside from introducing the same bug they also corrupt the copied elements so the copies are not even created properly.

On the other hand in this new version the error messages have been improved and now they give more information about the error source allowing the user the track easily the error and correct it faster than in the previous version. While in the previous version the error messages only warned about an undescribed error in this version the error message contains a description of the nature of the error and the data point which causes it. While this is true for the already present in the last version, the new added functions don't have any kind of error messaging implemented. The bugs encountered have been discovered through the use of the shell with program standing in standby without warning the user. Although this not critical for the operation of the program in the cases involving progress bars the user only sees the program stuck in a process which leave him waiting for several minutes before realizing that the operation has been stuck. This kind of behavior is not proper for a software intended to be user friendly.

In conclusion the overall state of the program is still not ready for public distribution. The fact that the critical bugs haven't been solved plus the addition of new bugs relating the new functions and the fact that some new buggy behavior has been introduced in the interface lead to the conclusion that this new version represents a little step back and not forward for the software development. Although this version has capability to create and export geometry which can be used for it's final purpose, the objects created are not original and can only be created with a limited use of the program methods. These limitations can render the tool useless for users with specific construction needs. The final conclusion is that this version still needs improvements to be considered for public release.
Chapter 4:

Getting started manual

This manual is intended to explain to new users all the functionalities provided by the BladeDesigner tool and help them understand the creation process of the geometry. Through this tutorial we'll learn how to use properly the program and we'll put in practice these information by creating a simple turbo machine.

4.1 - The interface

The BladeDesigner program is implemented in Python scripts but has a visual interface designed in Qt. This GUI makes the work easier for those who aren't familiar with working with command lines and a shell. It works like most of the Linux programs and has the buttons that we're all familiar with. Full report on the GUI can be found on [3] of the Bibliography.

First we'll take some time to familiarize with the interface. Once started the program welcomes you to the main turbo machine page (Figure 4.1). On the top bar we have the menu. The menu contains all the options available to change the program settings and the file operations. Below the menu bar there's the tool bar with all the buttons needed to interact with the opened file such as add or remove elements or copy and paste them. On the left hand side of the GUI we have the object hierarchy tree. In this window we find all the elements contained inside one file ordered from top to bottom elements and in the shape of deployable elements. Click in the turbo machine to deploy all the blade rows inside it and then click in each blade row to deploy all the profiles that are inside.

The main window contains the data for the elements of the file. Click on the hierarchy tree on any object and in the main window the properties of this object will appear to be changed. On the bottom left side of the main window we find the tabs to change between the different data types there are in each file. Parameters show us the raw data to conform the element, Plot shows us the 2D or 3D graphics for the element we defined. Special opens the window for exports and imports from other files.
Let's take a closer look at the menu. On the File tab we have the options related to the file containing the turbo machine. The classical options are available here:

- New: to open a blank turbo machine
- Open: to open a previously saved project
- Save: to save the current project under its actual name in its actual folder
- Save as: to name and save the file on the specified folder
- Close: to exit the program

The Edit tab has the functions needed to edit the properties of the elements inside the turbo machine. The options available are:

- Copy: makes a copy of the selected element (profile or blade row) to paste it further
- Cut: stores a copy and deletes the selected item
- Paste: inserts the previously copied or cut element in the position of the currently selected item, moving this one down.
- Up: moves the selected item one position up inside the hierarchy tree
- Down: moves the selected item one position down inside the hierarchy tree

The View tab has the options to change the visualization of the program. It works by checking or unchecking the following elements:

- Object hierarchy: hides or shows the hierarchy tree of the turbo machine
- Tool bar: hides or shows the tool bar
- Python shell: when clicked it opens the python shell to allow the user to see the
development of the program in python command lines

The settings tab contains the language options of the program. By clicking on Languages we get a deployed list with the possible languages available. So far only German and English are available and to change from one to another we just have to click to check the desired language.

The Help tab contains the tools the program provides the user to help through the creation process:
- About BladeDesigner: when clicking this options a new window is opened with 4 tabs
- About: contains the definition of the program, proprietors and Copyright and legal advice
- Authors: list of people involved in the creation of the program, roles and contact data
- License: copy of the GNU license as in all free software
- Thanks to: reminders to people who supported the creation of the program or helped.
- Manual: opens a new window with the contents of this manual

Next element is the tool bar The tool bar contains ready to use buttons to edit the file just the same as the options on the Edit tab of the menu. There are three unique buttons:
- Add profile: adds a blank profile on the currently selected blade row.
- Add blade row: adds a blank blade row with one profile in the turbo machine
- Delete: deletes the currently selected item (profile or blade row)

The other buttons present in this tool bar are the same and perform the same actions as in the Edit tab of the menu. The buttons are Copy, Cut, Paste, Up and Down.

Inside the object hierarchy tree we can deploy or undeploy the view by clicking on the plus or minus symbol respectively. This way we can see the dependencies of each element in the bigger scheme of the turbo machine The only actions we can perform inside this window are Add profile, Add blade row or Remove by right clicking on the desired element we want to perform the action to.

The main window changes depending on which element we're working on and each has it's own different characteristics we'll explain now.
4.1.1 - The turbo machine's main window

The turbo machine's main window is divided in three tabs: Parameters, Plot and Special. The Parameters tab (Figure 4.2) contains the options to change the properties of the turbo machine. It is a very simple page consisting of one line edit, one deployable menu, and a button. In the line edit, we can see and change the directory where the file is stored. To change it, just write the new directory in the line edit or push the “…” button and select the desired folder from the window opened. The deployable list contains the debugging options available. The options are: No debug, Some debug or Full debug. The button “Calculate” is used to order the program to build the turbo machine's geometry using the parameters entered in the blade rows and profiles.

Once the geometry is built, we can click in the Plot tab to see it (Figure 4.3). When clicked, the window switches to the plot view with the geometry. To rotate the plot, click and hold the left button of the mouse and move it to focus on the parts you desire. The X, Y and Z axis in the bottom right side of the window will help you locate the geometry in space. To zoom in or out, you can either click and hold with the right button of the mouse and move right to zoom in or left to zoom out or if your mouse has a scroll wheel, scroll up to zoom in and down to zoom out. By clicking and holding the middle button of your mouse, you can move the geometry inside the window without rotating it. Very useful when the turbo machine has more than one blade row and we want to focus...
each one separately.

**Figure 4.3** – The turbo machines *Plot* window

The *Special* tab (**Figure 4.4**) contains the options to export the geometry built into a file format compatible with CFD simulators. The page consists of two check boxes and one button:

- Turbo machine to STEP: create a STEP file with the turbo machine geometry
- Turbo machine to IGES: create a IGES file with the turbo machine geometry

The check boxes allow us to choose with which file format we will export (we can check both at the same time and get the two exported files in each format) and the “Export” button starts the process.
4.1.2 - The blade row main window

The blade row main presentation is very much alike the turbo machine but it has an additional Plot tab. The Parameters tab presentation (Figure 4.5) is more complex than the turbo machine because it needs much more data to perform the build operations. It's composition differs in shape according to the blade definition method used which can be changed from the deployable list on the top right side of the window:

- Stacking line, stacking point & chord length: define the blade using the hub, shroud & stacking line in z,r coordinates.
- Stacking line, stacking point & chord length 2: define the blade using the hub, shroud, trailing edge and leading edge in z, r coordinates.
- Cascade of blades: define the blade describing the distance between blades, the length domain, stacking line distance & blade span.

Below that we fins the deployable list to choose the debugging options, which are the same as before (none, some or full). In the profile distribution list we have the following options:

- Isosurface: the distance between all profiles along the stacking line is the same.
- User defined: when chosen a chart with the active profiles appear for the user to specify the relative distance (0-1) to the stacking point of each profile.

Figure 4.4 – Turbo machine's Special window
The angle preservation flag list allows the user to choose which mapping preservation method to use:

- No angle preserving transformation: which is the same as length preservation method explained in chapter 2 and detailed in [2].
- 2D thickness distribution on 3D camber line: which means not using any of the above mentioned methods.

On the top left side there are two line edits which are the same for every method used to define the number of blades in the row and the angular speed defined for the blade row.

![Figure 4.5 – The blade row's Parameters window](image)

The Plot tab (Figure 4.6) works the same as in the turbo machine. Once built it contains the 3D presentation of one blade defined from the profiles inside the blade row.
The new tab parameter plot (**Figure 4.7**) is a 2D presentation of the profile data that conforms the blades. On the bottom left side of the plot we have the buttons to interact with it. From left to right:

- Reset view: returns the plot view to the original state
- Move left: switch to the previous profile data
- Move right: switch to the next profile data
- Pan axis: allows to freely move the view
- Zoom to rectangle: zoom in the selected area of the plot
- Configure subplots: opens a window to select the best view of the plot inside the window
- Save: allow the user to save the current plot as an image on the destination folder entered

On the bottom right side of the plot we have the buttons to change the presentation. From left to right:

- Type of line: deployable list to switch the line presentations, default is a line but can be switched to dotted line, crosses, intermittent line and more
- Color: deployable list with the available colors for the plot line.
- Data type: deployable list with all the data available to present. Default is angle
β1 but we also have angle β2, chord length, relative maximum thickness, approximate Nurbs nose radius, polynomial degree, stagger angle, circumferential offset and stacking line.

**Figure 4.7** – The blade row's *Parameters plot* window

The *Special* tab (**Figure 4.8**) contains the options to export the plot for use in other programs. The geometry exported can be either the blade or the hub/shroud geometry. In this case we also have the check boxes to choose in which formats do we want to export:

- Blade to STEP/STL/IGES/TurboGrid: export the blade into any of these format files.
- Hub/shroud to STEP/STL: export the hub/shroud to the these format files

Below the export options we have the options to import or export 3D profile points of blades. By clicking the Import button a browser is open to search and select the desired plot to import. The line edits allow the user to input the degree in both directions and the deployable list contains:
- Sharp: to export/import considering sharp edges
- Round: to export/import rounding the edges

Figure 4.8 – The blade row's Special window

4.1.3 - The profile main window

The profile presentation is also variable depending on the data we're going to enter. The parameter's page looks like in Figure 4.9. On the top of the page under the label Options there two deployable lists which allow the user to choose the Camber line distribution method and the Thickness distribution method. Camber line options are in between:

- Circular arc: describe the camber line with chord length and the thickness data.
- Power function: use the degree of polynomial, chord length, stagger angle and relative maximum thickness to describe.
- NACA 65: use the relative maximum thickness, Ca0, chord length and stagger angle.
- Nurbs: directly describe the position by entering the coordinates as well as the relative maximum thickness, error allowed, chord length and stagger angle.
Thickness distributions options are:

- Power function: enter the polynomial coefficients for the method. This option allows you to choose between an approximated approach defining the limit, radius of curvature and coefficient b2 with the p coefficients or a direct approach specifying both p and b coefficients.
- Nurbs: define the coordinates directly
- Double circular arc: (only available if circular arc is the camber line method) no need to specify any further data aside from the leading and trailing edge angles and radius and the resolution parameters

Depending on the combination of methods chosen the page will display the corresponding line edits and charts to enter the proper data into the program.

![Figure 4.9 – Profile's parameters page](image)

The Plot tab of the profiles (Figure 4.10) is a 2D presentation of the airfoil designed following the the specifications of the data entered in the parameter plot. The buttons on the bottom side of the plot are the same as in the parameter plot page of the blade row window but instead of having a deployable list or choose between parameters data here we have a deployable list to change the presentation between:
- Airfoil: presentation of the whole airfoil with the camber line inside
- Camber line: only presents the camber line
- Thickness: presentations of the upper side of the airfoil only showing the thickness distribution along the airfoil
- Combo: combination of the three views stacked one on top of the other.

Figure 4.10 – The profile's Plot window

4.1.4 - The SCMMesher window

The SCMMesher it's a tool designed to allow the user to create meshed geometry in order to use in CFD simulators that lack that capability. The SCMMesher can be accessed by clicking on it in the hierarchy tree as the last element of the deployed turbo machine. Once clicked the mesher's main page is displayed on screen as in Figure 4.11.
The main page is divided in six data areas. On the top left corner the **Options area** where we define the general options for the meshed turbo machine we want to create.

- **Omega**: radial velocity of the spinning blade rows
- **Mass flow**: mass of air flowing through the turbo machine per second
- **Max Entropy**: maximum entropy in the outlet
- **Min Entropy**: minimum entropy in the inlet
- **Debug**: allows the user to choose which debug option he prefers: None, Some or Full
- **AxialGradinBlock 1/2/3**: describe the grading parameters for every row.

The **BlockCellsAx area** is where the user defines the number of cells in each block as well as the in the radial direction.

- **Block 1**: cells in the first block
- **Block 2**: cells in the second block
- **Block 3**: cells in the third block
- **BlockCellsRad**: number of cells in the radial direction

The **Radial parameters area** contains the air data information. On the chart below and depending on the number of cells defined in the blocks the user has to input the data of row.
– total Pressure: total pressure in the defined row
– total Temperature: total temperature in the defined row
– C_U: circumferential velocity of the defined row

The Axial parameters area only contains the chart with Static pressure for every row defined in the Blocks area. The user has to input in each row the correspondent pressure value.

The CGNS Output area is an optional area which adds more options to better define the mesh. The options available are:

– Write Exponents: option to write or not the CGNS exponent values
– One Block per Row: holds together the 3 cell blocks
– Merge Blocks: holds adjacent ring space blocks together
– CU: adds a swirl at the inlet
– Static Pressure: adds static pressure hubs to the mesh
– CpFt: allows the user to enter manually the exponents

The Additional Nurbs Parameter area it's also an optional area which allows the user to further describe the parameters of the mesh.

– Sep: defines the separation between blocks
– Tolerance: define the tolerance when creating the mesh cells
– Max Iteration: maximum number of iterations

4.2 – My first turbo machine

Now that we know how the program works we're going to put this knowledge to a test by creating the simplest turbo machine. This project will consist on one blade row and five profiles (minimum required to create a good blade). Once finished this tutorial you'll be able to create your own turbo machines and make them more complex.

First step is to open a blank project. Go to the File tab of the menu and click New. The program automatically opens a blank turbo machine with one blade row and one profile. To ensure we don't lose any work the next step will be to name and save our project. The program automatically names the project after the folder where it is stored so in order to keep the documents in order we're going to create a new folder named “My first turbo machine” wherever we want to save the project and then specify the route. You can either type or browse the directory where you want to save the file on the corresponding line edit of the turbo machine's main page or go to File/Save as and then look for it in the browser. Now that our project has a storage directory we can save the progress at any time by clicking on File/Save.

Let's start creating. As this is only a tutorial we're not going to use the debug function so in the turbo machines main page select the option of No debug on the deployable list below the directory file. Then click on the blade row in the hierarchy tree. The blade
row main page should appear. Under the label General we'll input the number of blades and the angular speed desired for our turbo machine. Use 31 blades and an angular speed of 3000 rad/s. Next to it under the Flags label we have to choose the parameter options we're going to input. From the deployable lists select the following:

- Blade definition: stacking axis, stacking point, chord length
- Debug: No debug
- Profile distribution: User defined
- Angle flag: no angle preserving transformation

Now we'll have to enter the raw data that will define our blade row for the plot. Under the General options we have a chart labeled Hub. In the line edits next to the chart we're going to enter all these pairs of numbers:

<table>
<thead>
<tr>
<th>Z</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>240</td>
</tr>
<tr>
<td>20</td>
<td>240</td>
</tr>
<tr>
<td>30</td>
<td>240</td>
</tr>
<tr>
<td>40</td>
<td>240</td>
</tr>
<tr>
<td>60</td>
<td>240</td>
</tr>
<tr>
<td>100</td>
<td>240</td>
</tr>
</tbody>
</table>

Table 4.1 – Data example for the Hub

To enter them just write each pair just type the number into the corresponding line edit and click the Add button. If you made a mistake you can delete the pair of numbers selected by clicking the Delete button. In case you got them in the wrong order with the Up and Down buttons you can move the selected pair of numbers to the correct position in the chart. Notice the lock check boxes next to the line edits. When the box is checked the value inside will remain after we click the button add. In this case as all the "r" column has the same button we just have to write it once and check the lock feature.

Next to the Hub data we find the Shroud data. It has the same shape as the Hub data with the same buttons so the entering method is the same. The data corresponding to these chart is the following:
Now below this chart there's another one labeled Stacking line. We'll use the same method (remember to lock the corresponding label when the value is the same in the whole column) as previously to enter the following data:

<table>
<thead>
<tr>
<th>Z</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>240</td>
</tr>
<tr>
<td>35</td>
<td>250</td>
</tr>
<tr>
<td>35</td>
<td>260</td>
</tr>
<tr>
<td>35</td>
<td>280</td>
</tr>
<tr>
<td>35</td>
<td>300</td>
</tr>
<tr>
<td>35</td>
<td>380</td>
</tr>
</tbody>
</table>

**Table 4.3 – Data example for the Stacking line**

Now the bottom charts labeled Span and Circumferential offset on the bottom of the page have only one line because there's only one profile active in the blade row so we'll get back at them later.

We said that we needed at least 5 profiles to get a good geometry but the profiles can be identical for a simple turbo machine so we're going to create one and copy it 4 times to get five identical profiles.

Let's start with the active profile. On the hierarchy tree click on the active profile to bring up the Profile main page. In the *Parameters* tab under the *Options* label select Power function as method for both the Camber line and Thickness distribution. Below that find the *Weight* label and select from the deployable list named *Weight combinations* the option *Weight functions* and from the deployable list named *Weight condition* select straight line. On the right of the weight options and under the label *Polynomial* select 3 for the degree of polynomial camber line. For a good profile we
need a resolution of at least 200 points, input this value on the corresponding line edit.

With the corresponding function methods set now we'll enter the data that will define the profile. Under the label Angles we'll enter the data for the angles \( \beta_1 \) and \( \beta_2 \). On the corresponding line edits type 137.5 for \( \beta_1 \) and 102.5 for \( \beta_2 \). Under the label Additional select power function from the deployable list named Thickness method and enter 75mm for the Chord length and 27.5 for the Stagger angle.

At the bottom of the page and under the label Polynomial coefficients select from the deployable list Direct as method and enter on the corresponding line edits the following data:

<table>
<thead>
<tr>
<th>p0</th>
<th>2.37085805</th>
<th>a0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0.71143029</td>
<td>a1</td>
<td>0.08480047</td>
</tr>
<tr>
<td>p2</td>
<td>0.71322412</td>
<td>a2</td>
<td>0.1405026</td>
</tr>
<tr>
<td>p3</td>
<td>1.5</td>
<td>a3</td>
<td>-0.56161045</td>
</tr>
<tr>
<td>a4</td>
<td>0.33630738</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.4** – Example polynomial coefficients

Now if we click on the *Plot* tab we should be able to see the 2D presentation of the profile we've just created. If everything is correct then we can copy and paste it 4 times. Select the profile on the hierarchy tree by left clicking once over it and then click the Copy button on the tool bar. Observe that the Paste button has changed and now has a little profile shape on the bottom left side of the buttons. This means that we have a profile copied in storage. Now just click four times on the paste button and you'll see four profiles identical to the original inside the blade row.

To finish the work we have to create the 3D geometries. So go back to the blade row window by clicking it on the hierarchy tree. Now we can enter the missing data on Span and Circumferential offset. As we see now each chart has 5 lines corresponding to the 5 profiles active inside the blade row. We are not going to use circumferential offset so we can leave this chart with zeros. In the Span column enter these numbers in this order:

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>0.8</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 4.5** – Example for the span
Once done click the Calculate button on the bottom left side of the page. Once the loading bar reaches 100% click on the Plot tab to see the 3D presentation of one blade. Check the parameters plot if you wish to ensure that the presentation corresponds to the data entered.

Now with the blade built we're going to build the entire turbo machine Go back to the turbo machine's main page and click on the Calculate button. Wait until it reaches a 100% and then go to the Plot tab. The resultant geometry is your first turbo machine congratulations! Now would be a good moment to save the results if you haven't done so. And remember this is a simple turbo machine, actual ones consist in more than just one blade row. Feel free to explore the limits of the programs with many blade rows.

4.3 – Export your work

Now that we have a built geometry it's purpose is to serve in CFD simulations. There are many CFD simulators in the market some allow to build the geometry inside and others need a previously done geometry to work with. That's why BladeDesigner has an export feature built in.

We're going to export our brand new turbo machine to use in CFD simulators. If you closed the project after completing the previous chapter go to File/Open and retrieve “My first turbo machine”. Once in screen rebuild the blades and the turbo machines if the program doesn't do that automatically. With the geometry once again built select the blade row from the hierarchy tree and go the Special tab. Now we have to select what we want to export and in which format. For this tutorial we're going to export into STEP format as it is one of the most common format files and accepted in many CFD simulator programs. Check the boxes named “Blade to STEP” and Hub/Shroud to STEP”. Remember that you can check as many boxes as you want at the same time and if you need more than one file format you can check them all. Once selected the options you need just click on Export. Once the progress bar reaches 100% a message will inform you that the export was successful in each option that you checked and provide you with the directory where the files are stored. By default this directory is inside the geometry folder of the actual project.

Now we have the blade row but we also want to test on the entire turbo machine So to export the turbo machine go to the Special tab of it and repeat the process. Check the box named “Turbo machine to STEP” and click Export. Once the process is completed you'll find the STEP file inside the geometry folder of the project. Now these STEP files can be used to import into a CFD simulator and we can test how good is the turbo machine we've just created.

4.4 – SCMMesher

As we said some CFD simulators don't have any meshing capabilities built in so in order to use our work in BladeDesigner in those programs we need to create our own
mesh for the turbo machine. Now that we still have our first turbo machine we're going to create a mesh for it. It will be an example mesh and the data is not related to a real case but it'll illustrate how to do it. Using this functions is as simple as entering all the proper data and clicking on the Calculate button. So now we're going to define some data examples for you to use as a test.

Options:

<table>
<thead>
<tr>
<th>Omega</th>
<th>1000</th>
<th>Debug</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass Flow</td>
<td>10</td>
<td>AxialGradingBlock1</td>
<td>1</td>
</tr>
<tr>
<td>Max Entropy</td>
<td>2</td>
<td>AxialGradingBlock2</td>
<td>1</td>
</tr>
<tr>
<td>Min Entropy</td>
<td>1</td>
<td>AxialGradingBlock3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.6 – Example values for the options parameters

BlockCellsAx:

<table>
<thead>
<tr>
<th>Block</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block1</td>
<td>3</td>
</tr>
<tr>
<td>Block2</td>
<td>6</td>
</tr>
<tr>
<td>Block3</td>
<td>3</td>
</tr>
<tr>
<td>BlockCellsRad</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 4.7 – Example values for the BlockCellsAx

Radial Parameters:

<table>
<thead>
<tr>
<th></th>
<th>total Pressure</th>
<th>total Temperature</th>
<th>C_u</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10000</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10001</td>
<td>301</td>
<td>1,1</td>
</tr>
<tr>
<td>3</td>
<td>10002</td>
<td>302</td>
<td>1,2</td>
</tr>
<tr>
<td>4</td>
<td>10003</td>
<td>303</td>
<td>1,3</td>
</tr>
<tr>
<td>5</td>
<td>10004</td>
<td>304</td>
<td>1,4</td>
</tr>
<tr>
<td>6</td>
<td>10005</td>
<td>305</td>
<td>1,5</td>
</tr>
<tr>
<td>7</td>
<td>10006</td>
<td>306</td>
<td>1,6</td>
</tr>
</tbody>
</table>

Table 4.8 – Example values for radial parameters
**CGNS Output:**

<table>
<thead>
<tr>
<th>Write Exponents</th>
<th>False</th>
<th>CU</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>One Bloc per Row</td>
<td>False</td>
<td>static Pressure</td>
<td>False</td>
</tr>
<tr>
<td>Merge Blocks</td>
<td>True</td>
<td>CpFt</td>
<td>True</td>
</tr>
</tbody>
</table>

**Table 4.9 – Example options for CGNS output choices**

**CpFt:**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a11</td>
<td>3.08792717</td>
<td>a12</td>
</tr>
<tr>
<td>a21</td>
<td>1.24597184E-03</td>
<td>a22</td>
</tr>
<tr>
<td>a31</td>
<td>-4.23718945E-07</td>
<td>a32</td>
</tr>
<tr>
<td>a41</td>
<td>6.74774789E-11</td>
<td>a42</td>
</tr>
<tr>
<td>a51</td>
<td>-3.97076972E-15</td>
<td>a52</td>
</tr>
</tbody>
</table>

**Table 4.10 – Example values for the CpFt coefficients**

**Additional Nurbs Parameters:**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep</td>
<td>20</td>
</tr>
<tr>
<td>Tolerance</td>
<td>1E-6</td>
</tr>
<tr>
<td>Max Iteration</td>
<td>20</td>
</tr>
</tbody>
</table>

**Table 4.11 – Example values for the additional Nurbs parameters**

**Axial Parameters:**

For the static pressure values start with 10000 and increase in one unit every row.

Once all the parameters have been entered just click on the Calculate button and wait for the result (we remind the user that in this actual version this function is not yet operative).
Chapter 5:

Final Conclusions

During the course of this project we've been able to run test series in two versions of the program one being the latest available and the previous to this one. This fact allowed us to experience the expansion of the software which is still a work in progress. With the work done it's time to evaluate the tool, describe it's actual readiness state and suggest guidelines for future expansions of the software.

As it's been widely stated throughout this document BladeDesigner is a CAD program designed by students and for students mainly. The primary role of the tool is to provide a geometry development environment to create turbo machine's 3D plot to be used in CFD simulators. Turbo machine's design is the first step in the creation of aeronautical engines and before the material construction of them the CFD simulations allow engineers to test in a virtual environment the behavior of the designs made to optimize them for maximum efficiency. That's why an accurate virtual geometry is needed to assure the validity of the simulation results. BladeDesigner ensures the proper construction of the geometry through numerical calculations which we've seen on chapter 2. In addition to the geometry construction functionalities BladeDesigner is also prepared to play it's part in a bigger chain of events starting by the geometry constructions and following with CFD simulations. This continuity is assured by the implementation of exporting and meshing functions. Exporting functions provide BladeDesigner with the ability to translate it's own format files into standard CAD format files to be read by other programs which require the 3D geometries to operate like Meshing programs or CFD simulators. Meshing options provide the necessary functions to skip one step in the chain and provide directly out of BladeDesigner a ready to test CFD adapted mesh. With these options fully functional BladeDesigner 3D accomplishes its objective and therefore is evaluated as a successful tool which could become important for development engineers in the future.

Unfortunately BladeDesigner is not ready for public distribution yet. The full extent of its functionalities are not operational yet. Through this testing process major bugs that threaten the program's proper operation have been detected. As detailed in the testing report in chapter 3, only half of the intended functionalities work and the ones that work are threatened by secondary functions bugs. We've been able to determine that the building process of the geometry works but a flaw in the GUI's design prevents the user from creating new profiles forcing him to work over preset examples and altering the data. This design flaw, though it may be avoidable, eliminates the possibility of using all the calculation methods implemented in the program for profile's thickness distribution and camber line. These valuable resources were implemented in the program for a reason and the inability to use them limits the output possibilities of the program.

With the limited options of geometry available the next choice is to export them or mesh them. The export capabilities of the program have been certified to work in all of its options and to a full extent. This is the first of the two ways to realize a full CFD oriented geometry creation process and leads to the conclusion that the software
although flawed can perform his function with limitations. Although it's not enough for public release a skilled user could still achieve useful results from BladeDesigner. The other option to finalize a full creation process is to created a meshed file for the geometry created. Unfortunately this is a option not available for the time being as the function implemented in program is not operational and so unavailable for the user. This is another limitation for the full operation of the program which enforces the conclusion that the program is still not ready for public release.

In addition to all these functionality limitations some secondary functions implemented in the program and oriented to aid in the construction process introduce bugs to the correct operation of the few operational functions. These functions don't affect directly the creation process as their use is optional but their reason to be is to speed up the process and help the user to ease the process. Their behavior prevents the user to use them in order to achieve a full geometry construction. This limitation caused the program to be time-inefficient and not a proper tool for business like applications which depend on accuracy on the result and time optimization.

The interface created for the program has a good and intuitive design that helps the new users get around and familiarize quickly with the use of the program. However in the last version tested some buggy operation of some buttons was detected. The repetitive and not random nature of these bugs suggest flaws in the GUI's implementation and enforce the overall conclusion that the software needs major corrections in order to be ready for a public release. The idea of translating the program's layout to different languages (German, English, Spanish & Catalan) and the use of free software licenses are evidences of the developer's will to create a tool available worldwide for all publics and countries.

As a work in progress, BladeDesigner is always in need of expanding and adding new features to complement it's work and making of it a good engineering resource. Although much work is needed to bring the actual program to a fully functional state, new functionalities also have to be taken into account to the task of perfecting the program.

The program's objectives have been fully reached by the use of the actual functionalities implemented and there's no need for further progressing in that direction. The GUI however it's a rather new item added in BladeDesigner which originally ran through python scripts. A good interface can make the difference when choosing a software for a company or university and so improvements must be made in order to perfect the overall program.

For starters in the actual version the program is able to display only one turbo machine at a time when most CAD applications nowadays are capable of working with more than one active file. With this feature built in BladeDesigner a user would be capable of working interactively with more than one file at a time to compare or exchange data between them. A multi-environment capable of handling several turbo machines at the same time could be a great improvement of the interface.

In addition to that the new functions could have improved visualization. For example
the meshing tool could add a visualization tab to show the built mesh in order for the user to inspect it and decide whether to change it before saving and sending it to CFD simulation. A *Plot* tab like the ones already built for the turbo machine, the blade row or the profile would be sufficient and add a great sense of perspective to the user when meshing.

To deal with the bug solving issue the usual procedure is to debug the source code and monitor the variables as they go through the functions looking for the command line which introduces the bug. This procedure can take a lot of time in programs like BladeDesigner with great amounts of code. Python integrates a feature to run unit tests in its codes. Unit test is a testing procedure which could help the developers cut in half the time needed to pinpoint the source of the errors in the program. For more information about unit tests got to Annex 1.

As part of the free software environment everyone is able to download and modify BladeDesigner and use it for his own purposes. For anyone who wishes to contribute to the expansion of this project, the official developer’s team will take any suggestions and corrections offered on the official project’s website:

http://sourceforge.net/projects/bladedesigner/
ANNEXOS

TÍTOL DEL TFC/PFC: BladeDesigner Beta Testing Process & Getting Started Guide

TITULACIÓ: Enginyeria Tècnica Aeronàutica, especialitat Aeronavegació

AUTOR: Antillach París, Jordi

DIRECTOR: Lang, Sebastian

DATA: 15 de juliol de 2011
6.1 - ANNEX 1: UNITTEST

Unit testing a programming method to test individual units of code program. Units are the smallest components of an active software, in the case of object-oriented programming an example of unit can be a class or a module. It is important to notice that unittest won't find all the errors in the code as it's tests are limited to assess the proper functionality of each unit individually and problems arisen by interaction between units are not detected by unittest. But testing that all functions are correct by themselves is half the way in cleaning the bugs in the code. Full description of unittest can be found on [5] of the Bibliography.

Python programming language has in it's libraries a functions called unittest that is the one that must be implemented in the source code for testing. unittest supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework. The unittest module provides classes that make it easy to support these qualities for a set of tests. To achieve this, unittest supports some important concepts:

test fixture
A test fixture represents the preparation needed to perform one or more tests, and any associate cleanup actions. This may involve, for example, creating temporary or proxy databases, directories, or starting a server process.

test case
A test case is the smallest unit of testing. It checks for a specific response to a particular set of inputs. unittest provides a base class, TestCase, which may be used to create new test cases.

test suite
A test suite is a collection of test cases, test suites, or both. It is used to aggregate tests that should be executed together.

test runner
A test runner is a component which orchestrates the execution of tests and provides the outcome to the user. The runner may use a graphical interface, a textual interface, or return a special value to indicate the results of executing the tests.

The test case and test fixture concepts are supported through the TestCase and FunctionTestCase classes; the former should be used when creating new tests, and the latter can be used when integrating existing test code with a unittest-driven framework. When building test fixtures using TestCase, the setUp() and tearDown() methods can be overridden to provide initialization and cleanup for the fixture. With FunctionTestCase, existing functions can be passed to the constructor for these purposes. When the test is run, the fixture initialization is run first; if it succeeds, the cleanup method is run after the test has been executed, regardless of the outcome of the test. Each instance of the TestCase will only be used to run a single test method, so a new fixture is created for each test.

Test suites are implemented by the TestSuite class. This class allows individual tests and test suites to be aggregated; when the suite is executed, all tests added directly to the
suite and in “child” test suites are run.

A test runner is an object that provides a single method, run(), which accepts a `TestCase` or `TestSuite` object as a parameter, and returns a result object. The class `TestResult` is provided for use as the result object. `unittest` provides the `TextTestRunner` as an example test runner which reports test results on the standard error stream by default. Alternate runners can be implemented for other environments (such as graphical environments) without any need to derive from a specific class.
6.2 - Annex 2: Troubleshooting

This a list of the error encountered during the program operations, it's nature and workaround to avoid or solve them. Most of the functions in the program have security measures built in to avoid errors caused by missed data entries which trigger error messages that inform the user of the cause of the problem to properly solve it. This guide includes only the errors not detected by the program and detected via the command shell. There are more errors concerning the program which couldn't be detected with the shell and so they're not reported here.

Import errors:

Error #1

```
IOError: [Errno 2] No existe el fichero o el directorio: '/home/administrador/Dокументos/Examples BladeDesigner/Test 12/geometry/Raw_0_Points.xml'
```

<table>
<thead>
<tr>
<th>Error :</th>
<th>The file or directory doesn't exist: “..../Row_x_Points.xml”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source:</td>
<td>When specifying a folder to extract the file to import the file is either not present or has the incorrect name.</td>
</tr>
<tr>
<td>Workaround:</td>
<td>If the file is indeed present in the folder specified check the name and ensure that the row number “x” matches the position where you're importing it.</td>
</tr>
</tbody>
</table>

Error #2

```
IndexError: list index out of range
```

<table>
<thead>
<tr>
<th>Error :</th>
<th>List index out of range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source:</td>
<td>When importing a file which has a different layout from the one that the program requires this message is issued in the shell</td>
</tr>
<tr>
<td>Workaround:</td>
<td>If the shell sends this error the file that you're trying to import is not a valid file. Only option is to rewrite the file for BladeDesigner to be able to read it.</td>
</tr>
</tbody>
</table>

Error #3

```
IndexError: index out of bounds
```


Error : Index out of bounds
Source: When importing a file if the degrees assigned are excessive or the teFlag option is incorrect this message will appear on the shell
Workaround: Enter lower degree values. If values are correct try selecting another teFlag option.

Error #4

**Exception: Number of Points in u-Direction to few for NURBS-Interpolation**

Error : Incorrect value for Degrees
Source: When importing a file if the degrees assigned are too low this exception will appear on the shell.
Workaround: Enter the correct degree values.

Meshing errors:

Error #1

```
--Return--
> /home/administrador/BladeDesigner/bladedesigner/src/GUI/QScmMesherWidget.py(53)
)debug_trace()--None
--> set_trace()
(Pdb) 
```

Error : Program freezes and becomes unresponsive. Main window turns white and no objects are displayed.
Source: When clicked the button Calculate to initiate the mesh if the CGNS options are checked but the CpFt is not this error will appear.
Workaround: If available input the CpFt coefficients If not available disable the CGNS options by unchecking the box.

Error #2

**AttributeError: 'QScmMesherWidget' object has no attribute 'LEA21'**
| Error:          | Program looks unresponsive to the calculate command. No action is performed by the program |
| Source:        | The source of this error is a design flaw in the function |
| Workaround:    | There's no workaround for this error as the problem lies in the code behind the function and not in the user's data or option chosen. |
Bibliography


