Digital Wind Tunnel

CAD Conversion and Pre-Processing

M.Sc. Software Engineering for Technical Computing

Aman Makkar

S321839

Abstract

The purpose of this report is to provide a breakdown of the work carried out to design and write a software that can convert CAD files into other CAD formats, that can then be used for the purposes of being loaded onto the Digital Wind Tunnel. Research has been carried out to understand the industry standards of CAD files that can be used for meshing and can be used for computational fluid dynamics purposes. Furthermore, research was carried out on open-source CAD

software to understand their architecture, how they carry out CAD file conversions and pre-processing work, and if these can be used in conjunction with a Python script to be able to convert CAD files and even mesh them directly via the command line.

The report also discusses the need for Computational Fluid Dynamics within the industry, commercial CAD software and the reason to use open-source CAD software for the purposes of this project. Furthermore, a thorough testing has been carried out on the CAD conversion software to determine its performance, suitability, maintainability, and portability.

Keywords:

Open Source, CAD files, Commercial CAD software, Computational Fluid Dynamics, pre-processing, Mesh, Python and Testing.

Acknowledgements

The completion of this MSc has been a very steep learning curve for me; two years of putting my life on hold so I could change my career and fulfil my

passion. However none of it would have been possible without the help of so many people, and I would like to dedicate this thesis to them.

First, I would like to thank my Supervisor Dr Karl Jenkins whose support and constant guidance is what made this thesis possible. Always available and providing great guidance with his expertise is something that I will never forget.

I also want to thank my wife Sundeesh, who stood by my side and supported me even when I was struggling and very difficult to live with. She was my rock during the two years I spent completing my MSc and provided me with the motivation needed when it was running low and tolerated me spending all of my free time in front of a computer screen, rather than in her wonderful company.

I want to thank my parents who have supported me throughout my life, whether that be financial or emotional support, and without which life would not have been possible, let alone this thesis. My sister, for her unwavering support and for showing me the bigger picture to complete my MSc.

I also would like to thank Dr Irene Moulitsas, who accepted my application to do the MSc even though my knowledge and background lacked in what was required to be accepted. Furthermore, for being there to answer all of my questions whether technical or professional in nature.

Finally, I want to thank Andy Dargle. He gave me the opportunity to work as a Software Engineer and pushed me to complete this MSc. He removed all of the hurdles that stood in my way to follow my passion, guided me through the maze that is professional engineering, and ensured that I could work in a place where I can always learn. He also helped me find a new job within the same Company when the Aerospace industry went into decline due to the pandemic.

Contents

List of Figures

List of Abbreviations

CAD	Computer Aided Design
OCCT	Open Cascade Technology
GPU	Graphics Processing Unit
GPGPU	General Purpose Graphics Processing Unit
CFD	Computational Fluid Dynamics
GUI	Graphical User Interface
STL	Standard Triangle Language
STEP	Standard for the Exchange of Product Model Data
IGES	Initial Graphics Exchange Specification
DXF	Drawing Exchange Format
UAV	Unmanned Aerial Vehicle
BREP	Boundary Representation

CAM	Computer Aided Manufacturing
CORBA	Common Object Request Broker Architecture
API	Application Programming Interface

This page has been intentionally left blank

1. Introduction

The purpose of this thesis is to design, develop and test a program that can convert CAD file formats, from one build format to another, as well as from part file to a mesh file. This conversion needs to be carried out without a need to open a whole other software, as demonstrated by Hinatea Teriierooiterai in her report “Digital Wind Tunnel Pre-Processing Computer Aided Design for Mesh Generation”. Hinatea used the Salome open-source CAD software to convert the files, which is a very long process for the mere purpose of converting files. The reason for converting the files is because the CAD software used for the DWT project is CATIA, and the files must be converted into formats that CATIA can accept before they can be tested in the DWT. The purpose and the importance of the DWT project will not be discussed in this thesis; a very thorough breakdown has been provided in the work carried out by students who worked on the project prior to this thesis.

There are various CAD software that are available to the market in this digital age, including commercial CAD software such as Solidworks, Inventor, Creo, CATIA as well as open source CAD software such as FreeCad, Salomi and GCad3D. All of the commercial CAD software are relatively similar with very subtle differences to choose from, and which one is chosen is based on preference, cost, and ease of use. However, there are various smaller or start-up companies that struggle with the cost of the commercial CAD software and they can carry out their design work using the open-source software. The same principle applies to CFD software and there are various commercially available software, such as Ansys, SimScale as well as open source CFD software, such as OpenFoam and SU2.

CAD software is used across various industries such as dentistry, fashion, shoe design, furniture, motor sport, Aerospace and many more. In dentistry, CAD can be used for the design of inlays, onlays, implant abutments, fixed partial dentures and so on (Irfan et al., 2015). In fashion, it serves the important purpose to increase productivity, standardise design methods and shorten lead times (Bertolotti et al., 2004). In the shoe industry, the usage of a CAD program results in time saving, an improvement in the quality of the shoes, and there is no need to experiment with making the shoe of a new design when it can be designed and reviewed on a CAD software, thus saving money and time (Paris & Handley, 2004). In the furniture industry, the use of CAD software leads to the improved quality of work, as well as a shorter lead time from concept to production (*The Use and Implementation of CAD in the Swedish Furniture Industry - ProQuest*, n.d.).

Like other industries, CAD plays a vital role in the design and product development within the Aerospace industry. CAD systems within the Aerospace industry benefits the engineers in a way that they can draw a new design much quicker, be able to share the information across the board using for instance the STEP file format (Williamson, n.d.). The usage of CAD in Aerospace leads to various other advantages, including but not limited to supporting multiple engineering views, satisfying space constraints, maintaining spatial integration and so on (Bowcutt, 2003).

The CAD convertor program needs be used as a plugin for the Digital Wind tunnel project. Open-source software that already exists can be used in conjunction with the program to be able to convert the files, the intention is to use popular open-source software that is well maintained, and the documentation is readily available to be able understand how the program and all of its module's work. The work on this thesis is carried on from the work that was carried out by previous students on the DWT projects, in particular from the work carried out by Hinatea Teriierooiterai in her report "Digital Wind Tunnel Pre-Processing Computer Aided Design for Mesh Generation". Furthermore, a comparison of which CAD file formats are state of the art in the industry and are supported by most CAD software was carried out, to determine which file conversions were necessary, to be able to run the files through the CATIA software. To determine which of these open-source software would be useful for this thesis, a comparison was made between the CAD software. This included the ease of use, maintainability of both the CAD software as well as the program for the DWT, available documentation, and ease of implementation.

The project requirement is to write a program that can convert various CAD file formats into part and mesh formats that are accepted by the CATIA program. The breakdown of the objectives for this thesis is as below:

- Review the various open-source CAD software to determine how the file conversion takes place.
- Based on the above review, determine which software would be most suitable for the DWT, based on various factors such as the maintainability, ease of use and the information available.
- Determine which CAD file formats are state of the art within the industry.

- Write a program that can use modules from the open-source CAD software to convert the files from the command line.
- Explain how the Input and Output procedures of the conversion software work.
- Carry out tests on the conversion program to determine its limitations and what can be done in the future to be able to improve this program.
- Discuss the results of the test and determine if it meets the project requirements.

2. Literature Review

2.1. Digital Wind Tunnel

Digital wind tunnel is an ongoing project at Cranfield University. The aim of the project is to be able to provide a program that can carry out all the aspects required for an aerodynamic simulation process (Wildt, 2018). The simulation process has three major parts: pre-processing, processing, and post-processing (Jelen, 2016).

Pre-processing includes formulating the problem, modelling geometry and flow domain, establishing initial and boundary conditions, and generating the grid. Processing includes establishing the simulation strategy, establishing input parameters, and performing the simulation. Post-processing includes acquiring the results, making comparisons of the results and documentation (Jelen, 2016). The work carried out in this thesis will fall within the pre-processing part of the simulation, and a breakdown of the pre-processing process is provided below.

2.2. Computational Fluid Dynamics in Aerospace

CFD has played an instrumental part in the design and analysis across various industries, including the Aerospace, Automotive, and Naval industries. Some parts of CFD are in a mature state, however there are emerging states within CFD that could help further develop the Aerospace design, figure 1 below shows three states within CFD, green area is the mature CFD state, blue states have some use CFD and red areas show where CFD is expected to make a difference in the future (Venkatakrishnan, 2016).

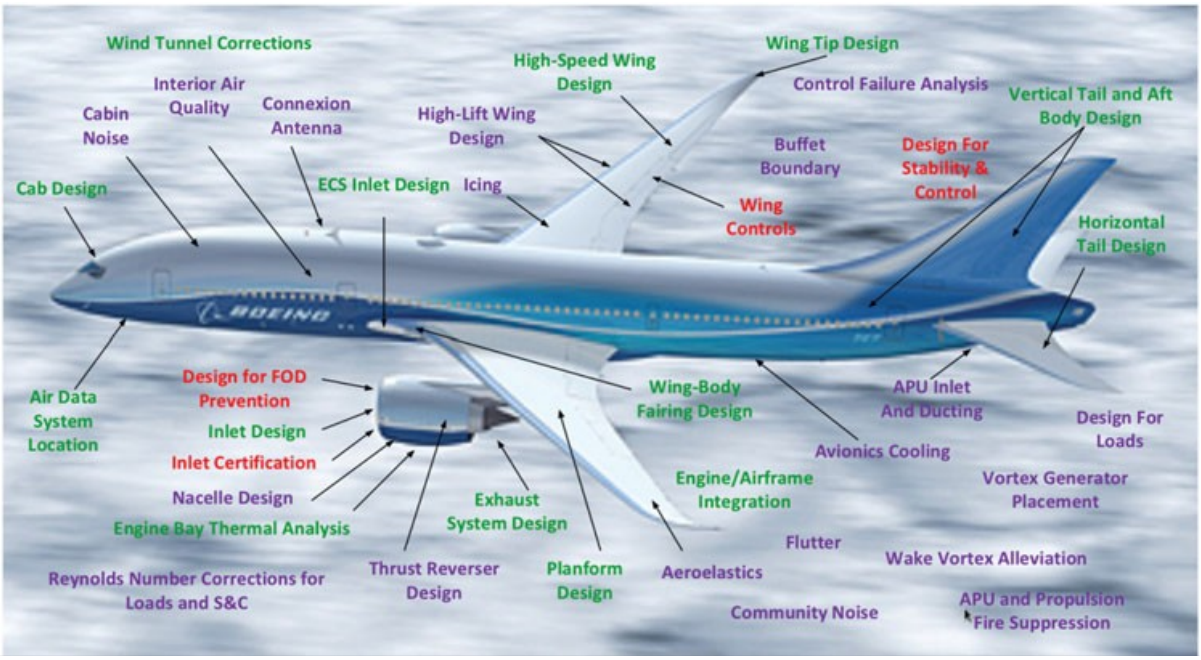


Figure 1: Boeing Aircraft with various CFD states

Using CFD is a multi-disciplinary approach within the Aerospace industry, and it is used for the design and analysis of flexibility of the wings, icing models, far-field noise propagation models and conjugate heat-transfer. The advances in CFD have reached a stage where the results from CFD are now compared directly to the data gathered during flights tests, rather than the data gathered during wind tunnel testing (Venkatakrishnan, 2016).

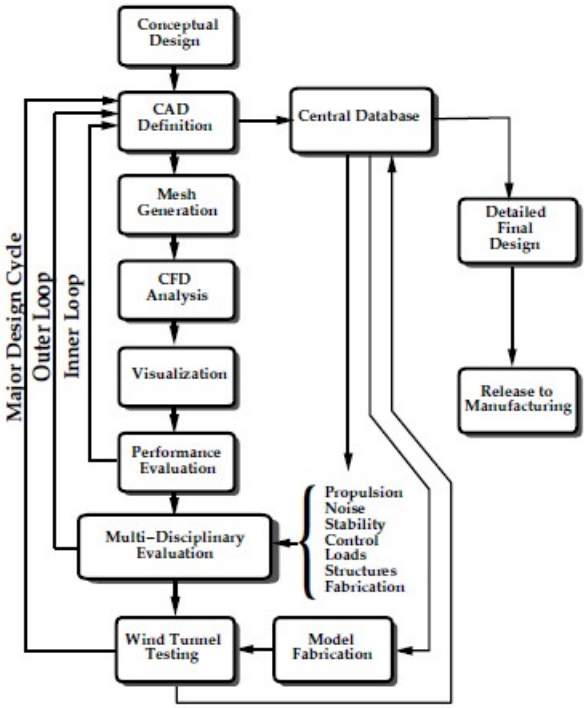


Figure 2: Aerodynamic Design Process

Figure 2 (Jameson, n.d.) above shows the aerodynamic design process and the starting point in the process is CAD design. The inner design loop shown within the process requires that various design variations be evaluated for the wing definition, and each definition requires meshing prior to carrying out any CFD analysis. The results are then shown via a GUI of the software used for the CFD analysis.

CFD has been used in the Aerospace industry since the early 1960s, and it has been a useful tool that provides highly accurate results specifically in cruise conditions where the aircraft flaps are not deployed (Tucker & Liu, 2005). However, the challenge now lies with finding a way to get accurate results when the wings and flaps are at a high angle of attack or when the landing gear has been deployed. Based on the tests carried out in the above circumstances, the accuracy of the results are low where turbulence models cause the highest level of inaccuracy (Tucker & Liu, 2005). Most turbulence models use Navier-Stokes equations that predict the results based on time averaged quantities.

In recent years, with an improvement in computational power, carrying out Navier-Stokes equations for CFD models, a user can carry out high performance computing at a fraction of a cost (Mangani et al., 2021). Furthermore, with GPGPU now used specifically to speed-up scientific simulations it provides a useful platform to be able to carry out simulations with GPGPU acceleration.

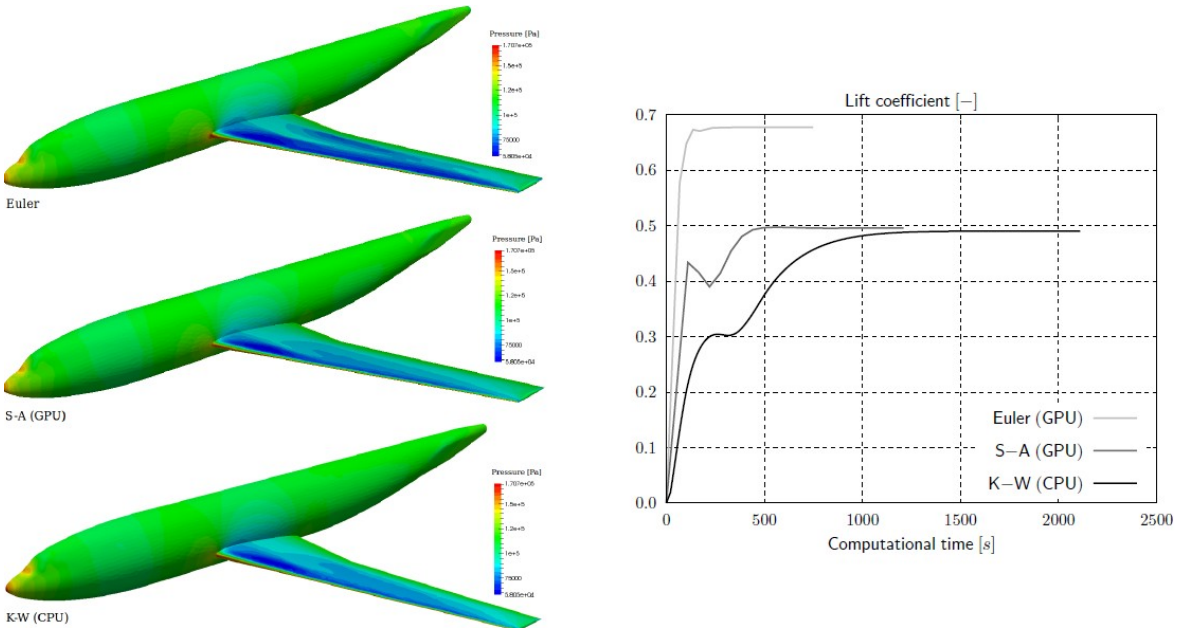


Figure 3: Comparison between CPU and GPU simulations

As can be seen from figure 3, it shows three different simulations of lift coefficients. The S-A simulation on the GPU uses density -based solver model with single precision and the K-W simulation uses the CPU with 4 processes and a double precision accuracy. The GPU used is AMD 290X and the CPU used is Intel i7 3930k (Mangani et al., 2021). The simulations carried out on the GPU are much faster when compared to the simulation carried out solely on the CPU.

The usage of CFD is not limited to full scale aircrafts only, a study carried out to design a small-scale propeller for a UAV has provided promising results (Kutty & Rajendran, 2017).

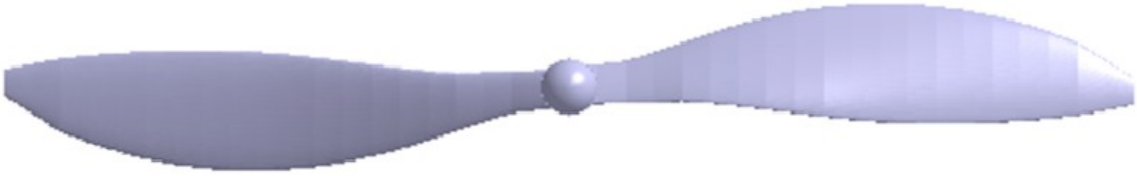


Figure 4: A 10 x 7 inches propeller blade

The above model was then meshed, and the tests were run using Ansys FLUENT, following on from which the figure 5 below was obtained.

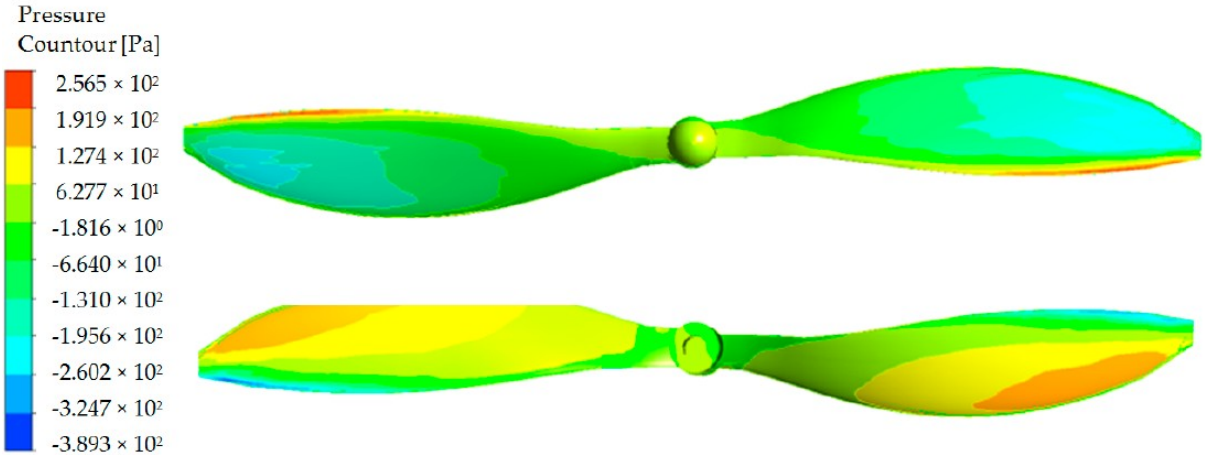


Figure 5: Propeller blade, (Top) front-side of propeller, (Bottom) back-side of the propeller

As can be seen from figure 5 above, the top side of the blade shows that it has less pressure when compared to the back side of the propeller. This is the expected result, as the pressure difference between the back and front of the propeller is what creates lift on an aircraft which allows the aircraft to fly (Kutty & Rajendran, 2017).

2.3. CAD file formats

There are various CAD file formats that are used within the industry such as STL, STEP, IGES and DXF.

- STL file format is the most used file format for 3D printing. It is made of an array of linked triangles that recreate the surface geometry (Neal et al., 2018). It is an open-source file format that is readily available to the public and can be enhanced and used in almost every CAD software program. All

the commercial as well as the open-source CAD software programs mentioned above can import/export STL format files.

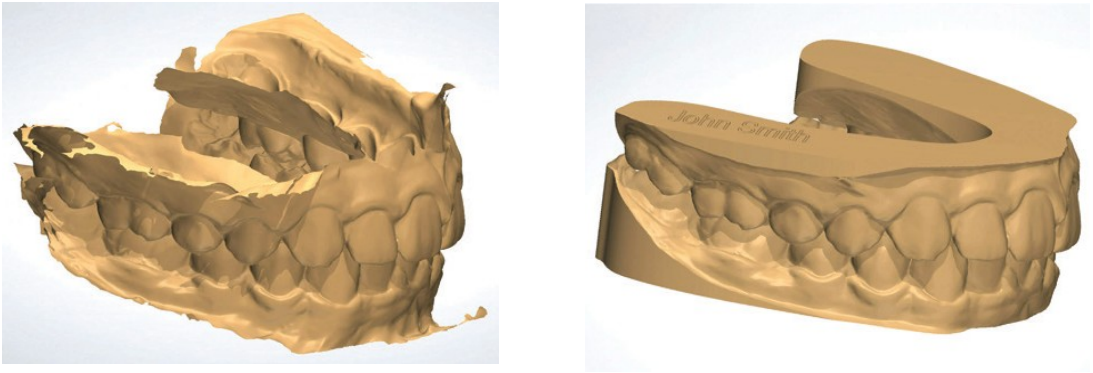


Figure 6: Raw STL file on the left and a repaired and cleaned file on the right (Neal et al., 2018)

Figure 6 shows two STL files, the one on the left is the raw format of the file and the one on the right shows the same file once it has been repaired. The more triangles that are added to the mesh, the smoother it becomes, however it should be noted that this also increases the size of the file. A mesh is used to describe the surface of the model.

- STEP file format is based on an international standard for 3D objects in CAD to be able to exchange data, i.e ISO 10303. The STEP format is made up of all the parts of the model defined individually such as the points, curves and the surface of the model, and then positioned. STEP file provides detailed dates that can be exchanged with other CAD software without the loss of any information. Due to the STEP file format being a part of an international standard, it is the most used format across all industries (Zhou, 2005).

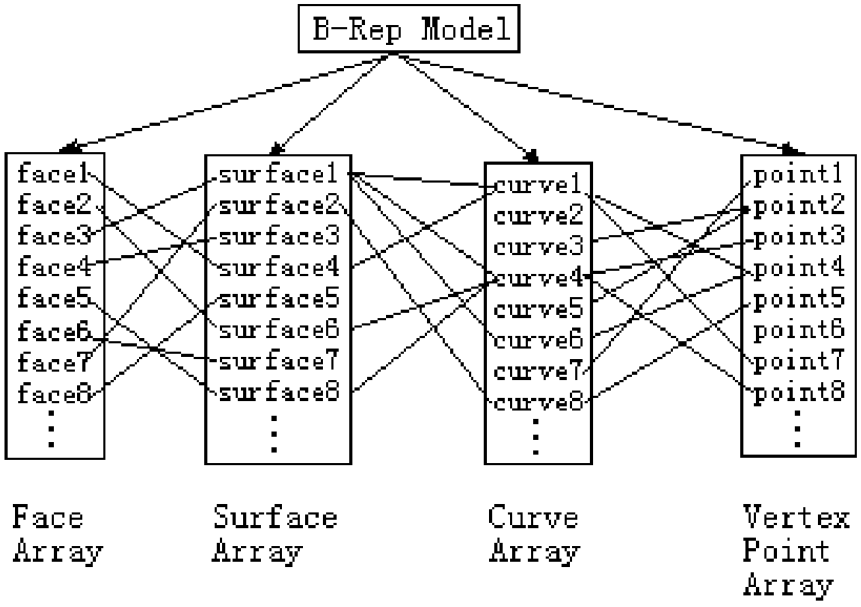


Figure 7: The inner geometry of the data structure of a STEP file (Zhou, 2005)

- IGES file format is similar to the STEP format and it is controlled by the National Computer Graphics Association (NCGA). STEP file format is the newer agreed standard for 3D file formats however, there are many older models that exist in this format and that is the reason that most of the CAD software programs support the IGES format as well as the STEP format (*Chapter Ten. Modeling for Manufacture and Assembly - Modern Graphics Communication, Fifth Edition*, n.d.).
- DXF file format is a proprietary format developed by Autodesk in 1982. It is a text-based format which makes it easier to be able exchange data between CAD software programs that are part of the Autodesk family of software (*A Brief History of AutoCAD | Scan2CAD*, n.d.).

2.2 . Pre-processing

To be able to carry out analysis on a solid model, it is necessary to carry out the pre-processing work on the model. There are various commercial and open-source software that can carry out this pre-processing work, and in some cases this process can be automated. Some of the open-source software that can be used for pre-processing are FreeCad, GMSH and Salome. The commercial software includes Ansys, SimLab, HyperMesh, SimScale and Simula Abaquas.

Pre-processing is one of the steps when carrying out design work. The process includes: CAD solid modelling (although a pre-existing model can also be used with some amendments), meshing the solid CAD model and post-processing (*Chapter 5. Rapid Prototyping - Product Manufacturing and Cost Estimating Using CAD/CAE*, n.d.).

1) CAD model – this can be an existing model with necessary changes required for the specific design or, a new CAD model created.
2) Meshing – for the purposes of consistency, the most used mesh file format that is used across the industry is the STL file format. Therefore, the CAD model (usually in STEP, IGES or similar file format) is converted into the STL file format. Meshing using triangular facets to represent the 3D geometry of a solid model, the more triangles would mean better approximation of the shape, however there is one disadvantage in that more triangles also mean bigger file size (*Chapter 5. Rapid Prototyping - Product Manufacturing and Cost Estimating Using CAD/CAE*, n.d.). Therefore, a fair compromise must be made between the size of the file and granularity of the CAD model.

2.3 . Open Cascade

Open Cascade is an open-source 3D modelling kernel, specifically designed for the development of dealing with 3D CAD (*Open CASCADE Technology - Open Cascade*, n.d.).

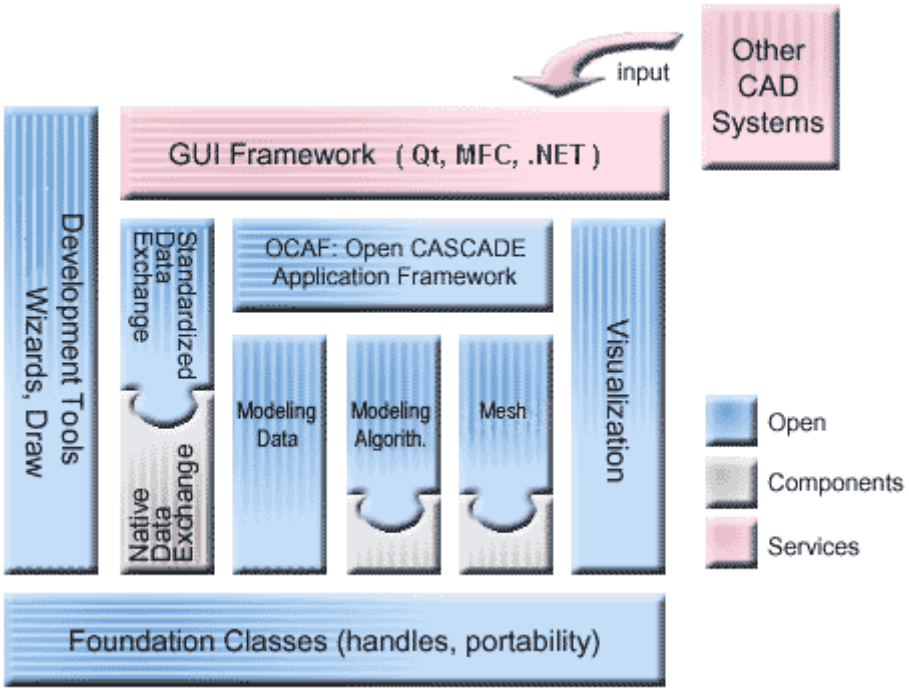


Figure 8: Open Cascade Architecture

Figure 8 above shows the structure of Open Cascade and how it can be used for the development of CAD software.

- Foundation Classes (module) – These are the core classes essential for the portability of OCCT, it saves various primitive information such as string classes, types of colour, date and time information and exception classes. Moreover, it also provides various other more general-purpose services such as run-time information, tools to deal with configuration files, memory allocation and the handling of dynamically created objects (*Open CASCADE Technology: Introduction*, n.d.).
- Modelling data (module) - This module provides the data structures for the boundary representation (BRep) of the 3D objects, which is the geometry of the shape within topology. It provides geometry types and services such as point, vectors, algorithms of construction, computation of point coordinates. Topology is used to define the relationship between shapes such as the Vertex, Shell, Solid and so on. (*Open CASCADE Technology: Introduction*, n.d.).
- Modelling algorithms (module) – This module provides the algorithms for the geometry and topology. OCCT has two kinds of algorithms: high-level and low-level mathematical support functions. The low-level algorithms

provide support for the calculation of intersection of two curves, construct lines and circles, find planes for the edge's location, convert shapes (*Open CASCADE Technology - Open Cascade*, n.d.).

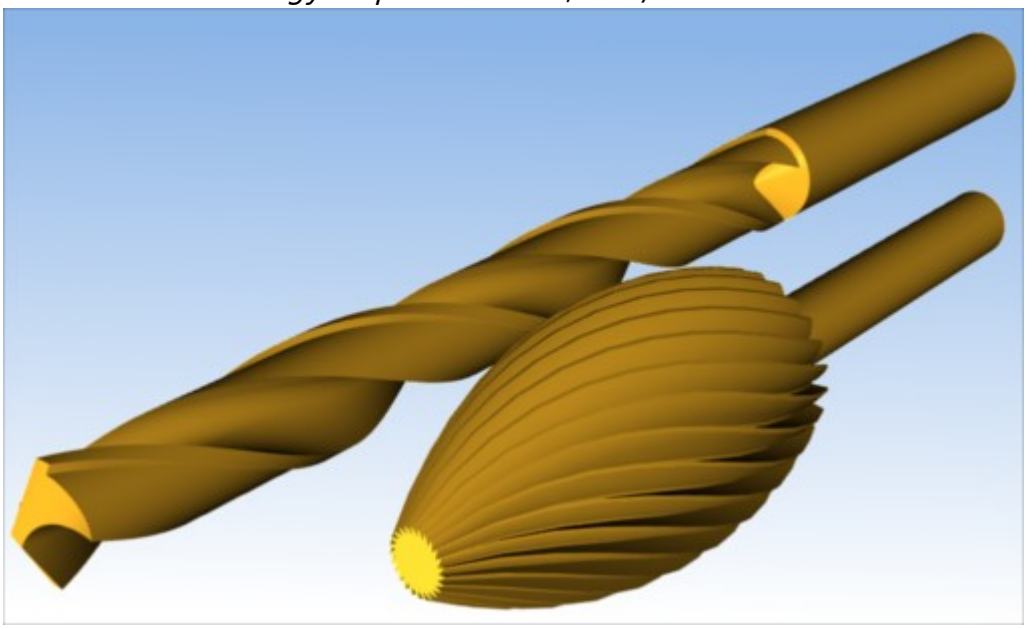


*Figure 9: Shape showing the use of the sweeping function (*Open CASCADE Technology: Introduction*, n.d.)*

The high-level algorithms provide support for functions such as hollowing, shelling, tapered shapes, and fillets.

- Mesh (module) – This module provides the functions to be able to represent a shape with triangular facets. It contains support to store the surface mesh data, to be able to build the surface mesh and for displaying the meshed shape (*Open CASCADE Technology: Introduction*, n.d.).

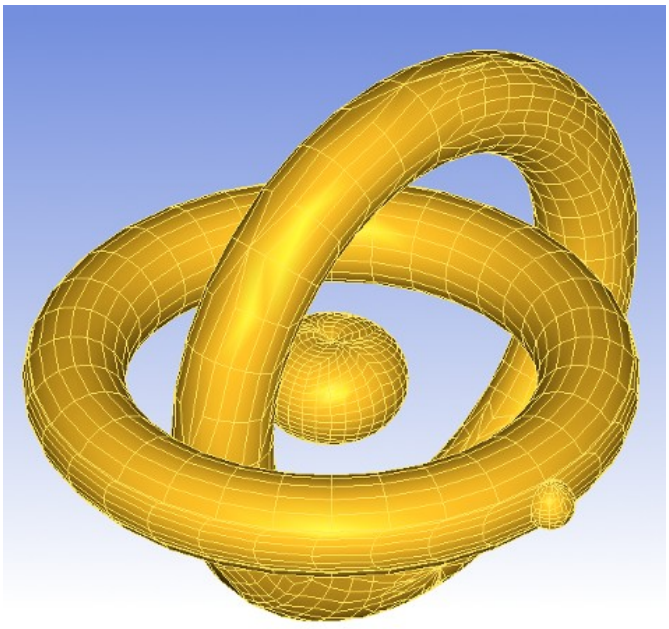


*Figure 10: Mesh module example (*Open CASCADE Technology: Introduction*, n.d.)*

- Visualization (module) – This module is used to create the graphics representation of a CAD model. To carry out this task, the visualization module makes use of other external toolkits such as OpenGL (*Open CASCADE Technology: Introduction*, n.d.).
- Data Exchange (module) – This module is used to comply with various CAD data formats such as IGES, STEP, STL and so on. It provides the capability of exchanging data across various CAD software packages while making sure there are no problems caused by the differences in the model (*Open CASCADE Technology: Introduction*, n.d.). It also provides a library called shape healing that can correct shapes that are imported from other CAD systems. This includes analysing the shape characteristics, fix and complete shapes and upgrade shape characteristics.
- Application Framework – This framework provides various services such as data managing, data storage, managing multiple documents and copy-paste functions (*Open CASCADE Technology: Introduction*, n.d.).

2.4. History of CAD

The development work for CAD/CAM started as far back the 1950s when Patrick Hanratty developed PRONTO (Program for Numerical Tooling Operations) and it is estimated that over 70% of all current CAD/CAM systems can trace their roots back to PRONTO (*How CAD Has Evolved Since 1982 — Past, Present & Future | Scan2CAD*, n.d.).

Following on from the development of PRONTO, Ivan Sutherland developed Sketchpad in 1963 as part of his thesis at MIT where users could interact with the program using a screen, a pen, and buttons. Although Sketchpad never became available commercially, it provided a baseline on which many CAD programs were based.

In the early 1980s when IBM PCs were introduced, it kickstarted a new era for CAD programs. The CAD market was revolutionised and led to a large-scale adaption of CAD across various industries.

The commercialisation of the CAD software led to an increase in other industries buying and using CAD. In the late 1980s, Boeing used CATIA to design and draft its 777 aircraft. In the early 1990s, with the improvement in the computation power of PCs, this led to 3D CAD programs. In the early 1990s, the release of Solidworks on Windows brought another revolution within the industry, as this provided access to CAD to millions of Engineers across the World due to its price and availability on Windows (*How CAD Has Evolved Since 1982 — Past, Present & Future | Scan2CAD*, n.d.) .

In the 21st Century, with the growth and requirements of CAD and the internet, this has led to Cloud based systems that are more readily available online. Due to its availability to better pricing, CAD is now being used in even more industries such as fashion, retail, entertainment and so on (*How CAD Has Evolved Since 1982 — Past, Present & Future | Scan2CAD*, n.d.).

The future of CAD is looking bright with the rise of Augmented and virtual reality allows CAD models to be visualised in real-time. The usage of smart phones and tablets for AR has seen many successes over the last few years, including the introduction of the game Pokemon GO that was a huge surprise within the industry due to its enormous success. Virtual reality allows the user to get into a virtual world that requires a headset. This has also been very successful in the gaming industry with availability for the hugely successful Playstation 4. Due to the success seen in AR and VR, the continuous usage of CAD programs is only going to lead to further improvements in the software (*How CAD Has Evolved Since 1982 — Past, Present & Future | Scan2CAD*, n.d.) .

2.5. Open-Source CAD software programs

There are various well-known open-source CAD software programs that are available, these are well maintained by enthusiasts and are updated at regular intervals. Some of the open-source CAD software programs are Salome, FreeCad, GMSH, GCad3D, LibreCAD, OpenSCAD etc.

2.5.1. Salome

Salome is an open-source CAD framework that is used across various industries, including but not limited to nuclear power plants, turbines etc. Salome is composed of 8 modules, 2 of which are the core modules, these are the Kernel and Gui modules (Zhang et al., 2019). The other 6 modules are Geom, Mesh, Med, SuperV, Post-pro and Yacs.

Salome's interface is built over the OpenCascade open-source software, and the modelling can be carried out either through the GUI or a Python interface

(Bergeaud & Tajchman, 2007). The geometry can be created or loaded and the standards that are supported are STEP and IGES. The Salome Software has been written using Python, C++ and Fortran which supports both surface modelling as well as volume modelling.

Salome makes use of CORBA (Common Object Request Broker Architecture) standard for the design of the architecture, which means that the Salome software is cross-platform, cross-operating system and cross-language (Python and C++). The advantage of using the CORBA standard is that it is used to define a set of APIs, objects and communication protocols that can be implemented using different programming languages, which can run on different platforms and operating systems (Hu, 2019).

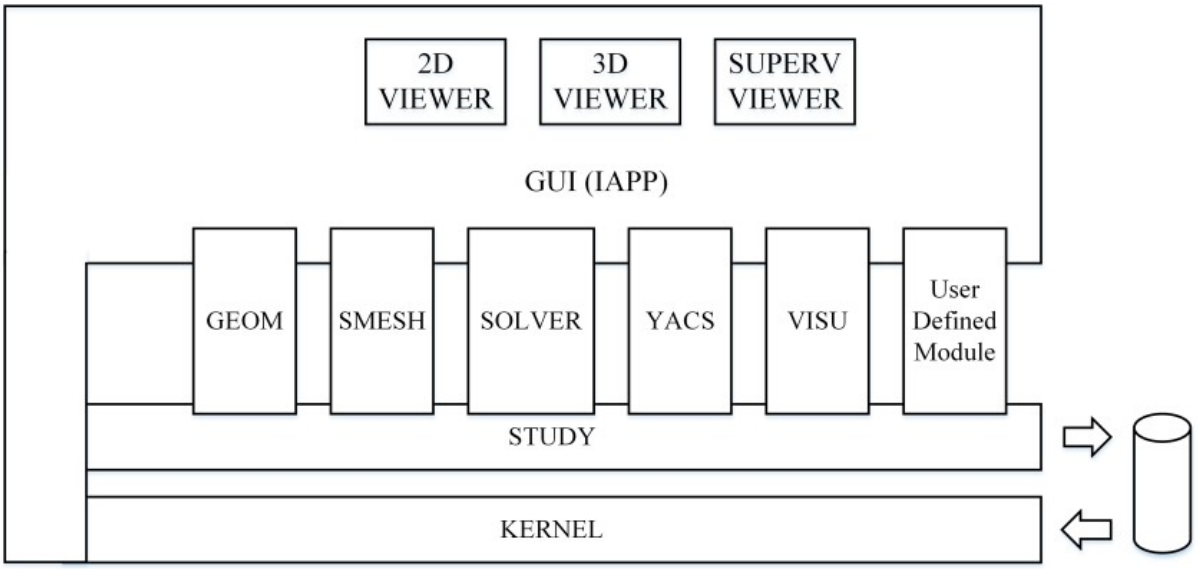


Figure 12: The Salome software architecture (Hu, 2019)

The modules that are available in Salome are as below (*Architecture — SALOME Platform*, n.d.):

- GEOM – This module is used to create, visualise, and modify a geometric shape in the CAD model. It has various uses such as importing different format of CAD files, carrying out shading, displaying/erasing a model and carrying out operations such as extrusion and revolution.
- MESH – The purpose of this module is to create meshes of the geometrical models. It has functionalities such as visualising meshes in 3D, computing meshes, exporting and importing meshes.
- MED – The purpose of this module is to provide a standard way of storing and recovering data and a way of sharing data between codes and solvers. The standard is based on the HDF format that is used by companies such as Boeing and NASA.

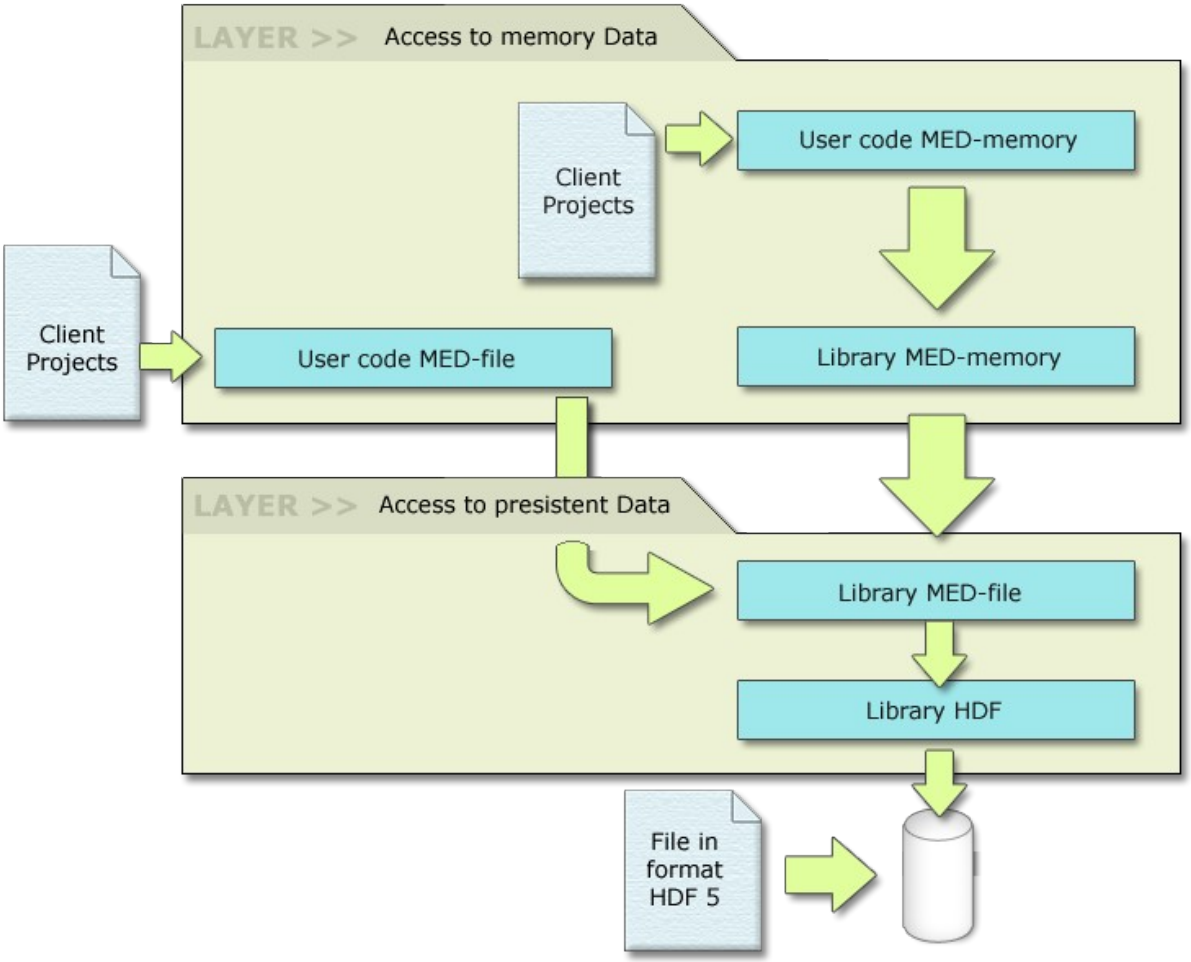


Figure 13: Breakdown of the MED module

- Post-Pro module – This module provides the tools required to visualise the results that are carried out after a numerical computation. This includes 2D and 3D shapes and it also imports ASCII files for curve representation.
- YACS – This module is used to build, edit, and execute calculation schemas that are used for the coupling of computer code.

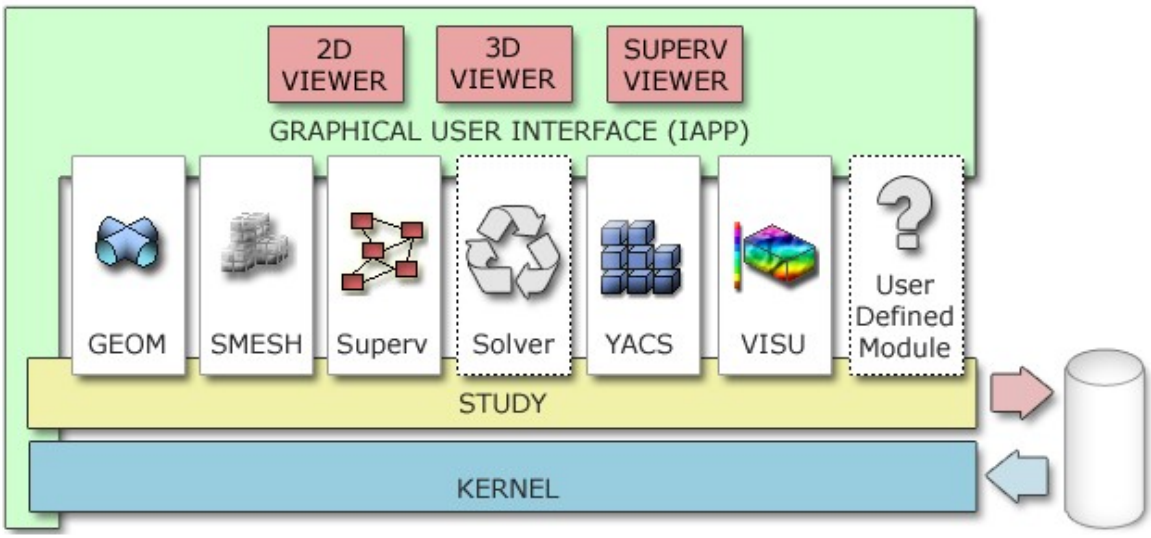


Figure 14: Breakdown of the Salome architecture

Salome provides the method to be able to use Python to build and modify shapes as well as import and export various file formats. Below is a Python script that shows the method to import and export CAD files (*SALOME Geometry User's Guide: Import/Export*, n.d.):

Importing and Exporting CAD files

```
# Import/Export

import salome
salome.salome_init()
import GEOM
from salome.geom import geomBuilder
geompy = geomBuilder.New(salome.myStudy)

import tempfile, os

# create a sphere
sphere = geompy.MakeSphereR(100)

tmpdir = tempfile.mkdtemp()

# export sphere to the BREP file
f_brep = os.path.join(tmpdir, "sphere.brep")
geompy.ExportBREP(sphere, f_brep)

# export sphere to the IGES v5.3 file
f_iges = os.path.join(tmpdir, "sphere.iges")
geompy.ExportIGES(sphere, f_iges, "5.3")

# export sphere to the STEP file, using millimeters as length units
f_step = os.path.join(tmpdir, "sphere.step")
geompy.ExportSTEP(sphere, f_step, GEOM.LU_MILLIMETER)

# export sphere to the binary STL file, with default deflection coefficient
f_stl1 = os.path.join(tmpdir, "sphere1.stl")
geompy.ExportSTL(sphere, f_stl1, False)

# export sphere to the ASCII STL file, with custom deflection coefficient
```

```
33 f_stl2 = os.path.join(tmpdir, "sphere2.stl")
34 geompy.ExportSTL(sphere, f_stl2, True, 0.1)
35
36 # export sphere to the VTK file, with default deflection coefficient
37 f_vtk1 = os.path.join(tmpdir, "sphere1.vtk")
38 geompy.ExportVTK(sphere, f_vtk1)
39
40 # export sphere to the VTK file, with custom deflection coefficient
41 f_vtk2 = os.path.join(tmpdir, "sphere2.vtk")
42 geompy.ExportVTK(sphere, f_vtk2, 0.1)
43
44 # export sphere to the XAO file
45 f_xao = os.path.join(tmpdir, "sphere.xao")
46 geompy.ExportXAO(sphere, [], [], "author", f_xao)
47
48 # import BREP file
49 sphere_brep = geompy.ImportBREP(f_brep)
50
51 # import IGES file
52 sphere_iges = geompy.ImportIGES(f_iges)
53
54 # import STEP file, taking units into account
55 sphere_step1 = geompy.ImportSTEP(f_step)
56
57 # import STEP file, ignoring units (result is scaled)
58 sphere_step2 = geompy.ImportSTEP(f_step, True)
59
60 # import STL files
61 sphere_stl1 = geompy.ImportSTL(f_stl1)
62 sphere_stl2 = geompy.ImportSTL(f_stl2)
63
64 # import XAO file
65 ok, sphere_xao, sub_shapes, groups, fields = geompy.ImportXAO(f_xao)
66
67 # clean up
68 for f in f_brep, f_iges, f_step, f_stl1, f_stl2, f_vtk1, f_vtk2, f_xao:
69  os.remove(f)
70 os.rmdir(tmpdir)
```

2.5.2. FreeCad

FreeCad is an open-source CAD software, that is popular across various industries, and it has been for over 18 years. FreeCad is built using the Open Cascade Kernel open-source software, Open Cascade is used by open source as well as commercial CAD software. FreeCad was designed and inspired based on the Catia V4 Dassault software (di Donato & Abita, 2019) .

FreeCAD is built on top of various external libraries such as OpenCASCADE, OpenInventor, QT, PySide and Xerces XML amongst others.

- OpenInventor – The rendering work carried out in FreeCAD uses the Coin3D library, which is a part of OpenInventor. The reason for using this library is so that a developer who may be writing a new module for FreeCAD does not have to write extra code in OpenGL for the rendering of a model. Moreover, to carry out this development work, a python wrapper is needed as Coin3D is a C++ library and this wrapper is called Pivy (*FreeCAD_Mod_Dev_Guide/FreeCAD_Mod_Dev_Guide__20190912.Pdf at*

Master · Qingfengxia/FreeCAD_Mod_Dev_Guide · GitHub, n.d.). Although there are various other libraries that can be used to carry out the rendering work, the reason for using OpenInventor is because of its open-source licence as well as its performance which works well with the python wrapper Pivy. Furthermore, at the time of FreeCAD's release back in 2002, OpenCASCADE kernel did not provide rendering and therefore, OpenInventor was used instead.

The modules that are available in FreeCAD are (*FreeCAD_Mod_Dev_Guide/FreeCAD_Mod_Dev_Guide__20190912.Pdf at Master · Qingfengxia/FreeCAD_Mod_Dev_Guide · GitHub*, n.d.):

- Part – Used to make simple shapes such as cubes, cylinders and so on. This module is solely based on the OpenCASCADE kernel.
- OpenSCAD – This module uses the OpenCASCADE API to use extra functions that are required for the part module.
- PartDesign – Module used to design more complex 2D sketches, to produce a single continuous solid shape. This module works closely with the Sketcher module described below.
- Draft – This module is used to draw and modify 2D objects, the purpose is to be able draw this as a sketch as quickly as possible. However, these drawings can be modified later using various other tools provided within FreeCAD.
- Drawing and Sketcher – The drawing module is used to save 3D models as various formats and the Sketcher module works closely with the PartDesign module to build up a 3D part from a 2D sketch.
- Assembly – Used to be able to assemble parts.
- Mesh – Used to convert parts into triangle meshes used for the purposes of rendering.

It should be noted that there are various other modules in FreeCAD however, they are not visible to users of a workbench such as the Import module.

Due to the popularity of FreeCad within the industry and between hobbyists, there is even a how-to book that is currently available in the market called "FreeCad [How-to]" (*FreeCAD [How-To]*, n.d.). This book provides a breakdown of how the modules are set out, how to use the GUI, how to record the macros and so on. FreeCAD provides the flexibility of being able to convert and run CAD models in most known formats such as stl, step, iges and brep.

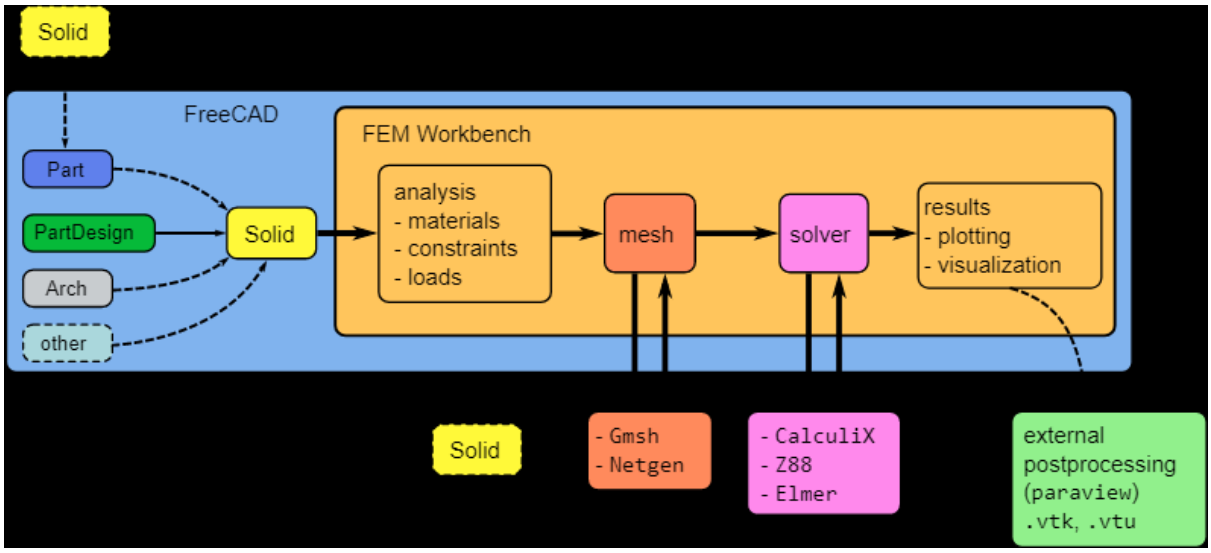


*Figure 15: FreeCAD software architecture (*FEM Module - FreeCAD Documentation*, n.d.)*

Loading and Saving using Python

As FreeCAD provides the capability of using Python to be able to build the shapes, it is also possible to import and export different file formats using Python. Below is a breakdown of some code from extracts that are available at (*Topological Data Scripting - FreeCAD Documentation*, n.d.).

To save a shape in a specific format, in this case .stp format:

```
import Part
s = Part.makeBox(10, 10, 10)
s.exportStep("test.stp")
```

To load a shape in a specific format, in this case .stp format:

```
import Part
s = Part.Shape()
s.read("test.stp")
```

To convert a shape to another format, in this case from .stp to .igs:

```
import Part
 s = Part.Shape()
 s.read("file.stp")       # incoming file igs, stp, stl, brep
 s.exportIges("file.igs") # outbound file igs
```

Furthermore, due to FreeCAD being open source, there are various examples on how to convert files which are also available on stackoverflow, the following

reference provide an accurate example of how to convert shapes using FreeCAD and Python (*Stl Format - Convert STEP File Type to STL - Stack Overflow*, n.d.).

```
import FreeCAD
import Part
import Mesh
shape = Part.Shape()
shape.read('my_shape.step')
doc = App.newDocument('Doc')
pf = doc.addObject("Part::Feature","MyShape")
pf.Shape = shape
Mesh.export([pf], 'my_shape.stl')
```

2.5.3. GMSH

GMSH is an open-source finite element mesh generator, with a CAD engine and a post-processor. Although GMSH has the CAD engine and the post-processing capabilities, its main purpose it to provide a light and fast mesh generation (*Gmsh 4.7.1*, n.d.). GMSH has the capability of carrying out the meshing work automatically and it is a well-known tool within the Open-source space. GMSH uses the “bottom-up” method of carrying out the mesh function, i.e. the discretisation of the curves takes place first followed by the mesh of the surfaces.

It should be noted that although GMSH is well known, FreeCad and Salome dominate the open-source space. The reason that GMSH is not quite so popular as the other open-source software is because, although it provides a CAD engine, it is not quite as well refined as the FreeCad and Salome software, as its focus is providing a light and fast meshing tool. However, there are also similarities between GMSH and the other open-source software, such as the Kernel used for the Geometry module which is also based on the OpenCascade kernel, and it also provides C++ and Python APIs. However, GMSH does hold an advantage in terms of providing APIs as unlike Salome and FreeCad, it also provides APIs for C and Julia programming languages. GMSH also has an app which provides the user with freedom to be able to build a model on the go, providing the capability for iPhone, iPad and Android devices (*Gmsh 4.7.1*, n.d.).

Moreover, GMSH also has its own built kernel as well as the OpenCascade kernel, and it has its own scripting language that has the file format of. geo. However, the drawback of building a model and saving it as. geo format is that it cannot be exported in this format and will need to be converted and saved as STEP file format, before it can be exported and used for simulation (*Gmsh 4.7.1*, n.d.).

2.6. Commercial CAD software programs

There are various well known commercial CAD software programs that currently exist in the market, with each having their own strengths and weaknesses. Some of the well-known commercial CAD programs are Solidworks, AutoCAD, Catia and Inventor.

2.6.1. Solidworks

Solidworks was founded by Jon Hirschtick in 1993 while he was a member of the MIT Blackjack team. The purpose of wanting to start this company was to be able to provide an affordable CAD software available on the Windows operating system. Solidworks provides CAD capability to various disciplines within the industry, including 3D CAD, Electrical Design, Product Development, Simulation and Technical communication (*A Brief History Of SolidWorks | Scan2CAD*, n.d.) . Solidworks is used in various industries including Aerospace, construction, and manufacturing. Solidworks is one of the most used commercial CAD software in the market. Since its release in the market, it is well known for its ease of use, where other CAD software would take a long time to learn, Solidworks could be learnt in just weeks, and at its release, the price for a commercial CAD software was around $18000 and Solidworks was only $4000 (*A Brief History Of SolidWorks | Scan2CAD*, n.d.).

Solidworks is user friendly in that it provides a variety of tools to Engineers. The major elements within the Solidworks user interface are the graphics window where the geometry of the part is shown, the FeatureManager that has a list of all the features in a part, PropertyManager for data input and CommandManager where commands for the software is provided (*Chapter 2: Navigating the SolidWorks Interface - Mastering SolidWorks*, n.d.) . A breakdown of all the features is provided in figure 16 below. Solidworks is owned by Dassault Systems that also own CATIA (discussed below).

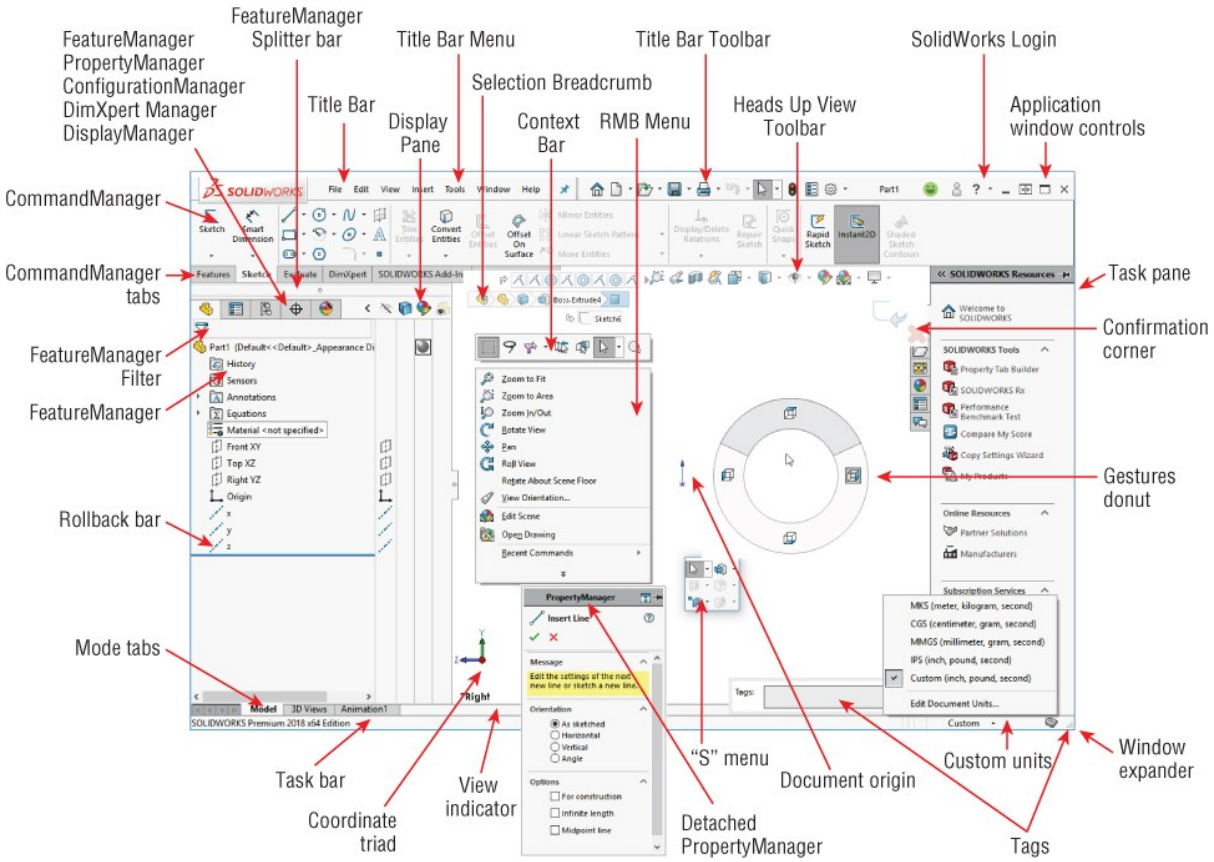


*Figure 16: Solidworks interface (*Chapter 2: Navigating the SolidWorks Interface - Mastering SolidWorks*, n.d.)*

2.6.2. AutoCAD

AutoCAD was the first ever CAD software available on PCs, being in the market since 1982. AutoCAD was developed by John Walker alongside 15 other co-founders who developed five automation applications together hoping that one of them would gain popularity and AutoCAD happens to be that one flagship product that did take off.

*Figure 17: AutoCAD 1982 (*A Brief History of AutoCAD | Scan2CAD*, n.d.)*

AutoCAD was initially designed for Mechanical Engineers however, over the years and due to its success, AutoCAD is being used by professionals across the field including architects, animators, and engineers. AutoCAD is used in various industries such as Construction and Animation. AutoCAD is owned by AutoDesk as is Inventor (discussed below). Figure 18 below provides the user interface breakdown of AutoCAD.

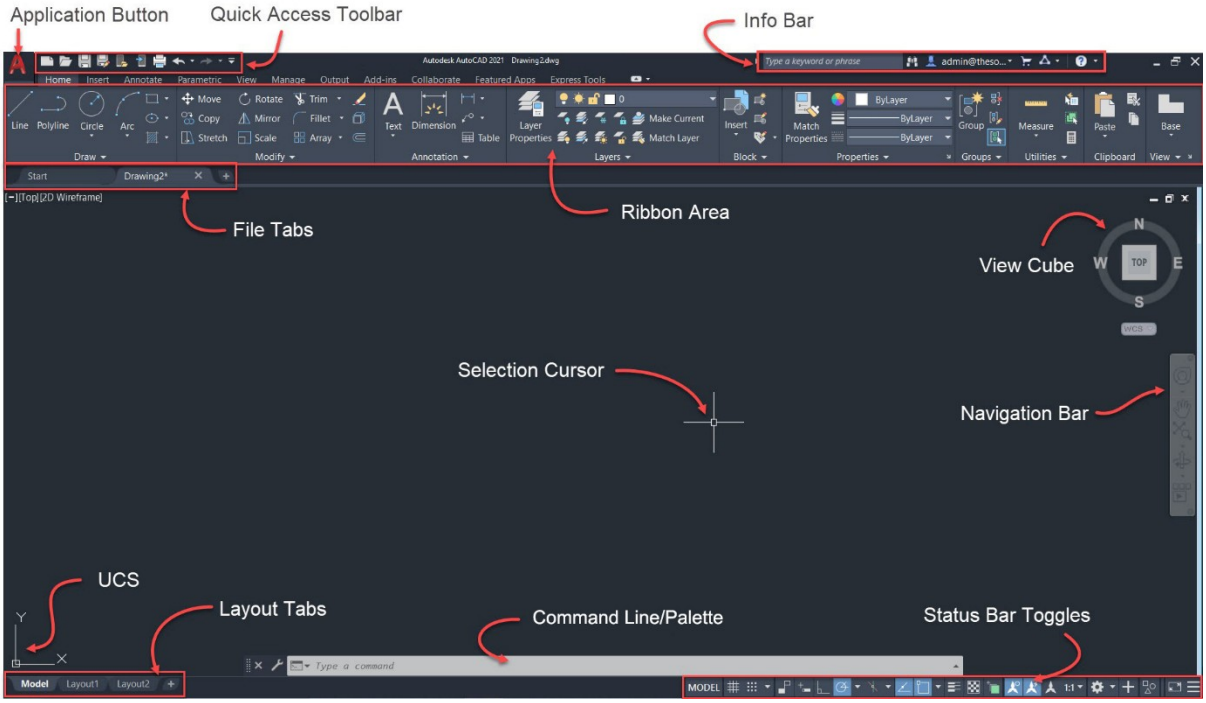


*Figure 18: AutoCAD USER Interface (*Understanding the User Interface - Practical Autodesk AutoCAD 2021 and AutoCAD LT 2021*, n.d.)*

2.6.3. CATIA

The CATIA program was developed in 1977 specifically to design the Dassault Mirage Fighter jet, however, the engineers who designed the software decided to do more than just that, and they created a software that could be used to design and develop more than just one Aircraft and the umbrella company decided to break off its software design arm that led to the creation of Dassault systems that owns CATIA (*CATIA vs SolidWorks | Software Comparison | Scan2CAD*, n.d.). Following on from this creation, CATIA worked closely with Boeing in the late 1980s to improve its Aeronautics design capabilities, and CATIA is currently used heavily within the Aerospace industry.

However, CATIA is more than just a CAD software these days, it has capabilities to manage a whole product lifecycle that provides tools for CAD, CAM and CAE all in one suite. Furthermore, in the latest version of CATIA, it has provided the capability of the Stellar rendering engine, that allows its users to be able to render their product and view them more realistically (*CATIA vs SolidWorks | Software Comparison | Scan2CAD*, n.d.).

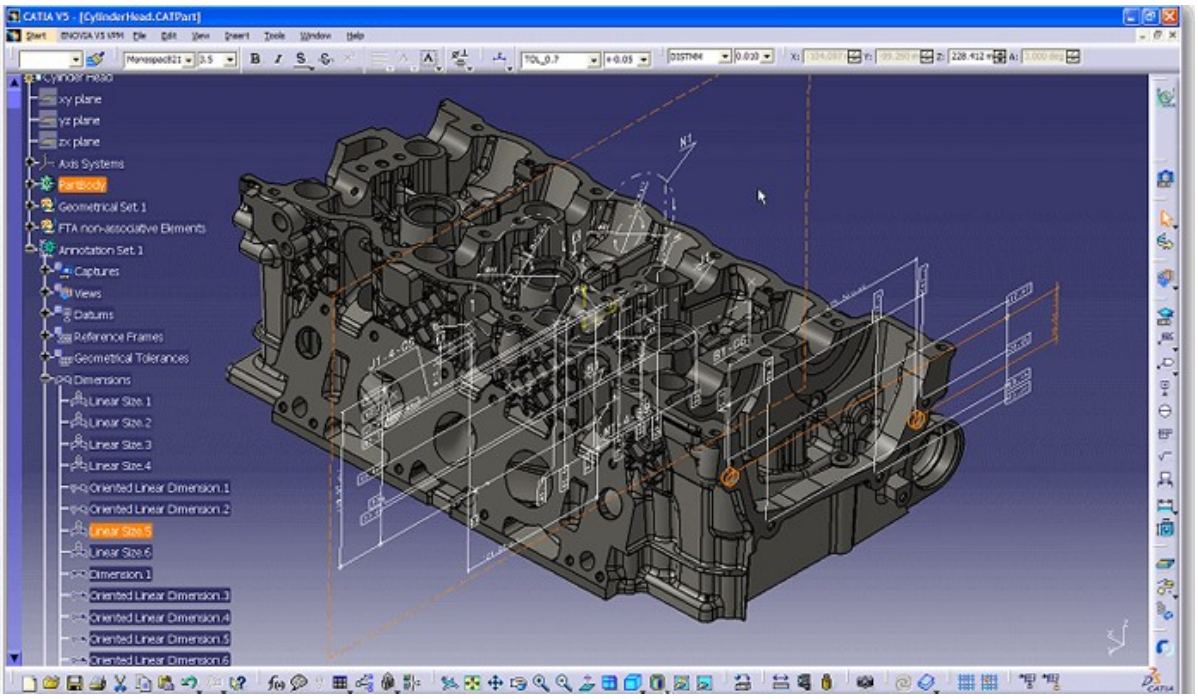


*Figure 19: CATIA V5 (*CATIA, All You Need to Know about This CAD Software - 3Dnatives*, n.d.)*

2.6.4. Inventor

Inventor is owned by Autodesk and was first released in 1999, as a competitor to Solidworks. Inventor is refined on a yearly basis and each new version of the Software is given a code name, the latest version is called “Ada” (*CATIA vs Inventor | CAD Software Compared | Scan2CAD*, n.d.). Inventor is used for designing, visualising, simulating and FEA and can be used for mechanical and electromechanical design.

Inventor is used in various industries such as Automotive, Robotics, Aeronautics, and process engineering. To promote its products within academia, Inventor and almost all the Autodesk suite is available to students free of charge. Furthermore, Inventor is like Solidworks in its ease of use, and it is easier to learn compared to some of its other counterparts such as CATIA (*CATIA vs Inventor | CAD Software Compared | Scan2CAD*, n.d.). The figure 20 below shows the user interface and provides a features breakdown.

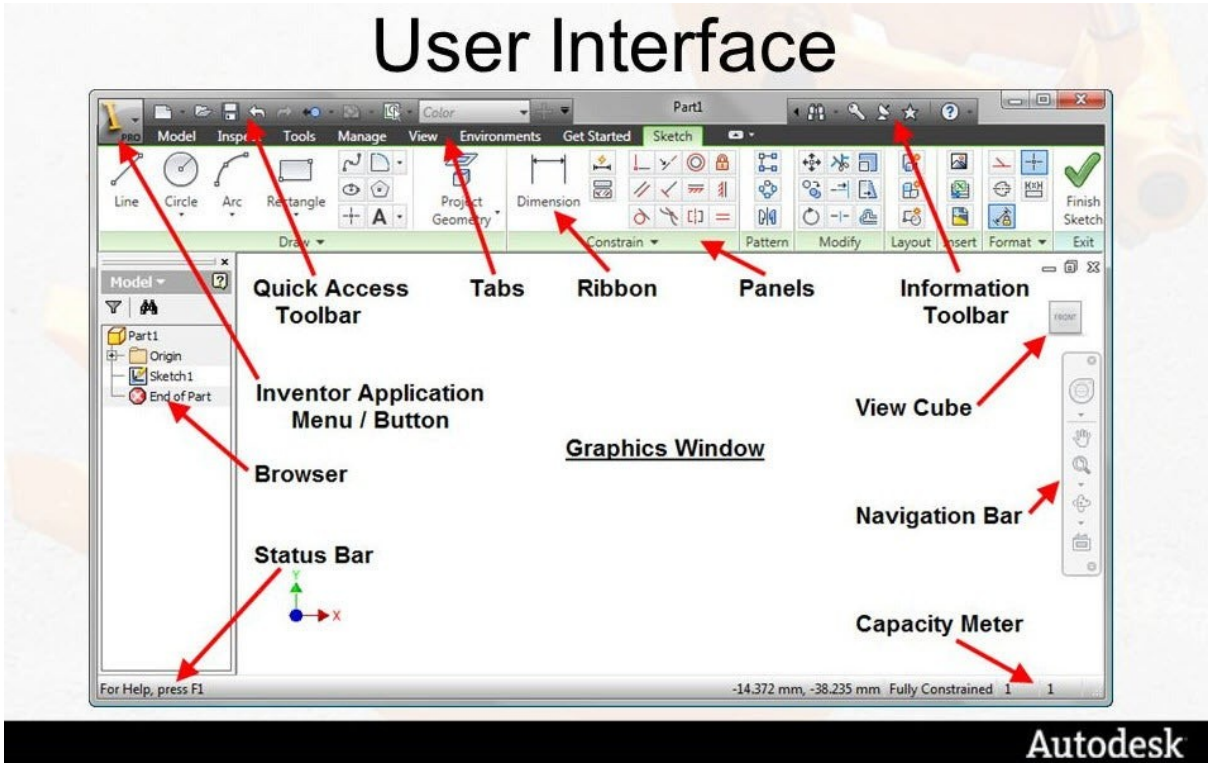


*Figure 20: Inventor User Interface (*Getting Started With Autodesk Inventor (Basics and User Interface) : 5 Steps - Instructables*, n.d.)*

3. Methodology

3.1. Design

To carry out the objectives of this thesis, currently available open-source CAD software modules will be used to convert the CAD file formats. Having carried out a thorough literature review above and based on the information that is readily available, it has been decided that the modules from the FreeCad software will be used. A python script will be written to be able to make use of these modules and file conversions will be available for. step, .stl, .obj, .igs and .brp file formats.

The purpose of the software is to be as portable as possible, although the requirement is to use it for the Digital Wind Tunnel project, it is not to say that it cannot be used in conjunction with other projects. Therefore, the software has been designed using Object Orientated Principles with one Class that has four methods.

The __init__ method is a requirement for all Python classes and in this case, it is used to ask for the input and output file information from the user, and it calls the publish_help_screen (discussed later) method if incorrect information is provided by the user.

The second method is the convertor method, where the bulk of the work to convert CAD files is carried out. It ensures that the user enters at least 3 arguments, including the name of the Python file, an input file name with its format and an output file name with the required format and if incorrect

information is provided it calls the publish_help_screen method. The method calls the Mesh and Part function from FreeCAD, the Mesh function is used to convert Part models into Mesh formats. The part_convertor (discussed later) method is called to convert part models to part formats. The method also measures the amount of time required to convert the files.

The third method is the part_convertor method, this is where part conversion takes place. FreeCAD's part module is made use of here, it has "export" methods within the module. For example, to convert a model to step format, the call would be part.exportStep(output file) and it can also convert other part formats into the step file format.

The fourth method is the publish_help_screen method, it is a simple method that is only used when the user enters incorrect information in the command prompt. It provides a breakdown of how the software is to be used, helping the user, and providing a better user experience.

3.2. CAD conversion process

To be able to make use of the FreeCad modules, it was necessary to understand which modules will be required to be able to convert the CAD files. There are various open-source tutorials and documents that are readily available that provide a breakdown of the scripting and the use of Macros in FreeCad. The FreeCad [how-to] book provides a breakdown of how to use various modules and how to open a new document while using macros. It was decided that the Python script will initially be written and tested in the macros screen within FreeCad, the script that was used to understand how the modules interact is provided across various sources and some of these are shown in literature review above.

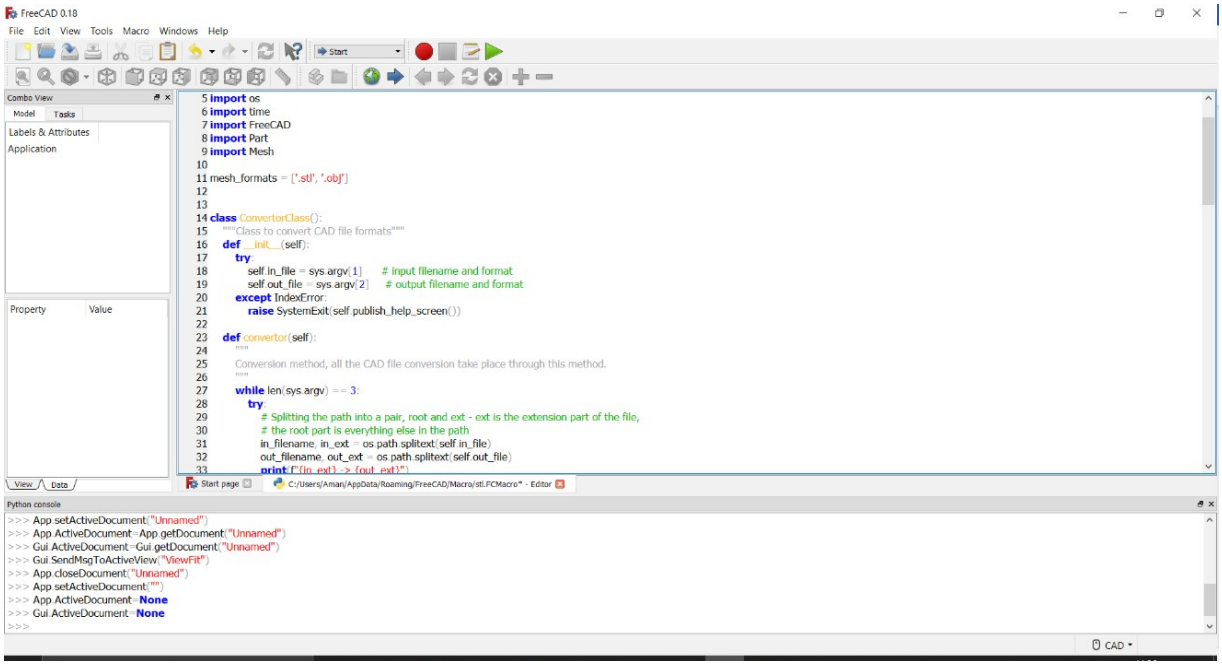


Figure 21: Python macro section of FreeCAD

Once the script has been written and tested within the macros screen in FreeCad, the script will be copied and saved as a .py file. This will then be tested again to make sure that the script still runs as intended, this test will be carried out through the command line. To determine the limitations of the program, the program will first be tested with a simple and small file, i.e. maximum 2mb. If the program passes this test, more complex shapes with much larger file size will be tested until the program eventually fails or if the file is so large that the amount of time it takes to convert the file is inadequate.

A breakdown of how the program works is provided in the steps below:

- Download FreeCad version 0.18 from the following link: https://www.freecadweb.org/downloads.php
- Open the CAD_convertor.py file and change the “FREECAD_PATH” constant to where the FreeCad .pyd files are saved. The .pyd files are saved in the “bin” folder inside the FreeCad folder.

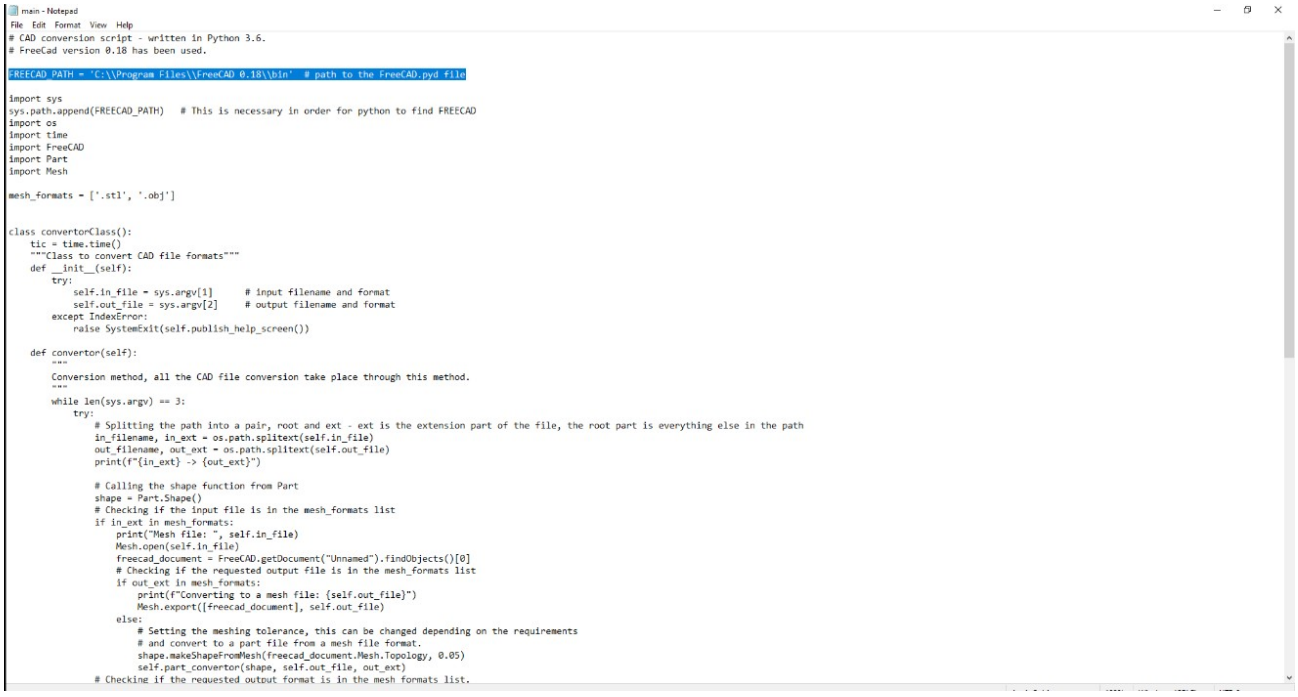


Figure 22: FreeCAD path variable

- Have a file that needs to be converted saved in the same folder as where the CAD_convertor.py file is saved.

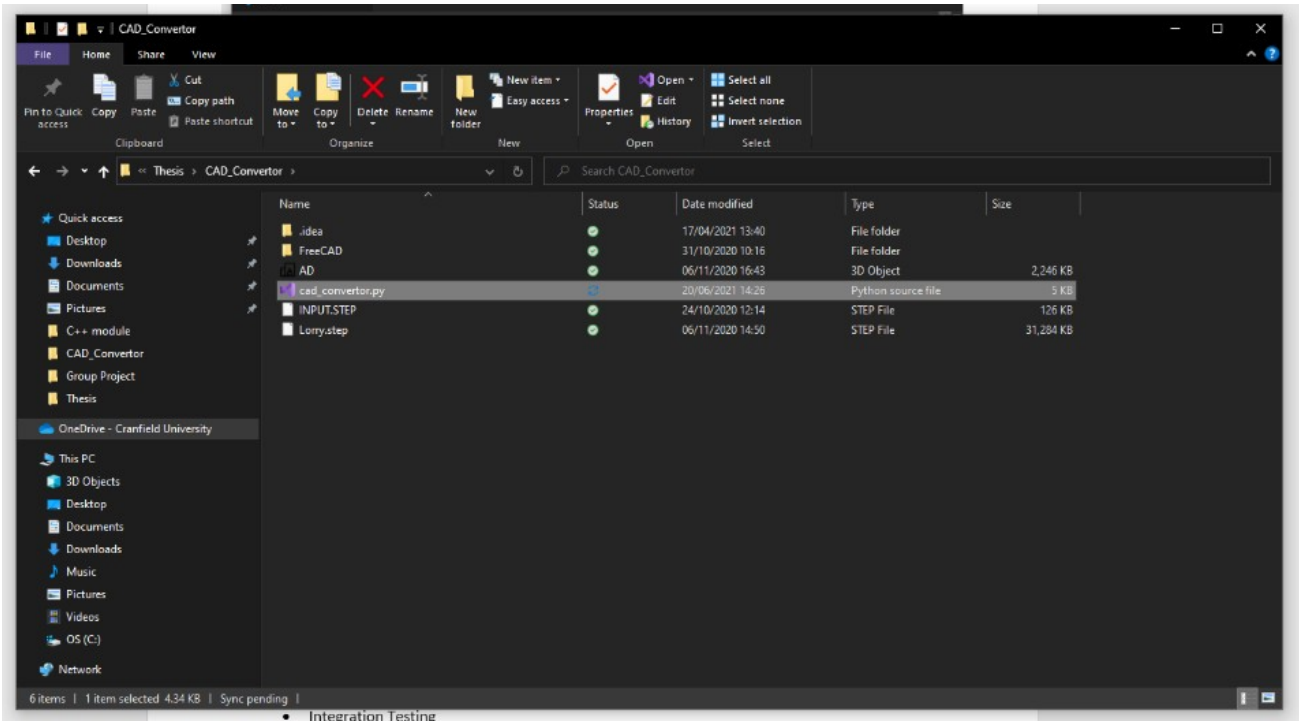


Figure 19: Input file and cad_convertor.py file

- Open a command prompt and navigate to where the CAD_convertor.py file is saved.

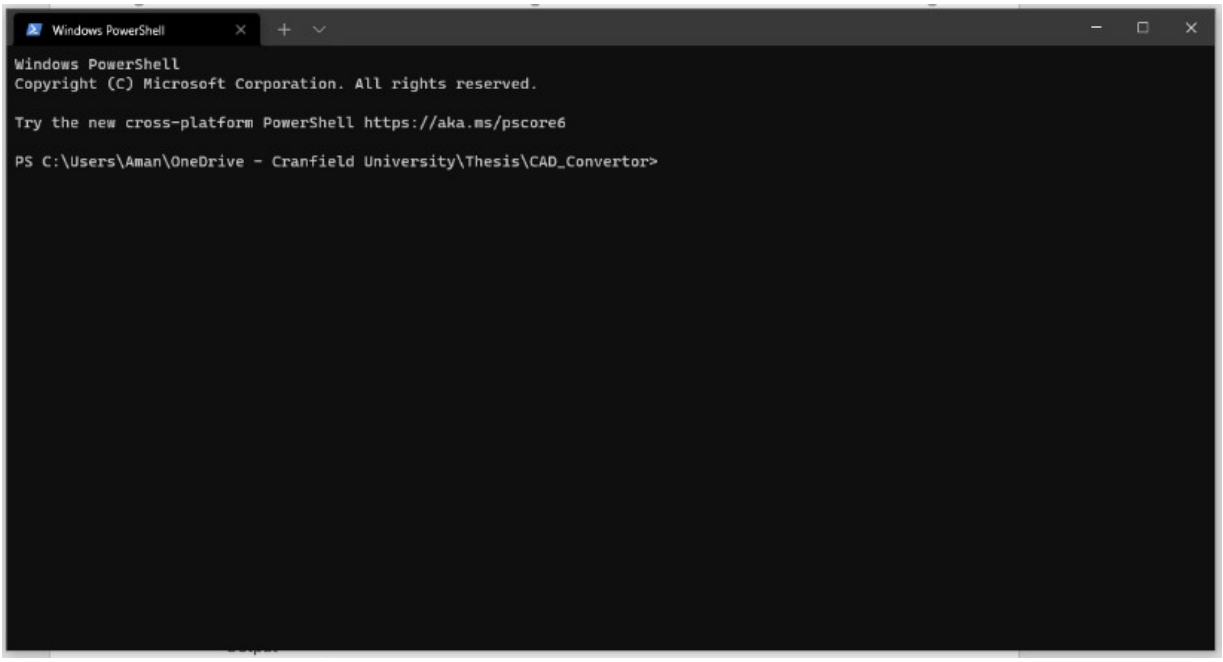


Figure 20: Command prompt pointing to cad_convertor folder

- Run the CAD_Convertor.py file with the following convention – “python cad_convertor.py input.step output.stl”. If the information is put in incorrectly, a help screen will be shown with an example of how to run the file.

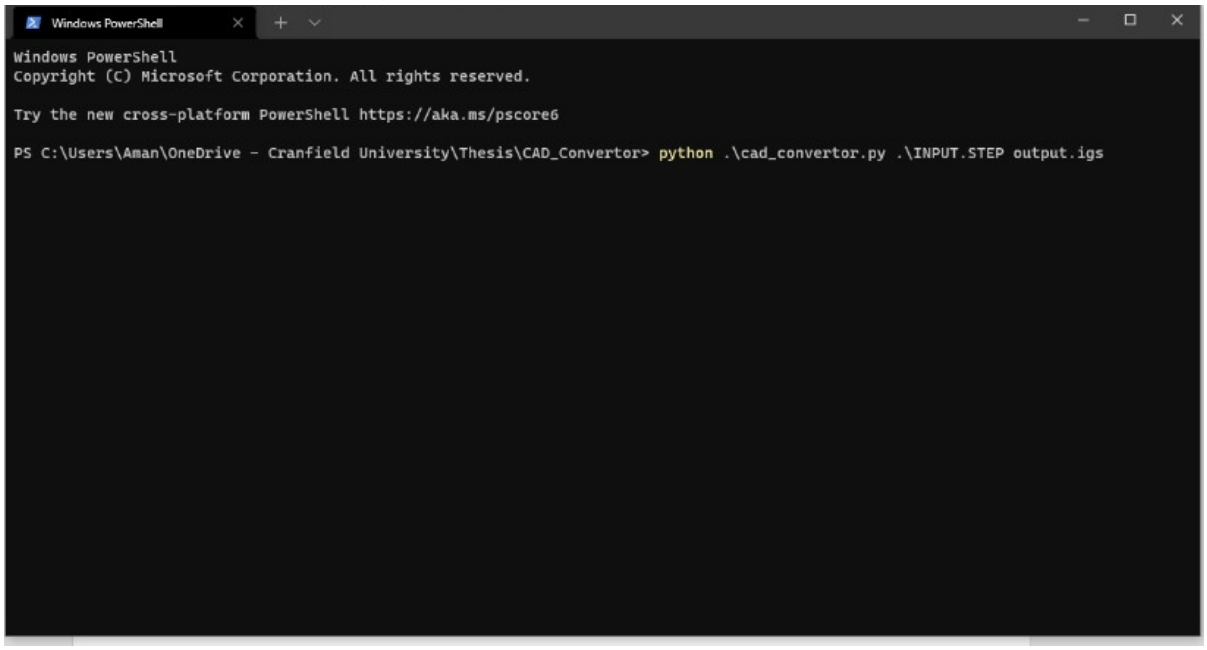


Figure 25: Running the cad_convertor.py script

- Open the folder where the CAD_convertor.py file is saved, and the output file will be in the same folder.

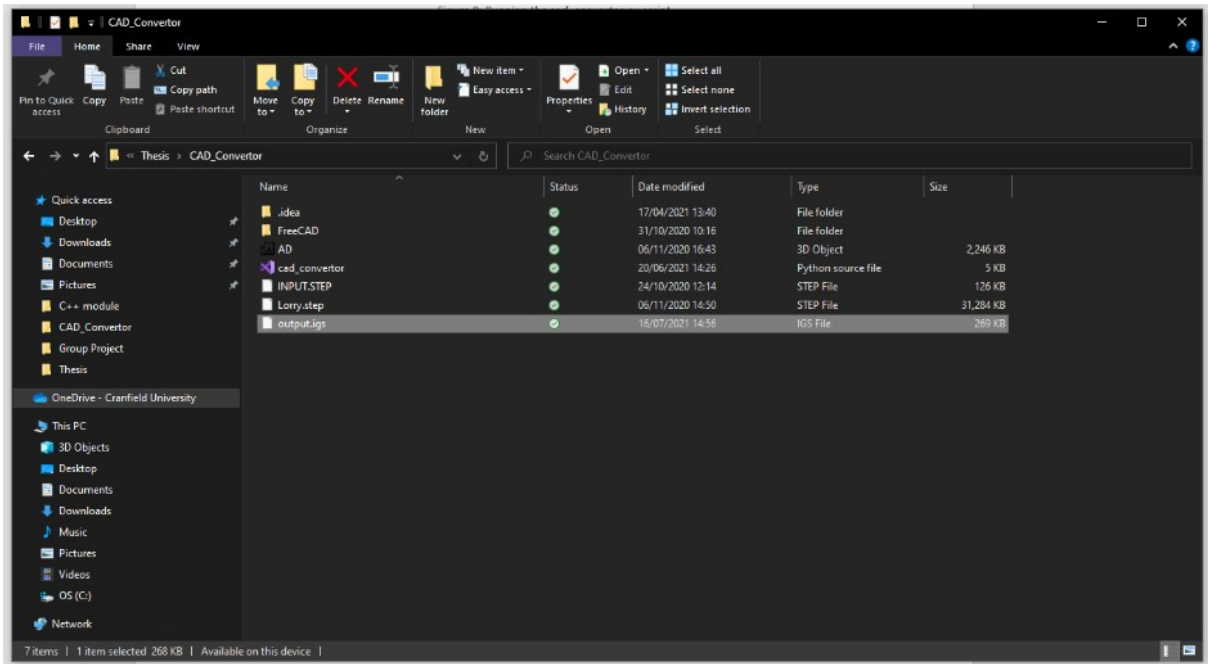


Figure 26: Output file saved in the same folder as the input file

- It should be noted that the Python version must not be any higher than 3.6 as that is the version supported by FreeCAD version 0.18. At the time of writing this, FreeCAD version 0.19 is in works which is expected to support later versions of Python however, this script will need to be re-tested if a newer version of FreeCAD and Python are to be used in the future.

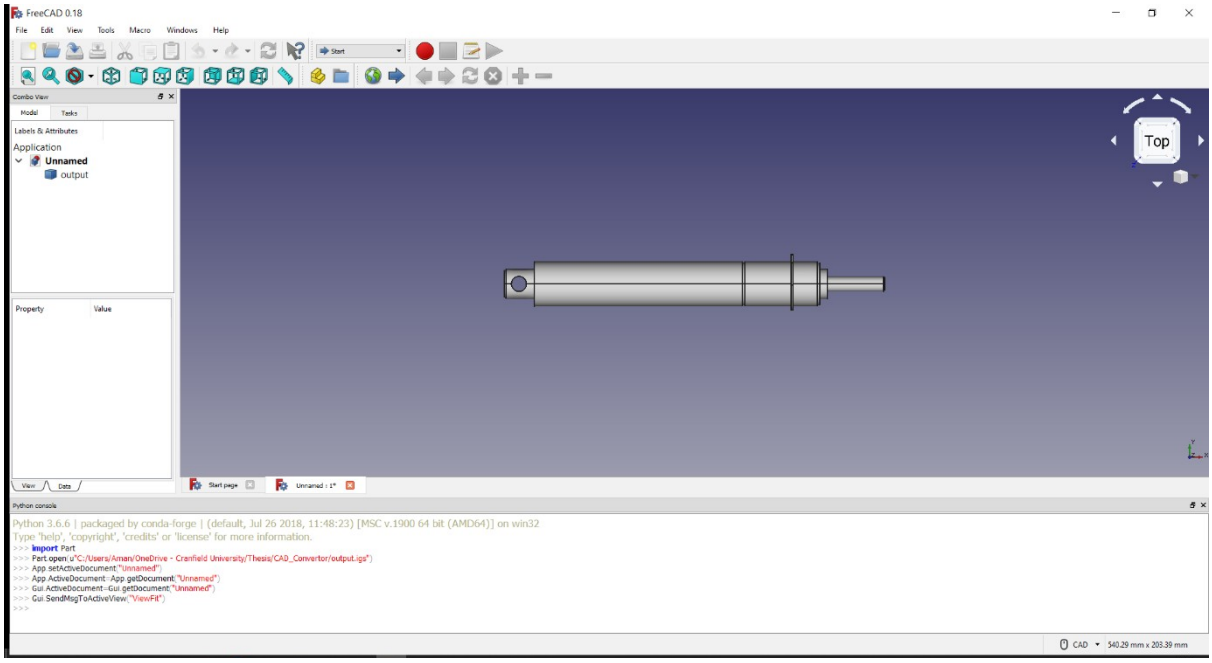


Figure 23: Output file opened with FreeCAD

4. Testing

To ensure that the CAD convertor program runs without any issues, various tests have been carried out. This includes running the tests on different PCs as well as using different operating systems.

4.1. Testing Synopsis

4.1.1. Parts to be tested.

- Input
- Output
- Acceptance testing
- Performance testing
- Integration Testing
- System Testing

4.2. Systems Requirement

As the purpose of this program is to work in conjunction with the software written for the DWT projects, and it must work across various platforms and it will be tested on a combination of different hardware and software.

- Standard off the shelf Alienware Laptop
  Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 2801 Mhz, 4 Core(s), 8 Logical Processor(s). Running Windows 10 operating system on a 64-bit system.
- Standard off the shelf Alienware Laptop

Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 2801 Mhz, 4 Core(s), 8 Logical Processor(s).
Running Ubuntu 20.04 operating system.

- Standard computer at Cranfield University – Running Windows 10 operating system on a 64-bit system.

4.3. Standard/Reference material

The CAD conversion software has been written using Python 3.6 alongside FreeCAD version 0.18 on the Windows 10 operating system using Visual basic community edition 2019.

5. Types of Testing

5.1. Acceptance Testing

The following parts have been tested:

- The program runs on the computer it was developed (with two different OS).
- The program runs on the University Computer.
- Functional tests carried out to test that all the functions provide the expected results.

5.2. Input/Output

Input/Output testing has been carried out to ensure that when the correct input information has been provided, the output provides the expected result. Testing has also been carried out if incorrect input information has been provided, which prints a help screen providing a breakdown of how the program should be used.

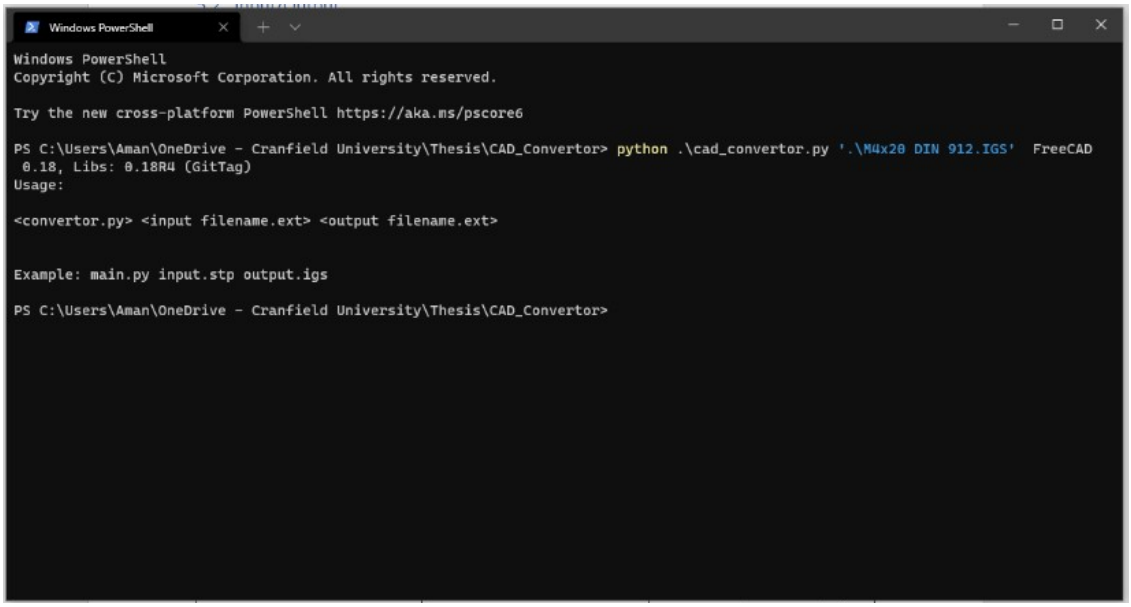


Figure 28: Testing incorrect input, provides a help screen

5.3. Integration Testing

Test Case Objective	Test Case Description	Expected Result
Each function will be tested	The program functionality will be tested individually	Display error message every time the program has been used incorrectly.
Incorrect file format	Provide an output file format that is not recognised by the program.	Throw up an error that states that incorrect file format provided.
Incorrect arguments	Provide incorrect number of arguments when running the program	Should show a help screen that states how many arguments are expected.
Test all accepted file formats	All the accepted file formats will be tested for input and output	All accepted file formats provide an output file that can be viewed on a CAD program.
Non-existent file	Provide a filename that is not in the correct folder	Throw up an error that states that the file does not exist.

5.4. System Testing

Test Case Objective	Test Case Description	Expected Result
Usability testing	The convertor runs as expected with the required inputs	To be able to test if the program is user friendly.
Recovery Testing	The laptop will be shut down while the program is running	To test if the program is affected and if it is recoverable.
Migration testing	The program will run on various computers and operating systems.	Expected that it will provide the same performance across various machines and operating systems.

5.5. Performance Testing

Bigger shapes cannot be converted into mesh files easily. Trying to convert a large part file into a mesh file does not work properly and the bigger part files must be broken down into smaller models before they are converted to stl. Performance testing has been carried out, which includes measuring the performance to convert a file from a part file to another part file, converting a mesh file to part file and converting a part file to a mesh file. The results are provided below in the discussion and results section.

6. Discussion and Results

A working software has been written that solves the given problem statement, i.e. be able to convert CAD files and the software can be used as a plugin for the Digital Wind Tunnel project. To ensure that all the functionalities are met and if the results are expected, a breakdown of various tests is provided below. This includes converting part files to part files, part files to mesh files and mesh files to part files to measure the performance as well as the output. All the same models have been tested for the conversion of CAD models to test if all the models and sizes can be converted, and if they cannot be converted, the reason will be discussed and possible solutions.

6.1. Part to Part conversion

Converting part files to part files worked well when tested, as the size of the file increased so did the time required to convert the files. Figure 29 below is a graph of a comparison where various sizes of step and igs files were converted, in some cases from igs to step and in other from step to igs format. This is expected behaviour from the software as the bigger, the file the longer it would take to convert.

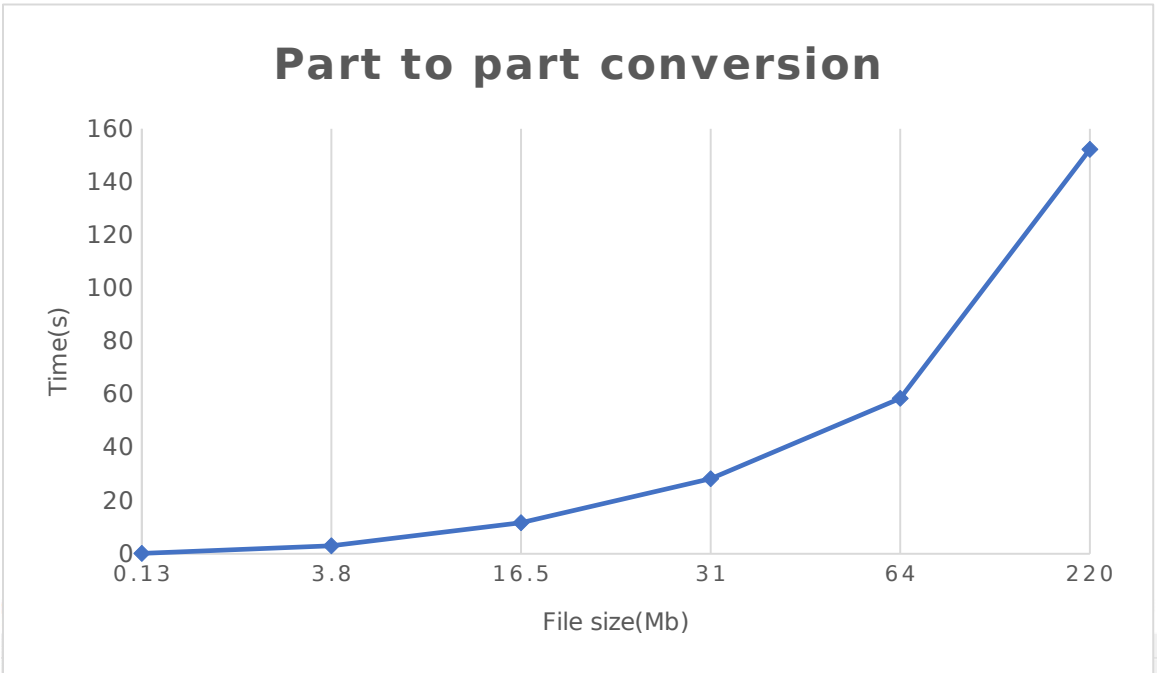
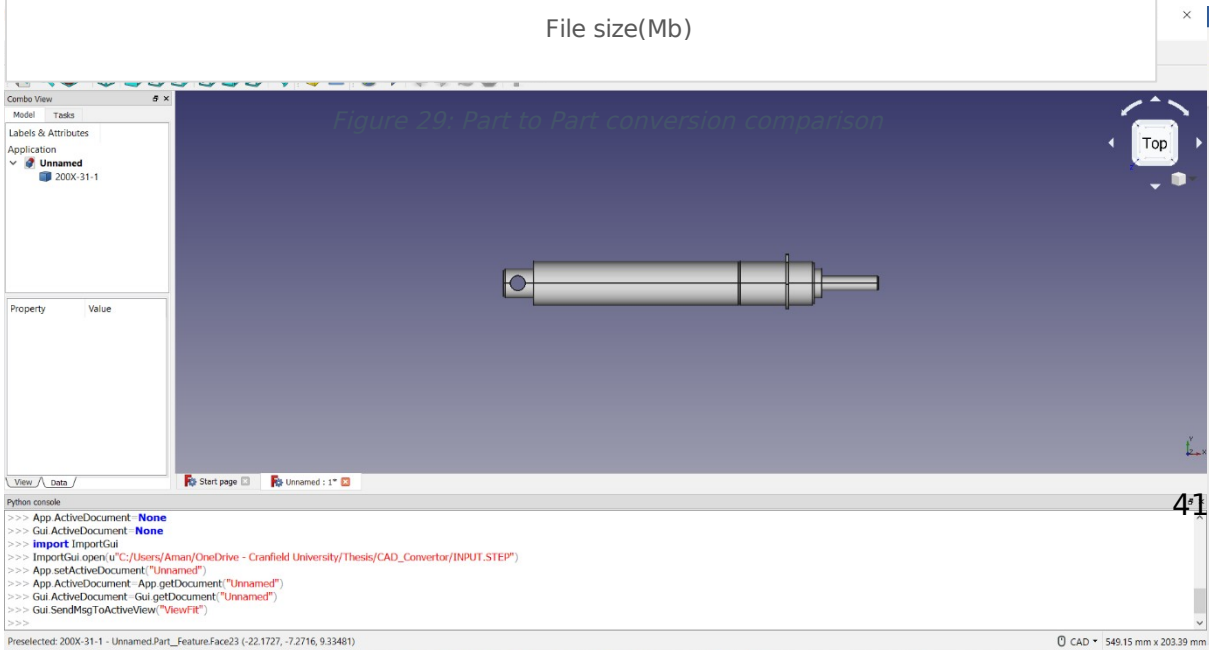


Figure 29: Part to Part conversion comparison



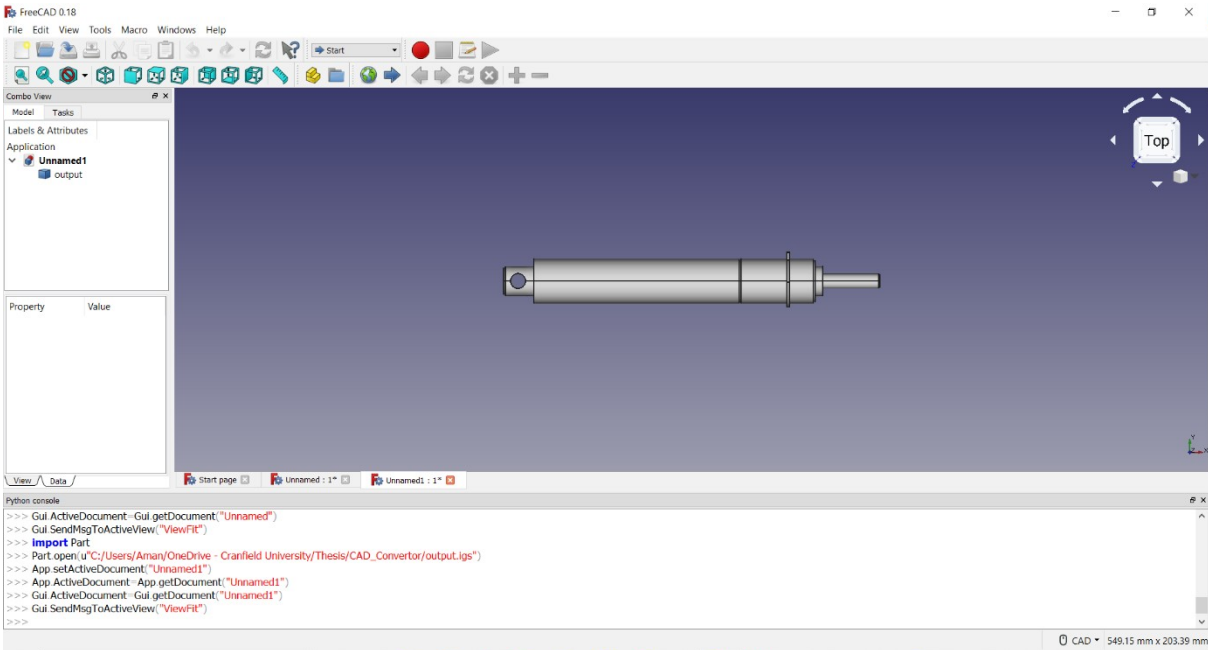


Figure 31: igs file after being converted from a stp file

6.2. Part to Mesh conversion

Converting part to mesh formats, the graph below shows how the amount of time required to convert the files varied depending on the size of the model.

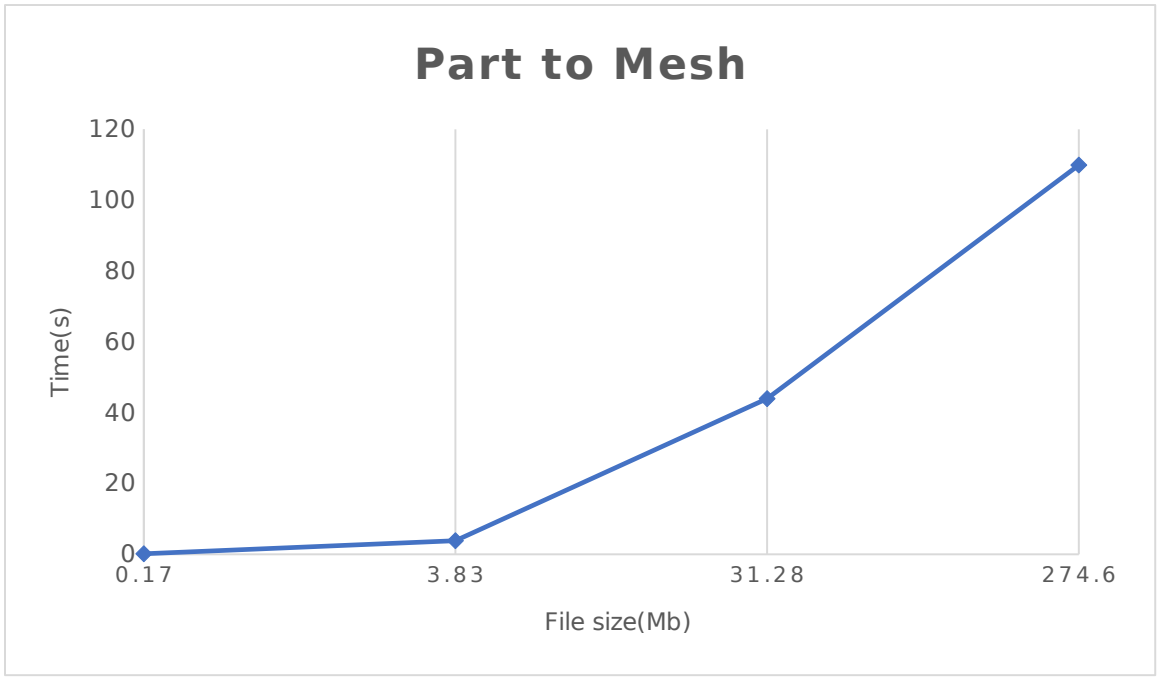


Figure 32: Part files to Mesh files

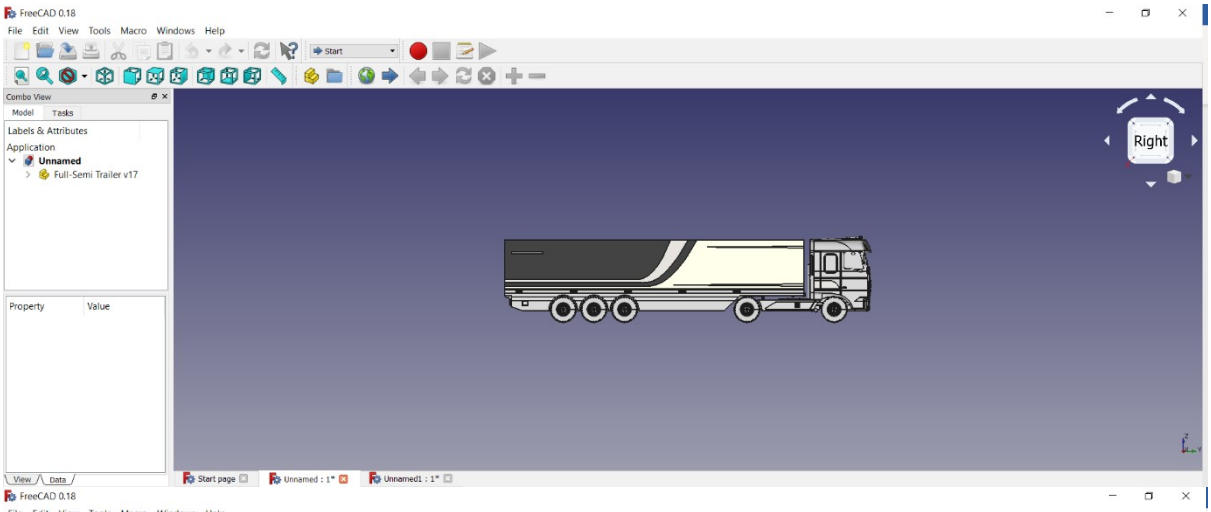
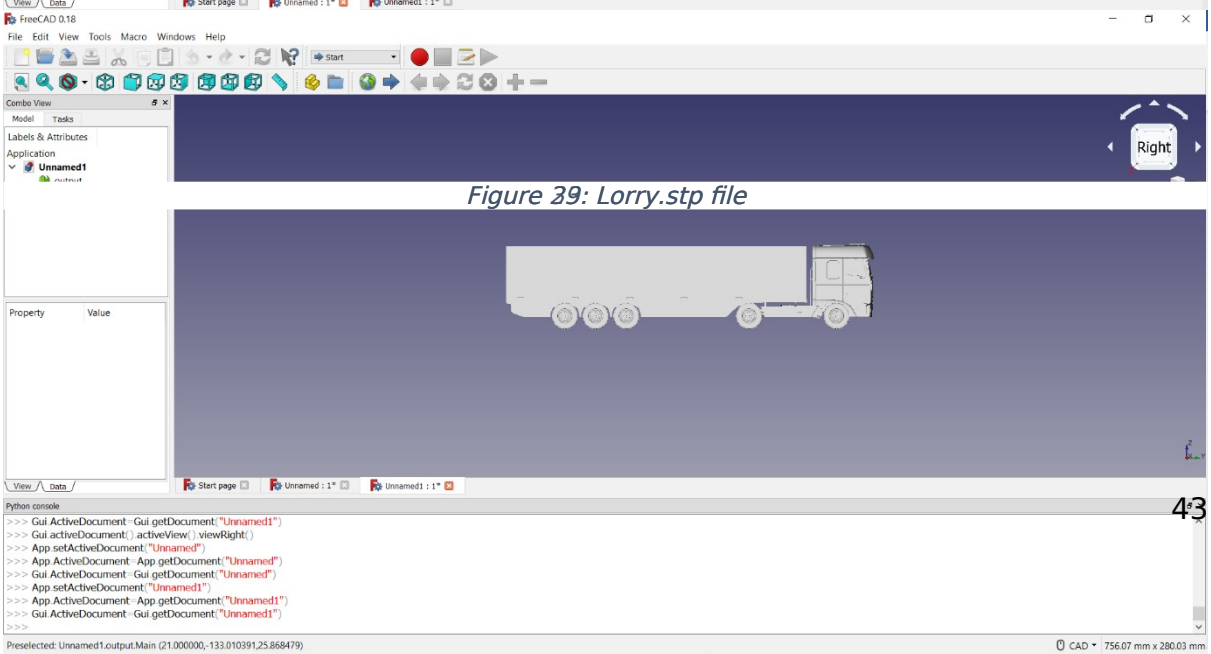


Figure 39: Lorry.stp file



As mentioned above, all the same models have been used for conversion reasons and the performance is provided in the graphs above. When converting part files to mesh files, after a certain sized file, the wait gets too long. For instance, when trying to convert a 64mb file the program was left to run for over 15 minutes and it still had not completed the conversion. Therefore, to convert large part files into mesh files, it is essential that the part models are broken down to smaller models, and then converted to mesh files.

6.3. Mesh to Part conversion

Converting mesh to part formats, the graph below shows how the amount of time required to convert the files varied depending on the size of the model.

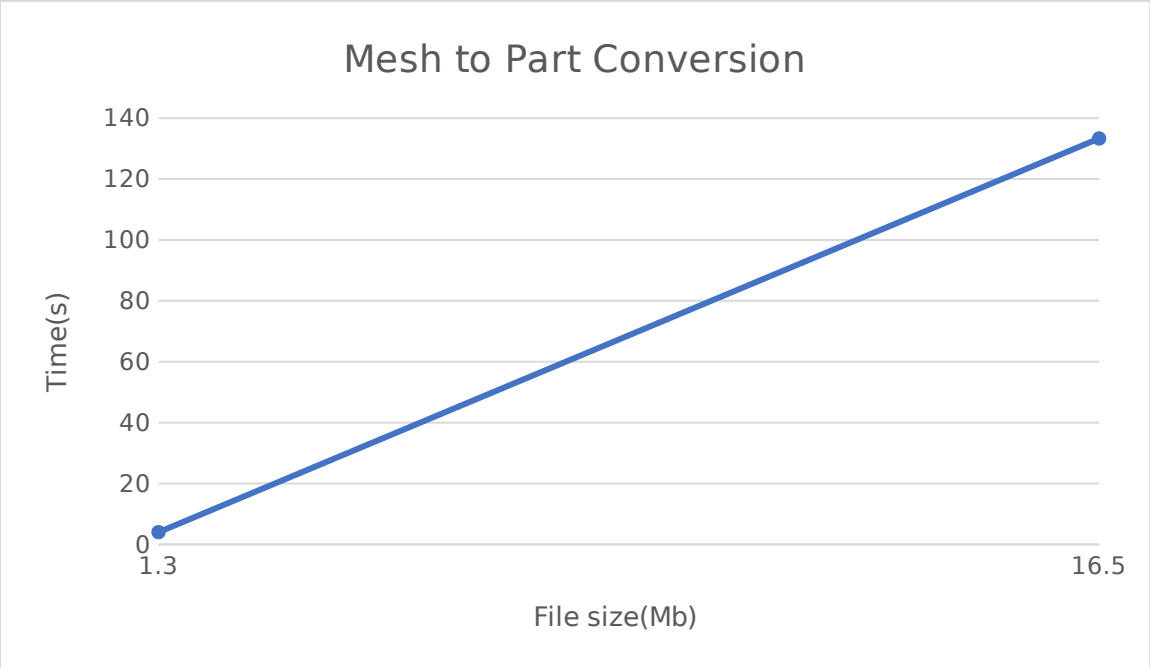


Figure 35: Mesh to Part conversion

In most cases, the requirement would be converting part files into mesh files. There is hardly ever a need to convert a mesh file into a part file unless the user has been provided a mesh file without the part file. Testing was carried out to

see if it is possible to convert mesh files into part files, the graph above shows that two mesh files were converted, one of them was 1.3 Mb and the second file being 16.5 Mb. As can be seen, the amount of time to convert the 16.5Mb file required a very long time, a 64 Mb was also tested, however the program was run for over 30 minutes and then cancelled as that was very long time to wait for a file conversion. Furthermore, converting from mesh files to part files did not provide very useful end results, as the part did not work properly, and the size of the part file converted from the mesh file was very large.

As mesh models are made up of many triangles, all the triangles must be individually read by the program before it can be converted into a part file, and this process is computationally very heavy. That is one of the reasons the graph above shows how the time increased exponentially and even a file size of just 64mb would take longer 30 mins to convert.

6.4. FreeCad and Salome

FreeCAD and Salome are very similar in terms of functionality that is available on both platforms, however Salome has a very steep learning curve, the documentation is harder to get hold of when compared to FreeCAD and even the GUI usage is more complicated in Salome compared to that of FreeCAD. One of the examples of ease of use when comparing FreeCAD and Salome is shown below.

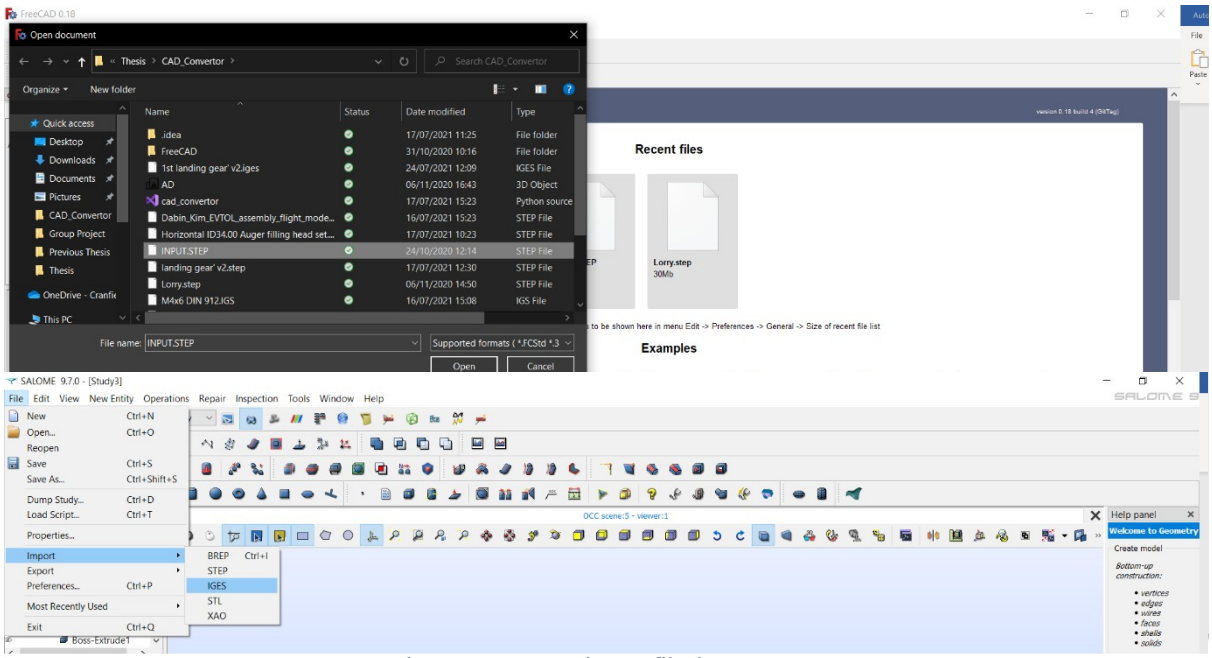


Figure 30: Opening a file in FreeCAD

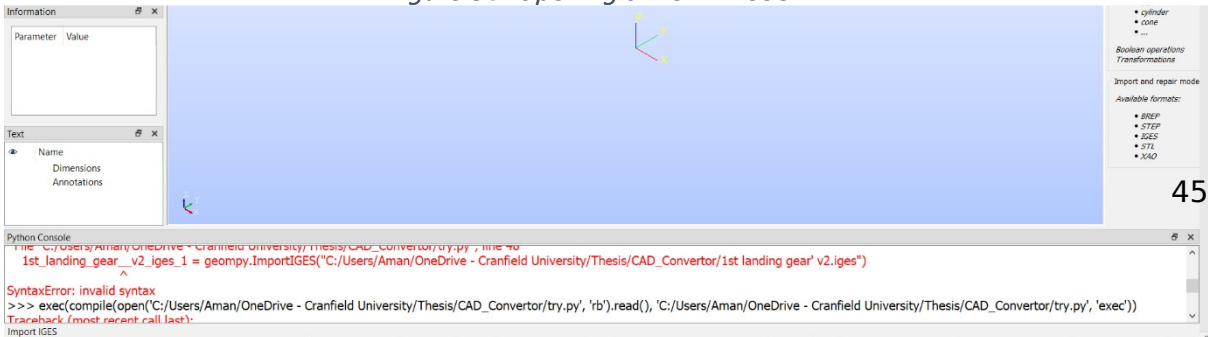


Figure 36 shows the process to open a file in FreeCAD, the process works like any commercial CAD software, simply clicking on “Open” and navigating to the file that the user wants to open, and all the supported CAD file formats could be opened by following this simple process. However, as can be seen in figure 37, the file opening process in Salome is much more complicated. In order to open a file, it is not as easy as clicking on “Open”, it requires that the file be imported and not all the supported file formats can be seen when opening a file, for instance, the user has to choose which file format the user wants to open and then navigate to where that file is saved, and only the format chosen for importing is visible and any other CAD files are not visible. This is a very complicated process to be able to simply open a CAD file.

Furthermore, the documentation for FreeCAD and code repositories are very easily available on Github, however Salome requires digging deep to be able to get to the source file, as can be seen below.

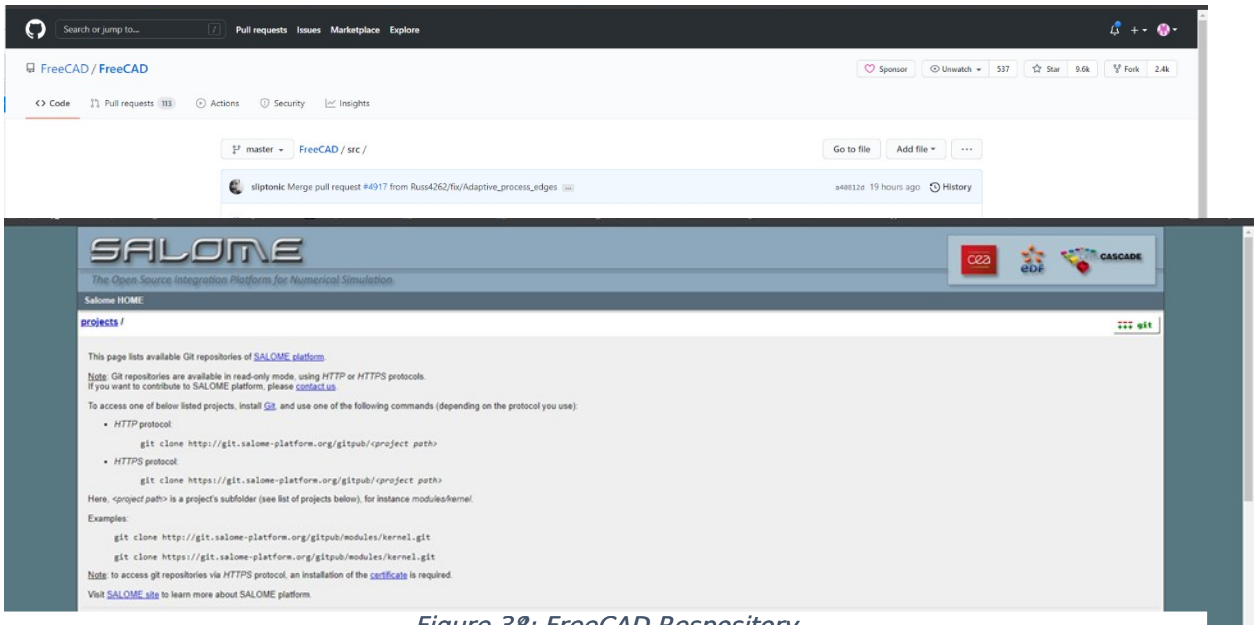


Figure 38: FreeCAD Respository

Figure 39: Salome Repository

Figure 38 shows FreeCAD repository that is easily available on Github and navigating the page easily provides information such as the latest version of FreeCAD, when it was released as well as info on many older versions. The Salome repository is only available on its own website, and it is much more difficult to navigate. For example, all the source code is not saved in a simple source code file, all the modules must be searched individually, and versioning information is not so easily available.

Furthermore, the FreeCAD documentation is up to date, using the information provided in the documentation for Python scripting, all the information provided was correct, up to date and worked without any issues. However, using the documentation provided on the Salome's website is out of date, using some of functions throw up an error stating that the syntax provided is out of date for the latest version. The documentation available on Salome's website is only up to date for version 7.8.0 whereas the latest available version is 9.7.0.

As can be seen above, FreeCAD is easier to use, the source code is easily available along with versioning information and the documentation being updated more often. These are just some of the reasons why FreeCAD has been used for the purposes of this research rather than Salome.

7. Future Work

Following on from the work carried out during this thesis, some further work to make improvements is possible.

7.1. DXF file format

The DXF file format is proprietary CAD file format owned by AutoCAD. As of FreeCAD 0.18 (version used for this project), DXF file format can only be imported into FreeCAD. However, from freeCAD version 0.19, DXF file format can be imported and exported. Therefore, the latest version of FreeCAD should be

tested and the Python script should be updated to be able to add the capability of being able to import and export the DXF file format.

It should be noted that the Python documentation for FreeCAD 0.19 does not currently provide the Python module information on how to convert DXF files. The conversion in FreeCAD 0.19 is currently provided via the GUI, however once the documentation has been updated with the module information on how to convert DXF files, this should be implemented in the Python program written for the purposes of this project and tested using FreeCAD 0.19.

7.2. Graphical User Interface

A graphical user interface can be designed for the software that has been written for this project, it would make the usage of the software easier, rather than using the command line the user can use the GUI to upload and convert the shapes. Furthermore, the GUI can be added to the DWT website providing a smoother experience for the user.

8. Conclusion

To conclude the CAD conversion report, the thesis required the work that was carried out by previous students to be reviewed and build upon the work that has already been carried out on the Digital Wind Tunnel project. The work on this thesis is a follow on from the work carried out by Hinatea Teriierooiterai, specifically the future work section that has been described in her thesis. The motivation for this thesis was to improve or automate the process that Hinatea had to follow to convert part files to mesh files and this thesis has met that requirement.

A thorough research has been carried out to understand the workings of various open-source CAD software and how the Open Cascade is the kernel that is used by open-source as well as commercial CAD software. Furthermore, a review of various part CAD file formats have been reviewed as well the .stl mesh file format. A review of open-source and commercial CAD software with a breakdown of their workings is discussed and based on this review, FreeCAD open-source software has been used for this thesis and CAD conversion.

A Python module has been written that can take a CAD part file and convert it into a mesh format or another part format. FreeCAD provides the capability of Python macros directly in its GUI and this is where the initial Python script was written and tested. FreeCAD is a very well maintained and regularly updated open-source software, making it ideal for the purposes of this project.

Various tests have been carried out on the CAD conversion software to ensure it meets the requirements, such as performance testing, integration testing, acceptance testing and system testing. Furthermore, testing was also carried out to check the software point of failure or at what point the amount of time required to convert a part becomes inadequate.

A discussion has been carried out provide a reason as to why FreeCAD has been used instead of Salome, with a comparison between their maintainability, ease of use and documentation. Future work that can be carried out by future students on the DWT project that can be followed on from this thesis has been provided.

Therefore, this thesis has met what it was set out to do, i.e. write a software that can convert CAD part file formats to other CAD part files as well as mesh files, that can be loaded on to the DWT website and can be used for simulation without the need of using the GUI of Salome or any other CAD software.

9 Bibliography

- *A Brief History of AutoCAD | Scan2CAD*. (n.d.). Retrieved May 8, 2021, from https://www.scan2cad.com/tips/autocad-brief-history/
- *A Brief History Of SolidWorks | Scan2CAD*. (n.d.). Retrieved May 8, 2021, from https://www.scan2cad.com/cad/solidworks-history/
- *Architecture — SALOME Platform*. (n.d.). Retrieved April 24, 2021, from https://www.salome-platform.org/user-section/about/architecture
- Bergeaud, V., & Tajchman, M. (2007). Application of the salome software architecture to nuclear reactor research. *Agent Directed Simulation Symposium, ADS 2007 - Proceedings of the 2007 Spring Simulation Multiconference, SpringSim 2007*, *1*, 383–387.
- Bertolotti, F., Macrì, D. M., & Tagliaventi, M. R. (2004). Social and organisational implications of CAD usage: A grounded theory in a fashion company. *New Technology, Work and Employment*, *19*(2), 110–127. https://doi.org/10.1111/j.0268-1072.2004.00131.x
- Bowcutt, K. G. (2003). *A PERSPECTIVE ON THE FUTURE OF AEROSPACE VEHICLE DESIGN*. https://doi.org/10.2514/6.2003-6957
- *CATIA, all you need to know about this CAD software - 3Dnatives*. (n.d.). Retrieved May 8, 2021, from https://www.3dnatives.com/en/catia-cad-310720204/
- *CATIA vs Inventor | CAD Software Compared | Scan2CAD*. (n.d.). Retrieved May 8, 2021, from https://www.scan2cad.com/cad/catia-vs-inventor/
- *CATIA vs SolidWorks | Software Comparison | Scan2CAD*. (n.d.). Retrieved May 8, 2021, from https://www.scan2cad.com/cad/catia-vs-solidworks/
- *Chapter 2: Navigating the SolidWorks Interface - Mastering SolidWorks*. (n.d.). Retrieved May 8, 2021, from https://learning.oreilly.com/library/view/mastering-solidworks/9781119300571/c02.xhtml
- *Chapter 5. Rapid Prototyping - Product Manufacturing and Cost Estimating using CAD/CAE*. (n.d.). Retrieved November 29, 2020, from https://learning.oreilly.com/library/view/product-manufacturing-and/9780124017450/xhtml/CHP005.html
- *Chapter Ten. Modeling for Manufacture and Assembly - Modern Graphics Communication, Fifth Edition*. (n.d.). Retrieved November 8, 2020, from https://learning.oreilly.com/library/view/modern-graphics-communication/9780134852751/ch10.xhtml
- di Donato, D., & Abita, M. (2019). Low-cost 4D BIM modelling a comparison between Freecad and commercial software. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences -*

ISPRS Archives, *42*(2/W17), 107–114. https://doi.org/10.5194/isprs-archives-XLII-2-W17-107-2019

- *FEM Module - FreeCAD Documentation*. (n.d.). Retrieved November 19, 2020, from https://wiki.freecadweb.org/FEM_Module
- *FreeCAD [How-to]*. (n.d.). Retrieved May 8, 2021, from https://learning.oreilly.com/library/view/freecad-how-to/9781849518864/
- *FreeCAD_Mod_Dev_Guide/FreeCAD_Mod_Dev_Guide__20190912.pdf at master · qingfengxia/FreeCAD_Mod_Dev_Guide · GitHub*. (n.d.). Retrieved January 23, 2021, from https://github.com/qingfengxia/FreeCAD_Mod_Dev_Guide/blob/master/pdf/FreeCAD_Mod_Dev_Guide__20190912.pdf
- *Getting Started With Autodesk Inventor (Basics and User Interface): 5 Steps - Instructables*. (n.d.). Retrieved May 8, 2021, from https://www.instructables.com/Getting-Started-With-Autodesk-Inventor-Basics-and-/
- *Gmsh 4.7.1*. (n.d.). Retrieved November 29, 2020, from http://gmsh.info//doc/texinfo/gmsh.html#Overview
- *How CAD Has Evolved Since 1982 — Past, Present & Future | Scan2CAD*. (n.d.). Retrieved May 8, 2021, from https://www.scan2cad.com/cad/cad-evolved-since-1982/
- Hu, J. (2019). *Macro Micro - Coupling Simulation Front - End Processing Tools Based on SALOME Platform*. *181*(Ice2me), 35–41. https://doi.org/10.2991/ice2me-19.2019.9
- Irfan, U., Aslam, K., & Rizwan, N. (2015). A Review On Cad Cam In Dentistry. *Journal of the Pakistan Dental Association*, *24*(3), 112–116. http://archive.jpda.com.pk/volume-24-issue-3/a-review-on-cad-cam-in-dentistry/
- Jameson, A. (n.d.). *FEDSM2003-45812 THE ROLE OF CFD IN PRELIMINARY AEROSPACE DESIGN*.
- Jelen, A. (2016). *Digital Wind Tunnel – Web GUI for OpenFOAM using Model – View – Controller architectural pattern*. *August*, 119.
- Kutty, H. A., & Rajendran, P. (2017). 3D CFD Simulation and Experimental Validation of Small APC Slow Flyer Propeller Blade. *Aerospace 2017, Vol. 4, Page 10*, *4*(1), 10. https://doi.org/10.3390/AEROSPACE4010010
- Mangani, L., Romanelli, G., Gadda, A., & Casartelli, E. (2021). *Comparison Of Acceleration Techniques on CFD Open-Source Software for Aerospace Applications*. 22–26. https://doi.org/10.2514/6.2015-3059
- Neal, K. D., Groth, C., & Shannon, T. (2018). CAD/CAM software for three-dimensional printing. *Journal of Clinical Orthodontics : JCO*, *52*(1), 22–27.
- *Open CASCADE Technology - Open Cascade*. (n.d.). Retrieved January 23, 2021, from https://www.opencascade.com/open-cascade-technology/
- *Open CASCADE Technology: Introduction*. (n.d.). Retrieved January 23, 2021, from https://dev.opencascade.org/doc/overview/html/index.html#OCCT_OVW_SECTION_3
- Paris, I., & Handley, D. (2004). CAD usage and knowledge base technology in shoe design and development. *International Journal of Computer Integrated Manufacturing*, *17*(7), 595–600. https://doi.org/10.1080/0951192042000273159

- *SALOME Geometry User's Guide: Import/Export*. (n.d.). Retrieved April 24, 2021, from https://docs.salome-platform.org/7/gui/GEOM/tui_import_export_page.html
- *stl format - Convert STEP file type to STL - Stack Overflow*. (n.d.). Retrieved January 23, 2021, from https://stackoverflow.com/questions/55714806/convert-step-file-type-to-stl
- *The use and implementation of CAD in the Swedish furniture industry - ProQuest*. (n.d.). Retrieved May 3, 2021, from https://search.proquest.com/docview/214633579/F560C3E3EA474D59PQ/1?accountid=10297
- *Topological data scripting - FreeCAD Documentation*. (n.d.). Retrieved January 23, 2021, from https://wiki.freecadweb.org/Topological_data_scripting
- Tucker, P. G., & Liu, Y. (2005). Contrasting CFD for electronic systems modeling with that for aerospace. *Proceedings of the 6th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Micro-Electronics and Micro-Systems - EuroSimE 2005*, *2005*, 618–625. https://doi.org/10.1109/ESIME.2005.1502877
- *Understanding the user interface - Practical Autodesk AutoCAD 2021 and AutoCAD LT 2021*. (n.d.). Retrieved May 8, 2021, from https://learning.oreilly.com/library/view/practical-autodesk-autocad/9781789809152/5fadba45-c3ea-45cc-9597-8de6d998bce9.xhtml
- Venkatakrishnan, V. (2016). *On the role and challenges of CFD in the aerospace industry Spalart Boeing Commercial Airplanes Seattle USA*. *120*, 1223–209. https://doi.org/10.1017/aer.2015.10
- Wildt, D. (2018). *Sensitivity analysis methodology for OpenFOAM Computational Fluid Dynamics*. *August*, 2017–2018.
- Williamson, M. (n.d.). *The Impact of CAD on Aerospace Design*.
- Zhang, K., Hugo, V., Espinoza, S., & Stieglitz, R. (2019). *Implementation of the System Thermal-Hydraulic code TRACE into SALOME Platform for Multi-Scale Coupling*. *July*.
- Zhou, M. Y. (2005). STEP-based approach for direct slicing of CAD models for layered manufacturing. *International Journal of Production Research*, *43*(15), 3273–3285. https://doi.org/10.1080/00207540500097809

Appendix

CAD Conversion Module

```
# CAD conversion script - written in Python 3.6.

# FreeCad version 0.18 has been used.
```

```
FREECAD_PATH = 'C:\\Program Files\\FreeCAD 0.18\\bin'  # path to the
FreeCAD.pyd file


import sys

sys.path.append(FREECAD_PATH)   # This is necessary in order for python to find
FREECAD

import os

import time

import FreeCAD

import Part

import Mesh


mesh_formats = ['.stl']



class ConvertorClass:
    """Class to convert CAD file formats"""
    def __init__(self):
        try:
            self.in_file = sys.argv[1]      # input filename and format
            self.out_file = sys.argv[2]     # output filename and format
        except IndexError:
            raise SystemExit(self.publish_help_screen())


    def convertor(self):
        """
        Conversion method, all the CAD file conversion take place through this
method.
        """
        while len(sys.argv) == 3:
            try:
                # Splitting the path into a pair, root and ext - ext is the extension part
of the file,
```

```
            # the root part is everything else in the path
            in_filename, in_ext = os.path.splitext(self.in_file)
            out_filename, out_ext = os.path.splitext(self.out_file)
            print(f"{in_ext} -> {out_ext}")

            # Calling the shape function from Part
            shape = Part.Shape()
            # Checking if the input file is in the mesh_formats list
            if in_ext in mesh_formats:
                print("Mesh file: ", self.in_file)
                Mesh.open(self.in_file)
                freecad_document =
FreeCAD.getDocument("Unnamed").findObjects()[0]
                # Checking if the requested output file is in the mesh_formats list
                if out_ext in mesh_formats:
                    tic = time.time()
                    print(f"Converting to a mesh file: {self.out_file}")
                    Mesh.export([freecad_document], self.out_file)
                    toc = time.time()
                    print(f"Converted the CAD model in {toc - tic:0.10f} seconds")
                    exit()
                else:
                    # Setting the meshing tolerance, this can be changed depending
on the requirements
                    # and convert to a part file from a mesh file format.
                    tic = time.time()
                    shape.makeShapeFromMesh(freecad_document.Mesh.Topology,
0.01)
                    self.part_convertor(shape, self.out_file, out_ext)
                    toc = time.time()
                    print(f"Converted the CAD model in {toc - tic:0.10f} seconds")
                    exit()
```

```
            # Checking if it is converting one part file to another i.e stp to igs for
example
            else:
                tic = time.time()
                print(f"Opening a part file: {self.in_file}")
                shape.read(self.in_file)
                self.part_convertor(shape, self.out_file, out_ext)
                toc = time.time()
                print(f"Converted the CAD model in {toc - tic:0.10f} seconds")
                exit()

        except IndexError:
            raise SystemExit(self.publish_help_screen())
    else:
        self.publish_help_screen()
        exit()

@staticmethod
def part_convertor(shape, out_file, out_ext):
    """Method to convert CAD part formats."""
    if out_ext == '.stp':
        print(f"Exporting to part file: {out_file}")
        shape.exportStep(out_file)
    elif out_ext == '.igs':
        print(f"Exporting to part file: {out_file}")
        shape.exportIges(out_file)
    elif out_ext == '.brp':
        print(f"Exporting to part file: {out_file}")
        shape.exportBrep(out_file)
    elif out_ext == '.stl':
        print(f"Exporting to a mesh file: {out_file}")
        shape.exportStl(out_file, 0.01)     # The second parameter is surface
deviation
```

```
        else:
            print(f"Exporting to {out_ext} is not supported.")

    @staticmethod
    def publish_help_screen():
        """Help screen to show the usage of the program."""
        print("Usage:\n")
        print("<convertor.py> <input filename.ext> <output filename.ext>")
        print("\n")
        print("Example: main.py input.stp output.igs\n")



# This imports the script without running it.
if __name__ == '__main__':
    run_convertor = ConvertorClass()
    run_convertor.convertor()
```