Escola Universitària d'Enginyeria Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Volume I
Report – Budget

TREBALL DE FI DE GRAU

"CREATION OF INTERACTIVE 3D DOCUMENTS TO SUPPORT THE SETUP PROCESS OF MACHINE TOOLS"

TFG presentat per obtenir el títol de GRAU en ENGINYERIA MECÀNICA
Per **Víctor Arnedo Blanco**

Barcelona, 11 d'octubre de 2016

Director: José Antonio Travieso Rodríguez
Departament de Enginyeria Mecànica (DEM)
Universitat Politècnica de Catalunya (UPC)

TABLE OF CONTENT

LIST OF FIGURES

RESUM

El present projecte consisteix en la recerca i implementació de solucions per a la millora de la creació de documents 3D interactius de suport a la enginyeria de fabricació. La idea és facilitar el flux de treball entre el dissenyador i l’operari de màquines creant plataformes en les quals es puguin veure clarament les parts dels assemblatges, les seves característiques i realitzar possibles feedbacks per tal d’optimitzar l’elaboració de les peces. A més, es pretén millorar la qualitat dels documents creats fins aleshores, implantant nous tipus d’arxius i millores gràfiques amb les que serà molt més difícil la pèrdua d’informació durant el procés.

Es compararan les solucions trobades tant tècnica com econòmicament, realitzant exemples de cadascuna de les formes d’obtenir els documents, i finalment es proposarà la més adient.

RESUMEN

El presente proyecto consiste en la búsqueda e implementación de soluciones para la mejora de la creación de documentos 3D interactivos de apoyo en la ingeniería de fabricación. La idea es facilitar el flujo de trabajo entre el diseñador y el operario de máquinas creando plataformas en las cuales se puedan ver claramente las partes del ensamblaje, sus características i realizar posibles feedbacks por tal de optimizar la elaboración de las piezas. Además, se pretende mejorar la calidad de los documentos creados hasta ahora, implantando nuevos tipos de archivos y mejoras gráficas con las que será mucho más difícil la pérdida de información durante el proceso.

Se compararán las soluciones encontradas tanto técnica como económicamente, realizando ejemplos de cada una de obtener los documentos, y finalmente se propondrá la más adecuada.

ABSTRACT

The current thesis consists on the research and implementation of solutions for the improvement in the creation of 3D interactive documents for the support in manufacturing engineering. The idea is ease the workflow between the designer and the operator creating new platforms in which it is possible to see clearly the assembly components, their features and carry possible feedbacks on so the elaboration of pieces can be optimized. Furthermore, it is pretended to increase the documents quality done until now, implementing new file types and graphic improvements with which the loose of information during the process will be more difficult.

The founded solutions will be compared technical and economically, making examples of each of the ways of obtaining the documents, and finally the most suitable one will be proposed.

ACKNOWLEDGMENT

I would not like to finish this thesis without thank and dedicate this work to some people first.

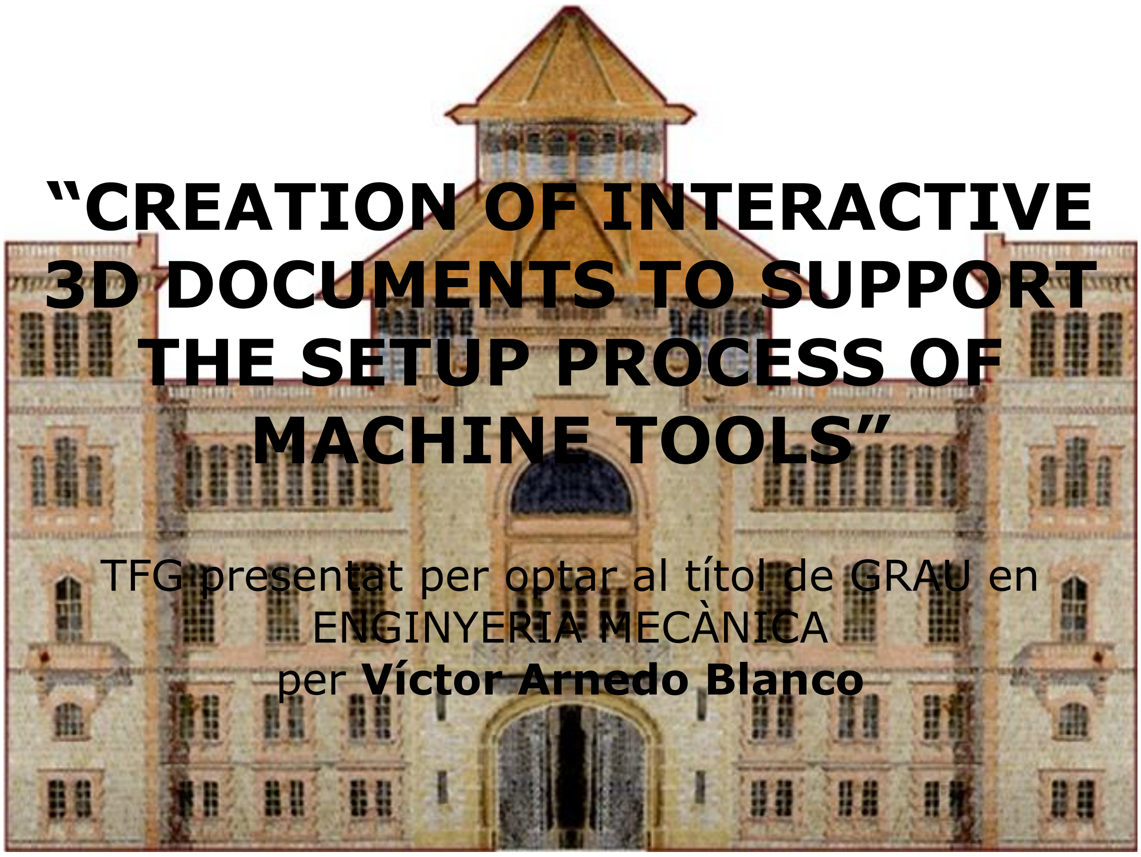
First of all, I would like to thank to Dr. Prof.-Ing Andreas Ellermeier for giving me the possibility of participating in such a project as AutoRüst in their laboratory and be a part of the team for a few months in OTH Regensburg.

Secondly, I want to show my gratitude to Scientific Assistant Herr. Daniel Vögele for the weekly meetings and help that have make this thesis having the direction to reach the goals defined.

Finally, this thesis would not have been possible without the support of my family and girlfriend. I would like to take the opportunity here to express all my gratitude to all their effort, advices and comprehension during all my stay in Regensburg. To Jennifer, thanks to hold me even being at thousands of kilometres from you. It has been a very difficult issue and you deserve it all. To Miquel and specially my mother, all my appreciation. It is always difficult to be separated from your sons, but your help, advices, strength given and the convincement to come here and enjoy this unforgettable experience has been priceless. To my father, it will have been impossible without your calls, advices and support all these months along. Thank you to all my family, brothers, sister, nephew… you are always there and it is so valuable to me. And of course, to my friends here in the Erasmus Exchange Program, my family for five months. To all of them thank you for making this experience something worth to live.

Escola Universitària d'Enginyeria Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Report



"CREATION OF INTERACTIVE 3D DOCUMENTS TO SUPPORT THE SETUP PROCESS OF MACHINE TOOLS"

TFG presentat per optar al títol de GRAU en ENGINYERIA MECÀNICA
per **Víctor Arnedo Blanco**

Barcelona, 11 d'octubre de 2016

Director: José Antonio Travieso Rodríguez
Departament d'Enginyeria Mecànica (DEM)
Universitat Politècnica de Catalunya (UPC)

CHAPTER 1: INTRODUCTION

Research and implementation of solutions for the automatic creation of interactive 3D documents to support the setup process of tool machines is a thesis work included in the AutoRüst project for the standardization and automatic solution in CAD/CAM production processes strings.

The aim of this research is to establish an efficient workflow in the production of pieces and assemblies between the designer, who works with the specialized software, and the operator, manipulating the manufacturing machines. This work is done by the creation of a series of interactive documents or application with which the operator will be able to interactive with the 3D model embedded, its information and carry some annotations or feedback out.

The reader will find inside the thesis theoretical information in the first part for a better comprehension of all the concepts explained in the second part. The latest could be described as a practical guide of how to implement the different chosen solutions for the creation of default templates and addition of the assemblies and the related data, with all the instructions and images in order not to be complicated to follow and understand.

CHAPTER 2: MARKET ANALYSIS

In this section we will establish the basis of the thesis. We will carry out a market analysis, a fundamental task before a project, necessary in order to understand in which conditions are we able to work, how we can create our solution and which tools do we have available.

First of all, it is good to know which formats are we going to use in the process, their history and usage, the capabilities and why do we use them. It is important, furthermore, to be aware of the software involved. Some of them can be commercial and, usually, difficult to afford because their high price. Others, however, are free and open source but usually have less quality or features than the first ones. It is precise to find a balance between quality and price in this case. Finally, we will talk about the possible solutions we can implement to reach the goal of our thesis and we will choose the most suitable one, according to our considerations.

2.1. Applicable standard formats for setup-documents

2.1.1. HTML

HTML is the abbreviation of HyperText Markup Language and it is a standard markup language used to create electronics documents on the World Wide Web called web sites and applications as well. The code ensures the proper formatting of text and images, so the internet browser, as Chrome or Mozilla Firefox, is able to display them. It describes the structure of a website semantically and includes cues for the appearance of the web page, and that is why is called a markup language, rather than a programming language. HTML allows images and other objects to be embedded and it is possible to create interactive forms, as 3D models, inside. [1]

A markup language works with typesetting instructions, such as those in LaTeX for example, and these instructions are encapsulated by tags. Tags are also used to specify hypertext links, that allows Web developers to direct users to other Web pages with only a click of the mouse on either an image or words. [2]

The last HTML update is HTML 5. It uses the same basic rules as its predecessor, but adds some new tags and attributes activated by using JavaScript [3]. If we want to work with HTML, we will need to use a text editor as long as it is saved with a .html file extension, but it is easier to use a HTML editor. Once the file is created, we can view it online in a browser after been uploaded in a web server or just locally in the editor.

In the last fifteen years, there have been a huge number of systems and proposals of how to embed 3D objects in the Web, but most of them have disappeared over time. Today we have a sort of systems that are based in Plugin technology. A few example of them, could be Adobe Flash, Silverlight or Java3D, but they have two main handicaps. First, they are not installed by default in most of the systems, so the user has to deal with the installation, security and possible incompatibilities that happen with the browser or operative system. In addition, developers who want to work along with DOM content, building up Web applications or pages that use both, have to solve the synchronization incapacities that can occur.

There are, as well, a small number of systems that do not use plugins and try to integrate the rendering system directly into the browser architecture. Due to the fact that the use of HTML is a good solution for reaching our goal, we will work with one of this systems, called X3DOM. [4]

X3DOM is an open source JavaScript framework[1] used to create declarative 3D scenes in Web pages. It is based on standard browser technology so it does not need any plugin to display the scenes. Declarative 3D is a concept that means that you can create and display an interactive 3D scene using a structured representation rather than writing a code, and, in the case of X3DOM, this representation is a part of HTML document that represents a Web page, so the 3D content acquires the same importance than text, links or images.

[1] In computer systems, a framework is often a layered structure indicating what kind of programs can or should be built and how they would interrelate [5]

The name of X3DOM corresponds to X3D, the royalty-free ISO standard for declarative 3D graphics that will be explained later and DOM, an abbreviation of Document Object Model. DOM describes the interaction and hierarchical representations associated with the content of HTML documents. The X3DOM elements can be manipulated with DOM operations, like other HTML elements. We can change attributes of a 3D model with a JavaScript just as we could change the text of a Web page [6].

In brief, we will use X3DOM for three main reasons. The first one is that there is no need to use a plugin to display the scenes, so the time for the setting up will be reduced. The second corresponds to the fact that the most of the parts of X3DOM are standard, so it is easy to exchange content between users and developers, and, furthermore, there are great facilities on learning the language. This is also related to the third reason. If you know how to build simple Web pages, there is the possibility to exploit and extend your knowledge on HTML and DOM, instead of learning a new API.

2.1.2. PDF

PDF, abbreviation of Portable Document Format, is a storage file format for digital documents, independent from software or hardware [7]. It is a complex format, that encapsulates a complete description with images, graphics, text and other information.

PDF started off in the earlies 90s on the purpose of creating a paperless office as a project of John Warnock, one of the founders of Adobe. The file format was created as an internal project, in order that documents could be spread throughout the company and displayed on any operating system, but it was not until 1993 that Adobe Systems made the PDF specification available for free. [8]

The first versions of the PDF documents were not very popular, due to the large size that they supposed, bigger than most of the documents similar to them. However, with the introduction of the broadband for commercial users, the publicity the company made and the easy way to use in every platform and operating system, this type of file became widespread worldwide.

Nowadays, this format of file has become an open standard and published by the International Organization of Standardization (ISO) from 2008, but Adobe Systems has the patent, granting the royalty-free rights necessary for use, sell and distribute implementations. [9]

PDF files combine different technologies, for embedding, replacing and allowing fonts to travel with documents or for storing and compressing data and elements when necessary, for example, but it is strongly based on PostScript technology [10]. PostScript is a page description language, or PDL, used in a usual way in printing studios as a transporting format. The concept was new because the utilization of a complete programming language for describing an image that, later, would be printed by a high quality printer, and the notation of the programming language was different from what it had been used, declaring the parameters of a command before the command itself. This is called the Reverse Polish Notation, or RPN. PDF, considered another page description language, is derivate from it but simplified.

There are many software, both free and commercial, capable of reading a PDF document on Internet. For instance, PDF Reader, Foxit Reader or Bluebeam, but the most common used worldwide is Adobe Reader or Acrobat. Precisely the last one is the software needed for the purpose of the project, embedding 3D models and all its information for an interactive use, due to the huge amount of features it has and its efficiency.

Adobe Reader and Acrobat have the possibility of interacting with 3D models, provided that it is a U3D OR PRC file. These are formats that will be explained later. While Adobe Reader only has the possibility of opening the document and interact with it, Acrobat Pro had the option of embedding the 3D model. But this is over now, the new updates will not be focused anymore in that technology, the responsible of this issues are now Tetra4D with Enrich and Converter software and Techno Soft 3D.

So, in summary, we will investigate how to deal with the problematic with PDF technology due to the versatility it has, the huge sort of possibilities we have and different documents and templates we can create and the ease of sharing documents independently from the software and operating system used.

2.2. Underlying 3d-formats

2.2.1. U3D

U3D, or Universal 3D, is a data structure and file format used in 3D PDF documents [12]. This format is an ECMA[2] standard with the aim of simplify the transformation of complex 3D data into a format that can be streamed, compressed and viewed on affordable, free software and hardware platforms. U3D encoding is used in a wide range of application areas, including engineering, manufacturing, medical or art [13]. The format is totally supported by PDF, so you can access to the 3D content by Adobe reader, for example. The goal is to have a universal standard for all kind of 3D data and develop an open source library for easing the use of the format.

[2] European Computer Manufacturers Association (ECMA) is an industry association founded in 1961 and dedicated to the standardization of Information and Communication Technology and Consumer Electronics [11]

U3D is a relatively new format. It was established in 2003 by Industry Forum, a group composed by a lot of industries from different sectors, like Boeing, Microsoft or Lego. These companies were supposed to base and extend the universal 3D format from the already defined X3D file format, but they broke free from this thinking and worked with a tool created a few years before, the Intel IFX v2.0 toolkit and its 3D file format, the W3D. The product of this co-working was the U3D file format. A year later, on 2004, it was approved by the ECMA so, as said before, it became a standard file format. [14]

U3D philosophy has two main points - the specification is open and the reference code is available as Open Source. There are a number of tools and libraries available for the creation of this kind of files, most of them commercial software, but there are also free options like U3D. Using these tools requires training and it is difficult to find one of them that does work a different format from a mesh. This is a problem because the use of a mesh implies that, if the designer is working with an assembly, he will lose a lot of properties, like color, and the different parts, becoming all together a unique solid. In case we want to work with a STEP file, like lots of CAD and CAM software do, we have problems when converting the files into U3D format and we have to use a mesh format like STL.

2.2.2. X3D

X3D is a royalty-free open file format, defined as an ISO standard and used for representing 3D computer graphics [15]. It is an extensible standard that can be supported easily by creation tools, browsers and other 3D applications, both for importing and exporting. It replaces VRML (Virtual Reality Modeling Language), although it provides compatibility with all the VRML content [16]. X3D extends its predecessor with the design and the possibility to employ XML for real time complete scenes modeling. It can work with other open source standards, like DOM and XPath.

During the World Wide Web boom there were some intends to integrate virtual reality in web browsers, but they all failed until VRML was developed. Its first version appeared in the mid-90s, but the second version was the famous one in the educational and investigation field. Because of the slowness of the internet, it was very difficult to view a low quality model in that time, and one of the most important companies behind VRML stopped the development of the format, so in the earlies 00s it was “forgotten”. However, a group of companies created the Web3D Consortium due to the necessity of continuing with the implementation of 3D content in the Web, so they created X3D format, that corrected VRML shortages, a few years later [17].

One of the problems of X3D is the necessity of installing plug-in technology in the browser for interpret it, and due to that, in 2009 a new specification was started to be developed, known as WebGL. It was thought of the way to introduce 3D content, permanently, in browsers. X3DOM, for example, is based in this technology nowadays.

X3D supports 3D and 2D graphics as well as animations, audio and video and CAD/CAM data that can be interactive with the user with mouse-based picking and dragging.

Summarizing, a good reason to use X3D is the extended compatibility with other formats and codifications, as VRML or XML that provide it integration with Web technologies. The possibility of creating Web applications and share them easily, since the final user only needs a browser for interacting with the content, and the huge amount of information spread out in the Internet will make this process highly optimized.

2.3. Software involved in the workflow

2.3.1. Tetra 4D

Tetra4D is a tool that allows the user creating interactive 3D PDF files easily. It is composed by two different products, 3D Converter and Enrich, both of them working in Acrobat as a plug-in.

Tetra4D was born in late 2013 as an end-user software, after Tech Soft 3D acquired it. Before that, it was Adobe who worked on 3D PDF development for several years and, since there was a huge work done by this company, Tetra4D’s aim has been refining their technology and extended it to more users worldwide [18].

One of the tools developed by this company is Tetra4D Converter, the successor of 3D PDF Converter. Its function is converting 3D CAD and CAM data into interactive 3D PDF documents with precise solid geometry and including Product Manufacturing Information. It is also used as a converter for formats as STEP or STL, and it works with software like Catia or SolidWorks [19].

Tetra4D Enrich is the other product the company offers us. This tool incorporates Tetra4D Converter functions and extend them, including all the required information for the product, like the 3D CAD, metadata or all the workflows related to the Model Based Enterprise, as 3D MBD Technical Data Packages with PMI. It is, furthermore, possible to create your own template or use existing examples in order to build interactive documents with work instructions, part catalogs or further information of the components. At the end we will have a document similar to an application, with which the operator will know all the necessary about the model [20].

All in all, the fact of being a very intuitive software, without need of any programming knowledge is what make consider Tetra4D a very powerful tool, suitable for our purposes of earning time and facilitate the creation of the 3D interactive PDF files.

Although all the features and benefits Tetra4D Enrich can provide us, it is difficult to afford due to its high commercial price. It depends on the company the decision to adopt this software as their main instrument for creating 3D PDF files.

2.3.2. 3D Tool

3D Tool is a software specialized in CAD files conversion with designing tools included. It includes a huge range of different formats that is capable of view and convert, always depending on the version acquired.

The tool has the possibility to publish 3D PDF itself as seen in figure 2.1, the goal we wanted to reach in this part of the thesis, but the document created only includes the 3D model, without the possibility of adding any useful information like a catalog of the parts or PMI information.

3D Tool has received many satisfactory feedbacks, due to the ease to use it, without any previous CAD knowledge for informatics, for example. The interface gives to the user facilities to find whatever he wants, like measure a part of the model, or convert from one format to another.

It is very useful for us since is one of the few software able to import STEP files efficiently and after that convert the 3D model into a U3D file. It does not suppose any difficulty to afford it, because it is a relatively low-priced tool, so it is perfect for companies or universities that need to produce U3D files from STEP and a lot of more formats like Catia’s, SolidWork’s or JT.

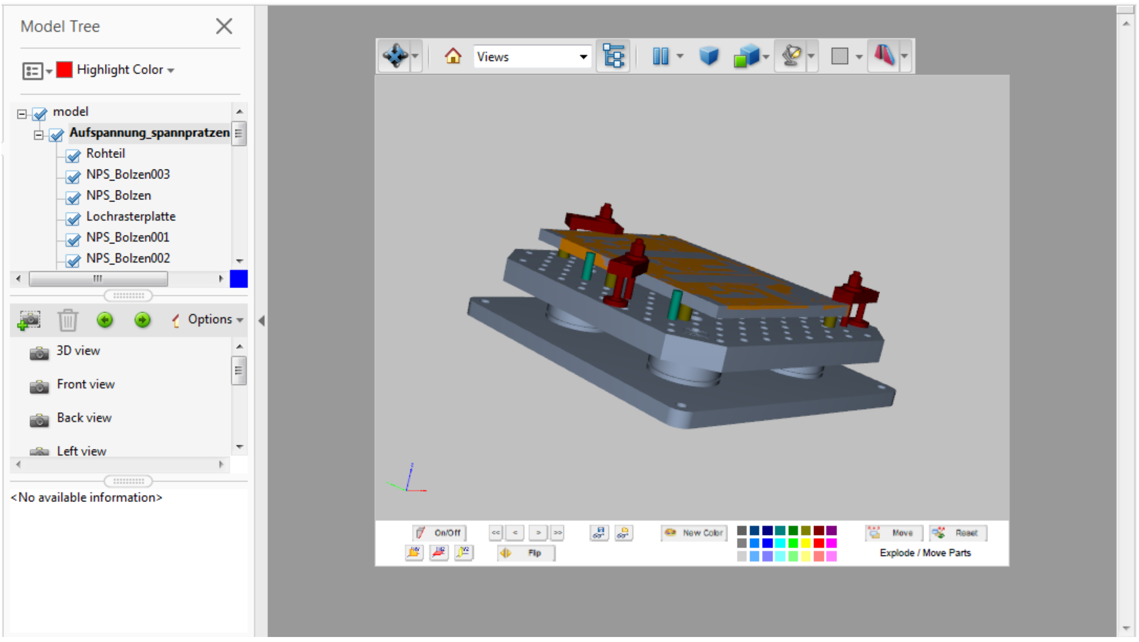


Figure 2.1. *3D PDF created by 3D Tool.*

2.3.3. LaTeX

LaTeX is a text preparation system, orientated for a high typographic quality. It is used specially, because of its features and possibilities, for generating articles and scientific books in which the use of mathematics expressions is so important, but also in statistics, economics or political science [21]. The writer uses code as tagging conventions to establish the structure of the document, defining the style of the text and adding tables and other elements such as pictures, animations or 3D models.

LaTeX is based on TeX commands. TeX is a typographic system written by Donald E. Knuth, that, annoyed with the low quality of his first books, decided to design his own language and he finished it in early 80s [22]. TeX orders are difficult to analyse, starting with an inverse slash ("\"), adding arguments into keys ("{}") and based on basic orders and macros. Most of them are included in original Knuth's plainTeX, LaTeX for technic science in its majority, and ConTeXt, for publications.

LaTeX was created in 1984 in order to facilitate the use of TeX, providing a high-level language, comprising a collection of its macros and creating a program to process LaTeX documents. It purposes a different work philosophy- working with instructions for focusing in the content of the document, not in the format details. Its graphics capabilities able the user to structure easily the document and he can obtain very attractive articles and books [23].

LaTeX requires two steps: in the first one we have to create the file in a text editor which contains the text we want to publish, and in the second we will process the text. In this part the text processor interprets the orders written on it and compiles the document, leaving it prepared for being sent to its destination. If we want to add or modify something, we will have to change whatever we want in the file and process it again, just as in high level programming languages like C or C++.

LaTeX documents have compatibility with any text editor, they consist of plain text so do not contain hidden formatting codes, and they can be rendered to PDF files using extension pdfLaTeX [24].

In brief, LaTeX is a very powerful tool for the creation of 3DPDF documents. It only needs the call to *movie15* or the newest *media9* packages. Its capabilities and features make this software a very useful way to implement our templates and establish our workflow between the 3D model and data and the PDF file.

2.3.4. CAD Exchanger

CAD Exchanger conceived for the CAD data interoperability. It allows to visualize 3D data, convert it in a numerous amount of different CAD, mesh, and other formats. The software works with standard formats, as IGES, STEP or STL and with modelling specific formats like ACIS, Parasolid or Rhino/Open NURBS [25].

CAD Exchanger was created by the company CADEX in 2014 by Roman Lygin, despite some beta versions were realised some years before. It started like other tech start-ups, as a hobby, but with rich background in software development and management and inspired in parallel computations, the founders prototyped a 3D visualization and conversion app that received positive feedbacks from users. They went on the improvement of the software and finally founded the company [26].

The problem with CAD data interoperability has been present since CAD systems started to be designed. The technical factors like the differences in format designations and descriptions of 3D data, as colours, materials or metadata, for instance, or the different mathematical descriptions used in geometrics underneath every CAD system come into play. Also the inclusion of features in a specific and encompassing format as STEP ends up in complications and ambiguous interpretations of the standards. All of this make big companies to create their own formats or work with neutral formats but giving priorities to their own needs [27].

As the complexity related to data conversion, because not only to the appearing of new version of existing formats but the grow of complexity and size of them, is growing, new efficient methods are required, and CAD Exchanger is based in one of these methods. The combination of using parallel computations, a system based on the idea that big problems might be solved with little solutions as a whole [28], is what makes this software a powerful tool for this issue. CAD Exchanger includes three different packages, that are:

-GUI, a graphical user interface app that allows visualize 3D models in different views and interact with the assembly, giving also different geometrical information of it.

-SDK, a set of C++ libraries for developers, so they are able to, for example, importing exact geometry in CAM systems from STEP or JT or analyse product structure and geometrical data of a 3D model.

-CLI, a server-based solution designed for the collaborative usage, so the designers can share all their data in cloud-based services in large companies, for instance.

In our case, CAD Exchanger will be used for the conversion from STEP to X3D files. The company offers free educational versions and site-wide licensing, so the software can be used on unlimited number of computers. It is very useful, in that way, the flexibility they offer us.

2.4. Ways to implement the solution

The creation of a 3D PDF was usually an unsuccessful process, since the 3D model was exported in a mesh format and was not possible to create a model tree, for example, the quality was no as high as we wished and was very slow, due to the fact that there was not a specific template to work with. The old workflow is shown in figure 2.2. Once the 3D model was created in OPUS, the CAM software used, it was exported as a STL file, as a mesh, and imported in MeshLab in order to convert the file in a U3D format. Finally, it was embedded in a PDF document with LaTeX, but with a very simple template created also in this software. In this process, a lot of information, starting from the colour and textures and continuing with PMI and other annotations were lost. There was, additionally, the possibility to create a 3D PDF document from OPUS. This option, though, was dismissed since there was no possibility to create a template or add any table or further information. The document was composed by the 3D model and nothing else, so the operator was able to interact with the assembly but had no access to its details.

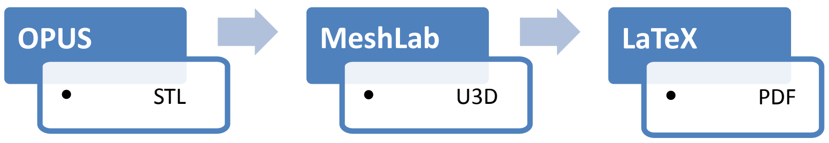


Figure 2.2. *Old workflow.*

Three solutions have been thought for this issue, but they have been based in the old workflow, with the idea of optimize it and improve it as much as possible. The aspects that have been present are the quality of the final product and the savings of money and time. A primordial requisite was, also, that the solution had to work with several CAD/CAM software but above all with OPUS.

With the guidelines given, we had to look for the suitable software in order to establish connections between them and develop a workflow with their connections. The implemented solutions have been the next ones:

- Utilization of a Web application for the interaction with a 3D model. In this case, we will work in the conversion of the STEP model in a X3D file without the loss of information and characteristics and the implementation of the model in a different environment, easy to share and use for the operator. This solution's workflow is exposed in figure 2.3.

- Use of Tetra4D for the embedding of the 3D model in a 3D PDF document. As a very powerful tool, Tetra4D Enrich can provide us an intuitive and brief design of the templates necessaries for the creation of the documents and the additional features as tables, annotations and interactive buttons.
  The problem of the solution will be the economic cost. The solution diagram is shown in figure 2.4.

- Design of an interactive 3D PDF with the use of 3D-Tool and LaTeX. This solution will suppose an economic and limitless solution, since the last step of the process, with the pdf editor will supply a lot of different options and with a STEP 3D model, so the quality will increase considerably. It will be, though, necessary to change some code by the time of changing the model in the template. In figure 2.5 we can see the diagram of the workflow.

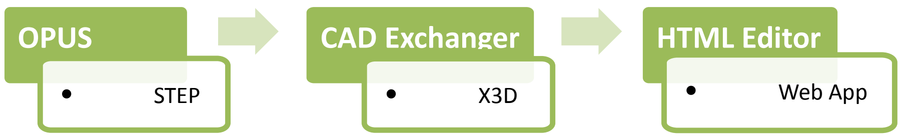


Figure 2.3. *Web App workflow.*

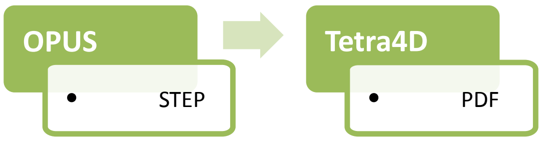


Figure 2.4. *Tetra4D workflow.*

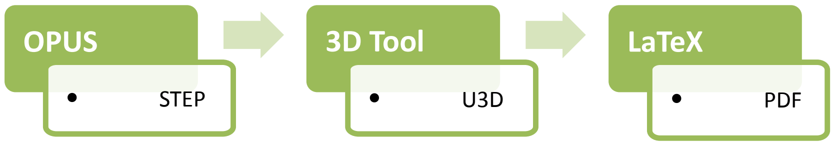


Figure 2.5. *3D Tool and LaTeX workflow.*

CHAPTER 3: IMPLEMENTATION

In this section we will talk about the way the solutions have been implemented and how we have created the interactive documents and web applications. The research has been based in three different aspects, finding the way to have a web application for X3D files and designing two kinds of PDF documents with different programs, features, effort to put and time to spend on it. The implementation of the solutions will be explained step by step in order to be as a guide for the creation of a workflow and a way to design templates or applications for 3D models in diverse sources. It will include explanations, images and code when necessary.

3.1. Web application in a browser

The use of a Web application for interacting with a 3D model is a different to implement the idea of the creation of a document. In this case, we do not create a template and embed the 3D model, but we create an app with which the user, in this case the operator, will be able to have all the views, move the model, have a tree of the pieces that compose the assembly and have further information in this app. The software needed for the start-up will be:

- CAD-Exchanger as file converter.
- Browser (Chrome, Firefox, IE, etc., with WebGL technology)
- HTML text editor and compiler (CoffeeCup Free HTML Editor)

3.1.1. Generation of the set-up web app

As said before, we will not need a set-up document as a template, but we will need to have a base, a place in which embed our 3D model, and this will be the application. With a non-very extended knowledge in HTML language, it is difficult to design something like this, but fortunately exists nowadays a huge amount of information and examples on the Internet, so it is possible to find free-open source work related and use it or modify, always considering the author of it, Christian Stein [28] in this case. Browsing we see a lot of material based on X3DOM, and websites specialized in this area. One of these (Doc.x3dom.org), provide us with examples and tutorials so we can learn the language and programming with practical cases and another one (GitHub) is a forum where developers and people interested in this issue can share and spread through their works and codes.

In the last, it is possible to obtain the code of a CAD Viewer, a Web application prepared for supporting the addition of 3D models and its data, as well as a tree of the assembly pieces and programming its views. The CAD Viewer looks like in figures 3.1. and 3.2.

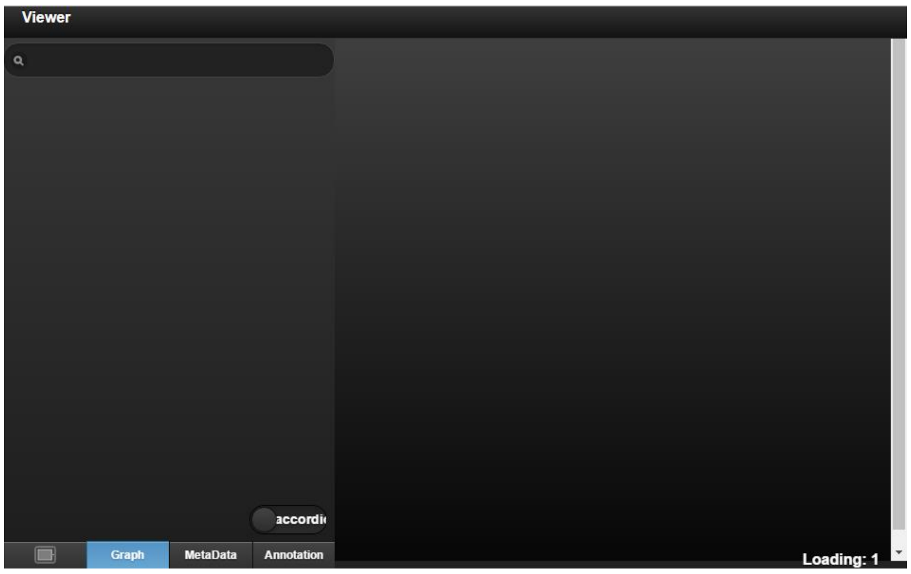
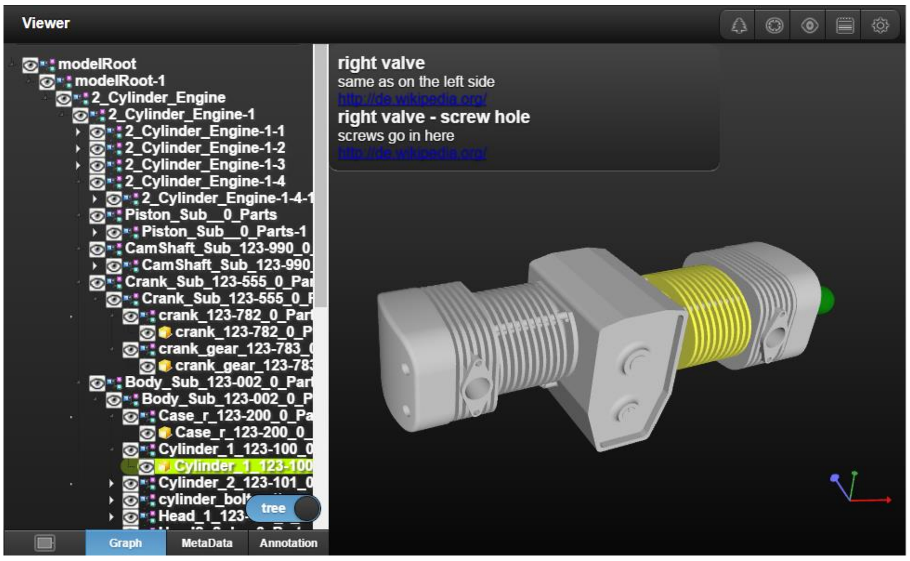


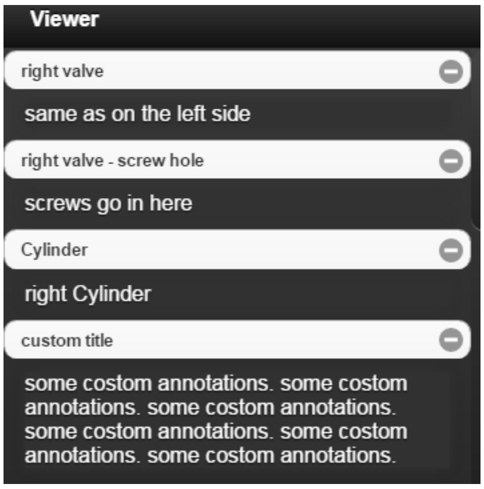
Figure 3.1. *CAD Viewer without any 3D model.*



right valve
same as on the left side
http://de.wikipedia.org/
right valve - screw hole
screws go in here
http://de.wikipedia.org/

Figure 3.2. *CAD Viewer with 3D model example features.*

The web example allows us to have an idea of the features and capabilities it has. As said before, we can open the assembly parts tree so we can select, in yellow in this case, the desired one. There are also different ways to view it. It has been implemented annotation and metadata sections so the developer can add further information if necessary, as seen in figure 3.3. We also have different views in order to facilitate the interaction with the model, showed in figure 3.4.



Viewer

right valve
same as on the left side

right valve - screw hole
screws go in here

Cylinder
right Cylinder

custom title
some costom annotations. some costom annotations. some costom annotations. some costom annotations. some costom annotations. some costom annotations.

Figure 3.3. *CAD Viewer example.*

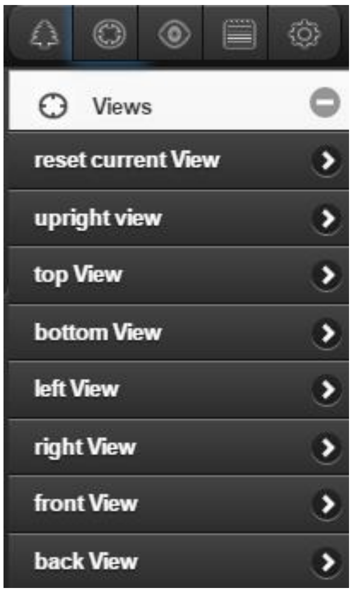


Figure 3.4. *CAD Viewer example views.*

If we want to obtain this application, we have to go to the pertinent website where users can share their projects and download the files located there and follow the instructions. Now we will have at our disposition the set up environment in which we will add the wished 3D model after a few modifications explained below.

3.1.2. Embedding of the 3d model

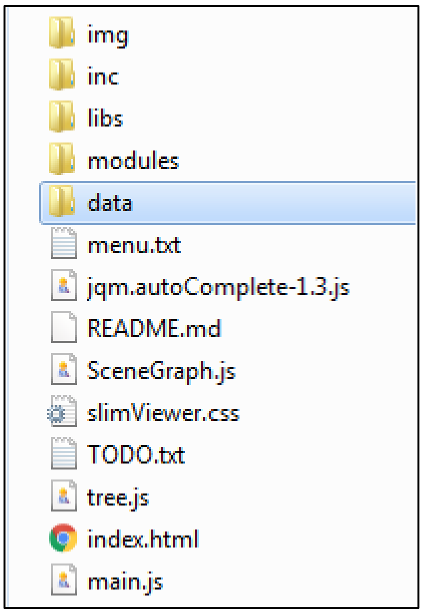
Once we have the application available, we have to focus in creating the X3D file from the STEP 3D model. We have different options. It is possible to generate a X3D file from a mesh format as STL in FreeCAD or MeshLab, but the problem would be the loss of information as the colour and that we will have a unique solid, not an assembly, so it will not be possible to interact viewing the different components, creating a part tree.

Considering that issues, we will use CAD Exchanger, a software capable of converting STEP files into X3D conserving all the characteristics. It is a commercial software, so it will suppose an economic cost to make use of it, but we will put the quality ahead of the economic wastes.

Opening the software, we will see that is very intuitive and easy to carry the conversion out. We only have to drop a file on the appropriated space for it or look for the file in the computer. The software will look as in the figure below, with the 3D model we want to use in this case.

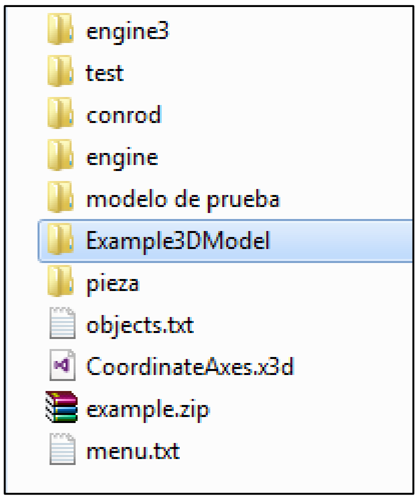
After that, we select Export in the menu, and we transform the model into a X3D file and save it in a folder we can remember easily. Done the conversion, we have to move again to the website from where we downloaded the Web app (GitHub). There we will see some instructions to follow in order to add the model into the app. Before continuing we will need to open another software. It is CoffeeCup Free HTML Editor, that will enable us the possibility of editing the code that we will need to modify, compile it and show the result in a browser or internal console. We can, though, use another HTML editor, it will not suppose a difference in our work.

The next step consists on getting the data folder, and create inside a folder with the wished name. The folder data is shown in figure 3.5. with the other downloaded files and the example folder we will create will be called Example3DModel, as seen in figure 3.6. After that, we will have to open the *main.js* file in the HTML text editor, and modify the name after `'MYAPP.model'` with the name of the folder [30], as we can see in figure 3.7. When finished that, the only thing we will have to do is pasting the X3D file in the folder Example3DModel, in our case, with the name "model.x3d". Every time we want to embed a new 3D model in the app, we will have to create a new folder, change the name in the code as shown before and save the X3D file in the new folder as "model.x3d" again.



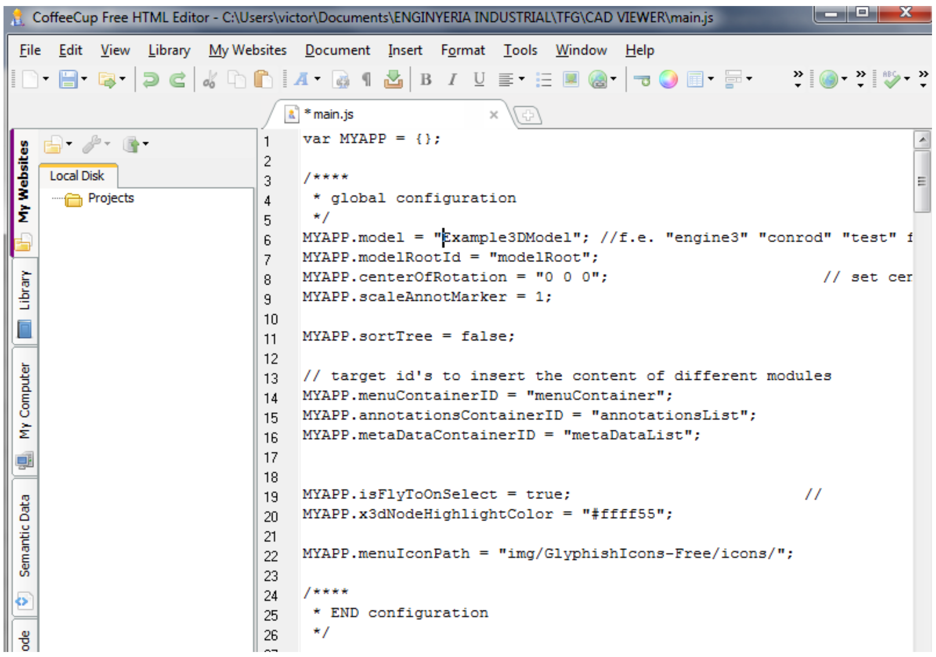
img
inc
libs
modules
data
menu.txt
jqm.autoComplete-1.3.js
README.md
SceneGraph.js
slimViewer.css
TODO.txt
tree.js
index.html
main.js

Figure 3.5. *CAD Viewer files folder.*



engine3
test
conrod
engine
modelo de prueba
Example3DModel
pieza
objects.txt
CoordinateAxes.x3d
example.zip
menu.txt

Figure 3.6. *Data folder.*



CoffeeCup Free HTML Editor - C:\Users\victor\Documents\ENGINYERIA INDUSTRIAL\TFG\CAD VIEWER\main.js
File Edit View Library My Websites Document Insert Format Tools Window Help
* main.js
My Websites
Local Disk
Projects
Library
My Computer
Semantic Data
ode

```
var MYAPP = {};

/****
 * global configuration
 */
MYAPP.model = "Example3DModel"; //f.e. "engine3" "conrod" "test" f
MYAPP.modelRootId = "modelRoot";
MYAPP.centerOfRotation = "0 0 0";                    // set cer
MYAPP.scaleAnnotMarker = 1;

MYAPP.sortTree = false;

// target id's to insert the content of different modules
MYAPP.menuContainerID = "menuContainer";
MYAPP.annotationsContainerID = "annotationsList";
MYAPP.metaDataContainerID = "metaDataList";


MYAPP.isFlyToOnSelect = true;                       //
MYAPP.x3dNodeHighlightColor = "#ffff55";

MYAPP.menuIconPath = "img/GlyphishIcons-Free/icons/";

/****
 * END configuration
 */
```

Figure 3.7. *Main.js code after modification.*

We also will be able to change the centre of rotation if necessary in the same configuration code, after `'MYAPP.centerOfRotation'`, or the colour of highlighted parts, after `'MYAPP.x3dNodeHighlightColor'` writing the HTML colour corresponding code, like #FF2D00.

Now, if everything told before is already done, our 3D model will be added in the Web application and we will be able to open it in a browser, Mozilla Firefox in this case, and move, rotate and interact with it, highlighting, furthermore, the wished parts and even hiding them. It is shown in figure 3.8.

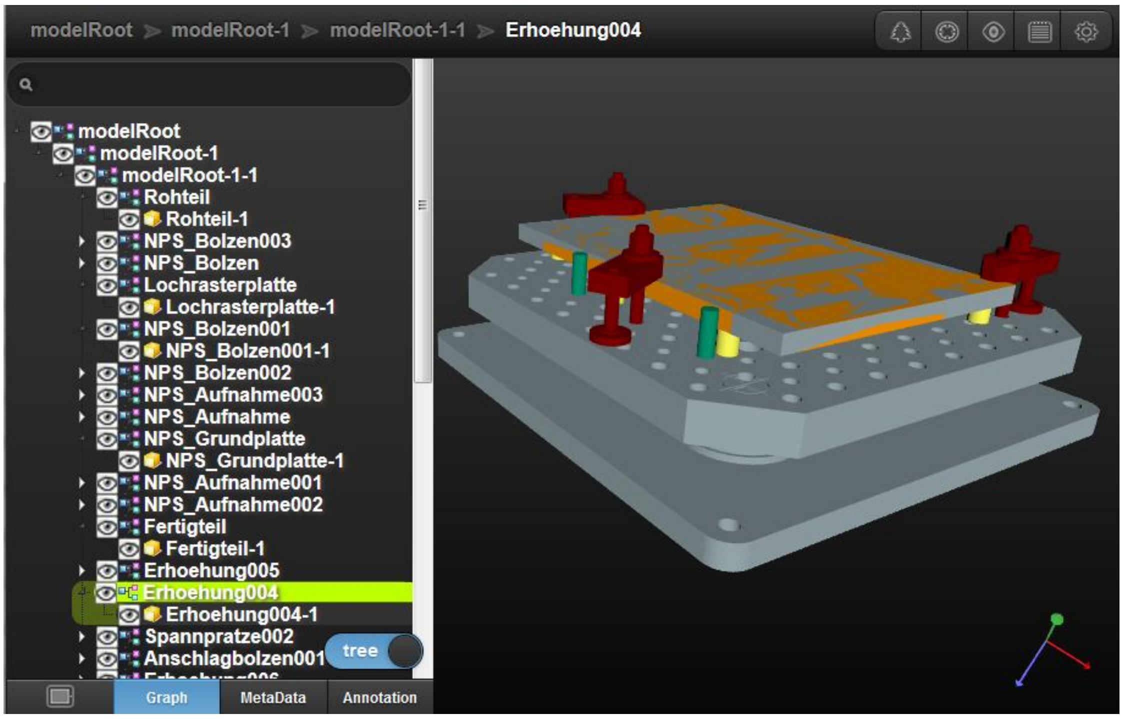
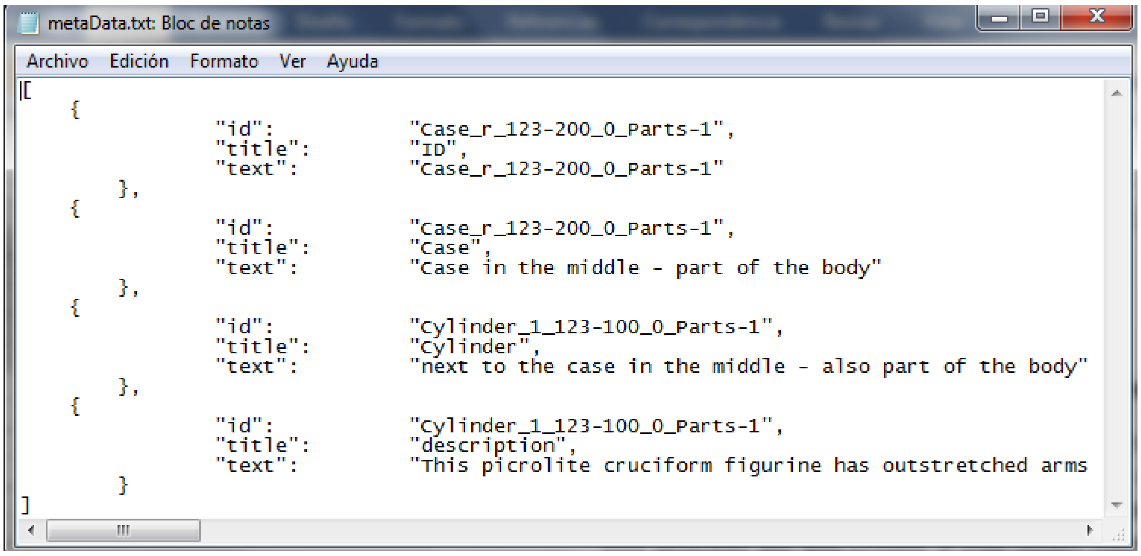


Figure 3.8. *Example 3D model embedded in the Web App.*

3.1.3. Embedding further required information into the web app

It exists the possibility of adding the 3D model data in the Web app. What we have to do is creating a TXT file and saving it in the same folder as the model, inside the data folder. With this process we will create a pull-down menu in the MetaData section of the Web application. The example given by the developer looks like figure 3.9.

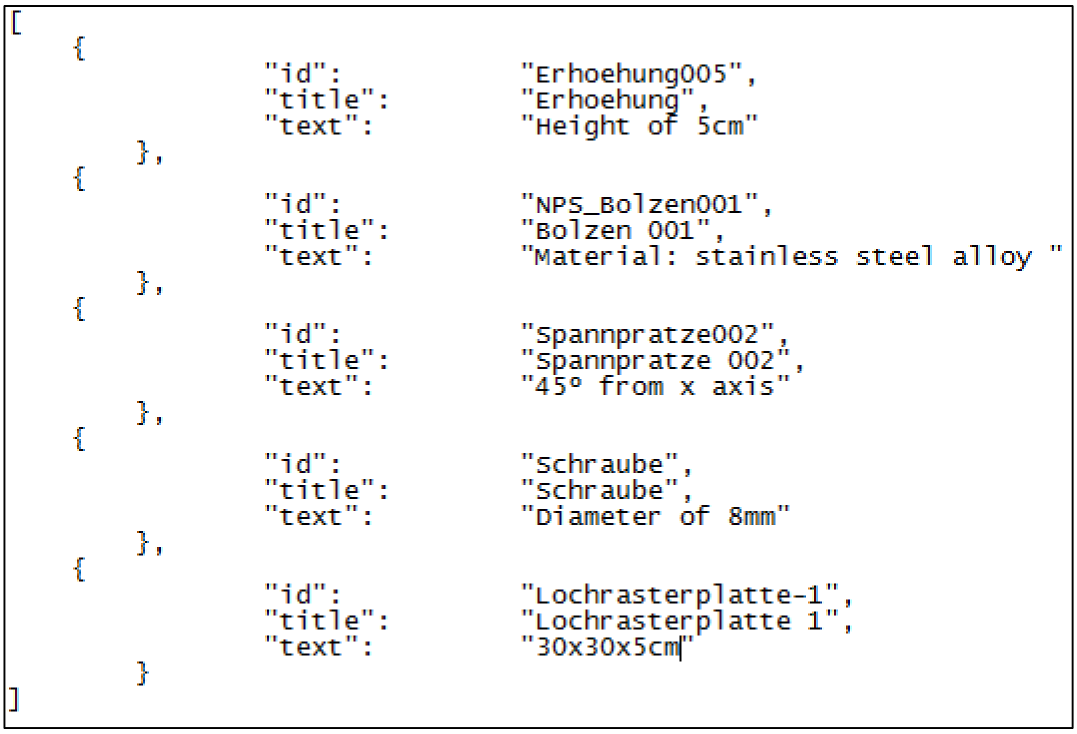


```
[
    {
            "id":           "Case_r_123-200_0_Parts-1",
            "title":        "ID",
            "text":         "Case_r_123-200_0_Parts-1"
        },
    {
            "id":           "Case_r_123-200_0_Parts-1",
            "title":        "Case",
            "text":         "Case in the middle - part of the body"
        },
    {
            "id":           "Cylinder_1_123-100_0_Parts-1",
            "title":        "Cylinder",
            "text":         "next to the case in the middle - also part of the body"
        },
    {
            "id":           "Cylinder_1_123-100_0_Parts-1",
            "title":        "description",
            "text":         "This picrolite cruciform figurine has outstretched arms
        }
]
```

Figure 3.9 *Metadata of the developer's example.*

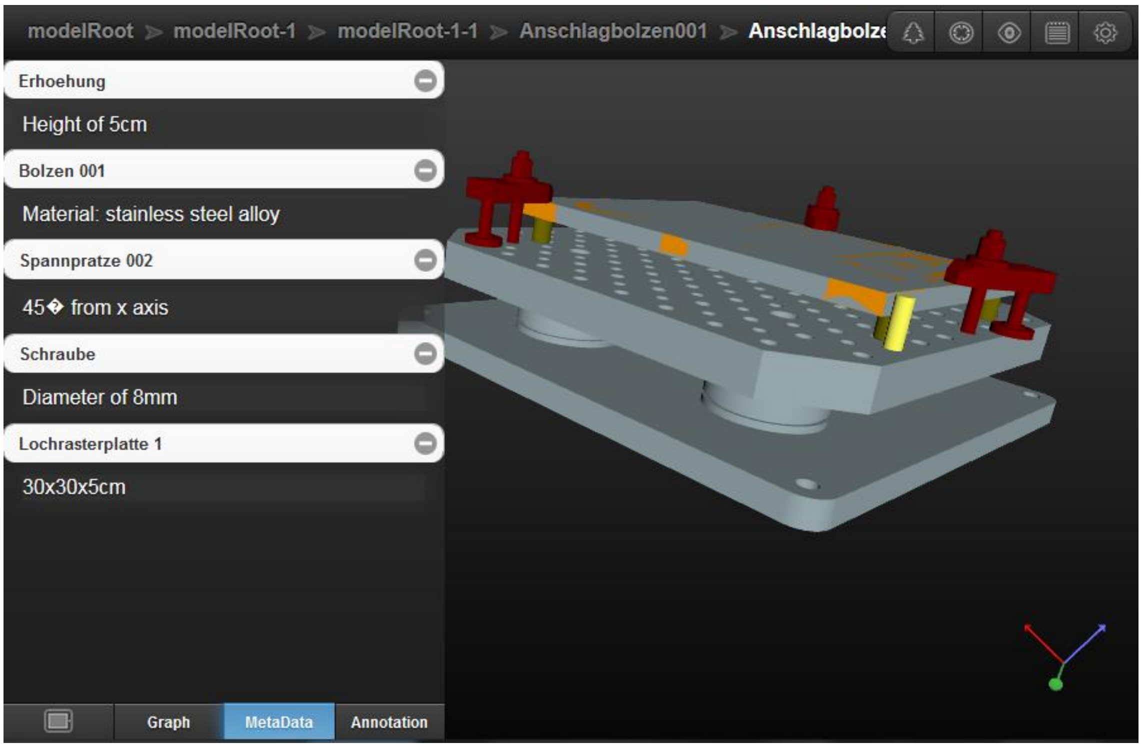
As we can see in figure 3.9, the TXT file has a specific structure, due to the fact that has to be read by the *main.js* code. The text is between the keys [] and every part is situated enclosed by the keys {}. In addition, we can relate each datum to a part of the assembly, typing first "id" and naming the part as in the tree. Then we name the part in "title" and after that, in "text", we add the information. We will implement some data in our Example3DModel. It is shown in figure 3.10. It is important to save it in Example3DModel folder, in our case, with the name "metadata.txt". It would be a further investigation issue trying to find a way to create this kind of file directly from the CAD/CAM software, with the same structure to reduce the workflow time. The overall view of the result is seen in figure 3.11.

Moreover, we can add annotations in the 3D model, a very useful fact for the operator, since the designer can make comments or explanations about a specific part of the work process or the collocation of a piece, for instance. It is done just in the same way as the metadata, but saving the TXT file as "annotations.txt". An example is done in figure 3.12.



```
[
    {
                "id":           "Erhoehung005",
                "title":        "Erhoehung",
                "text":         "Height of 5cm"
        },
    {
                "id":           "NPS_Bolzen001",
                "title":        "Bolzen 001",
                "text":         "Material: stainless steel alloy "
        },
    {
                "id":           "Spannpratze002",
                "title":        "Spannpratze 002",
                "text":         "45º from x axis"
        },
    {
                "id":           "Schraube",
                "title":        "Schraube",
                "text":         "Diameter of 8mm"
        },
    {
                "id":           "Lochrasterplatte-1",
                "title":        "Lochrasterplatte 1",
                "text":         "30x30x5cm"
        }
]
```

Figure 3.10. *Metadata of Example3DModel code.*


modelRoot > modelRoot-1 > modelRoot-1-1 > Anschlagbolzen001 > Anschlagbolze
Graph
MetaData
Annotation

Erhoehung
Height of 5cm
Bolzen 001
Material: stainless steel alloy
Spannpratze 002
45� from x axis
Schraube
Diameter of 8mm
Lochrasterplatte 1
30x30x5cm

Figure 3.11. *Metadata of Example3DModel in the Web app.*

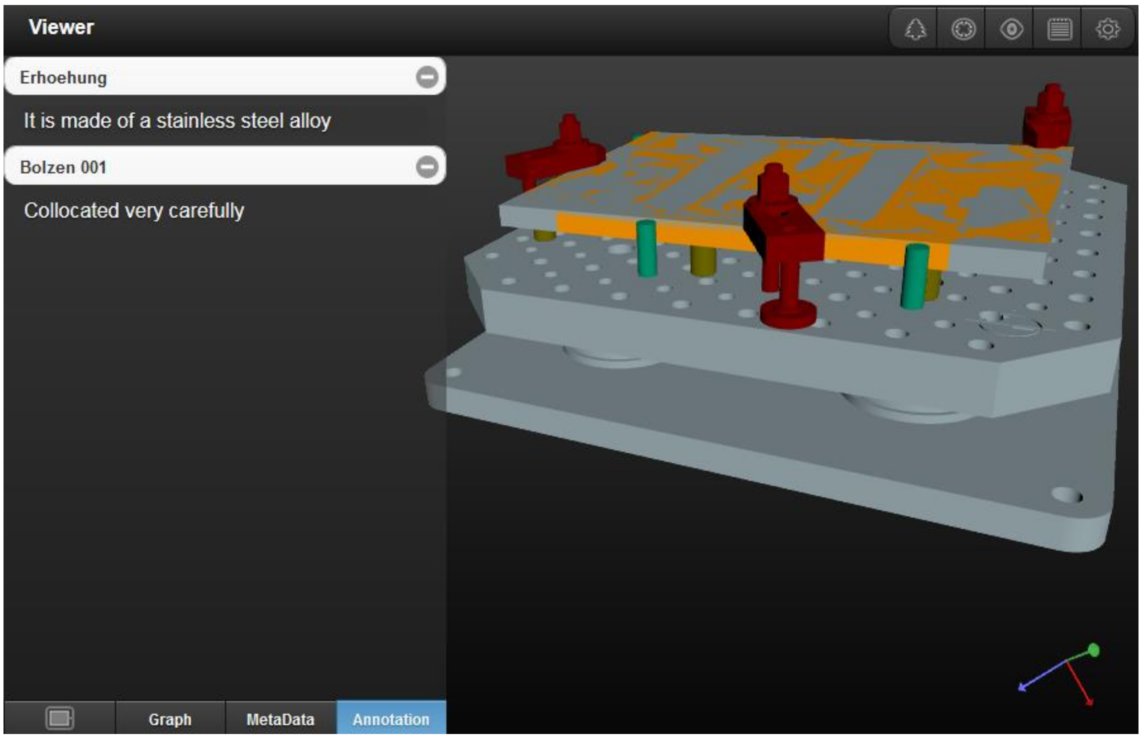

Viewer
Erhoehung
It is made of a stainless steel alloy
Bolzen 001
Collocated very carefully
Graph
MetaData
Annotation

Figure 3.12. *Annotations of Example3DModel in the Web app.*

To sum up, the use of the Web application for sharing the interacting 3D model is a good option, even not being a document itself. It also has disadvantages. We cannot insert a table of the parts of the assembly, for instance, or add a text field in which the operator can write commentaries and annotations about the model or the process so a feedback with the designer cannot be established. Furthermore, a document like a PDF file could be more easily interpreted organising all of its elements in a unique sheet. However, the facility of rendering the Web app in a mobile phone or tablet provides this method versatility and freedom to the user, since there is no need of a commercial software to visualize it. It is true that, in this case, we are using an already developed application, but it is a good starting point. Obviously we could learn the way to modify it depending on our necessities and requirements or developing a new program.

The fact is that implementing this solution we will stablish a powerful, reliable and easy workflow, since the generation of a new app is done by creating a new folder with the model and changing a name in the code. Additionally, it is a relatively cheap option.

3.2. Creation of pdf document with tetra4d

Tetra4D is a commercial and powerful tool with which we will be able to create interactive PDF documents easily. We have the possibility of making our own template, adding on it the 3D model with just a click of our mouse, without the necessity of having converted it into U3D or X3D, just with the STEP file. Its methods and features will provide us an earning of time and effort. The software necessary for this implementation will be:

- Tetra4D Enrich
- Acrobat Pro DC or XI

3.2.1. Generation of the set-up document

There are two different ways to create our template if we decide to use Tetra4D. The first one would consist in creating a document in text editor software, adding tittles and leaving space for the elements we want to embed, as the 3D model, a tables or other information related to the assembly. Once the document is organized we will have to save it as a PDF file and it will be ready to be manipulated in Tetra4D. This option, despite being a very easy one, has many disadvantages. It is, for instance, necessary to create a new document every time we want to publish a new 3D model, so this does not represent an optimization of the process. Our idea is creating a defined template with a default structure and in which we only had to change the name of the model, the 3D model itself, the table of its components and its PMI data. This is possible to do with Tetra4D.

For this the second option to implement our workflow, we will use the chosen software. We have, also, two different ways to design the template in this case. We can create our own template or use one of the defaults Tetra offers us. In the last case, the document is divided in different parts, including the company logo, a collection of buttons for the interactivity and space for tables or information about the assembly. It would look as in figure 3.13. It is possible, furthermore, to modify the appearance of the default template to our likeness, changing the colours, the font or the logos for example.

Tetra4D Enrich gives us also the chance to design our own template in a very intuitive way and with tutorials and instructions in order to help us. We will explain how to do it, creating our example.

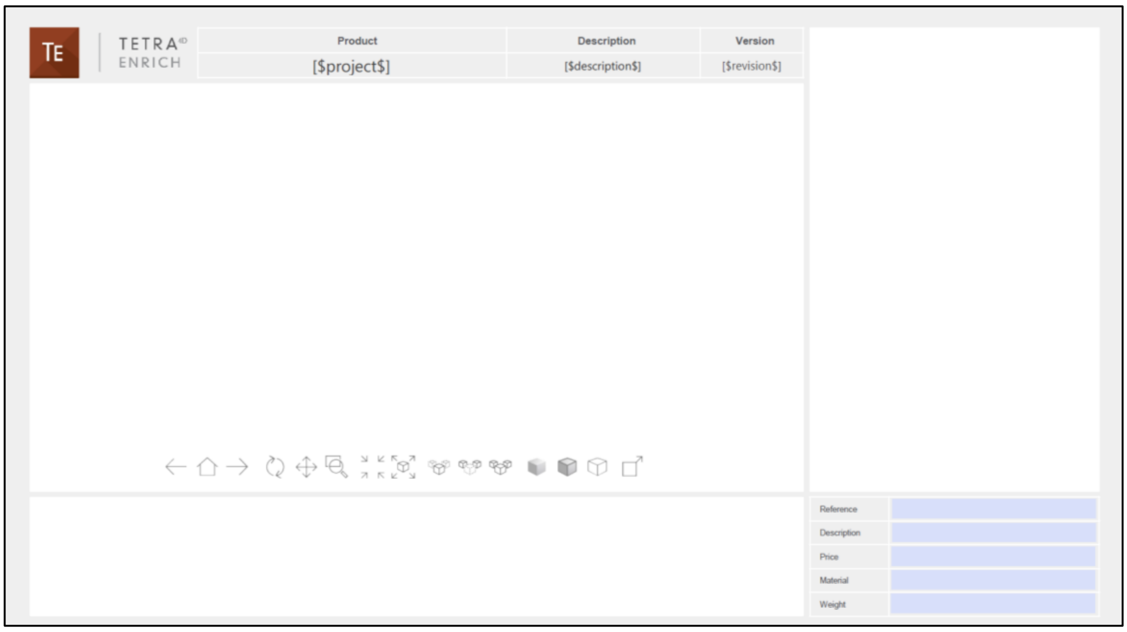


Figure 3.13. *Example template from Tetra4D Enrich.*

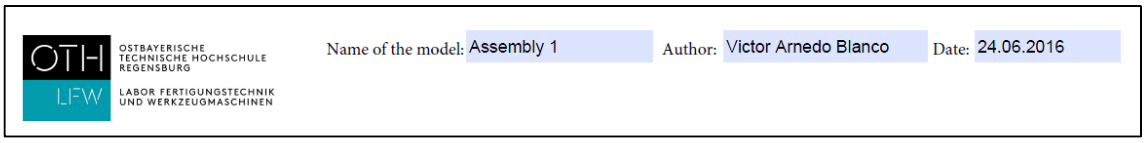
The first thing we should do is preparing the software involved in the workflow. These are, Adobe Acrobat Pro and Tetra4D Enrich. We will find 28 days' free trial versions in their Websites[3], but if we want to continue with this solution, we should buy the licenses with the price later will be exposed. Once everything is installed in the operating device we can start with the design of the template. The features we have available are creating tittle, subtitle and description, as well as interactive tables and spaces for further information and buttons with which the operator can interact with the 3D model, changing the view of it for example.

We have to create a white sheet document in a text editor like Microsoft Word and save it as a PDF file, so we will have the base in which work in Tetra4D Enrich. Once done, we should add a default structure to our document; for example, adding a firma or university logo, a space for the tittle, the author and the date the assembly was done or published. We will do this with Acrobat Pro, editing the PDF and adding text and images. With the space left for the things said before, we could use formulary boxes so every time we open a new template we just have to change the text in the corresponding box. The result can be seen in figure 3.14.

It would be appropriated to follow the idea of the company examples, since it is suitable for our purposes. We should leave a space for the 3D model in the left side of the document and another one below for buttons, views or text field for annotations, and on the right for the table of pieces and PMI

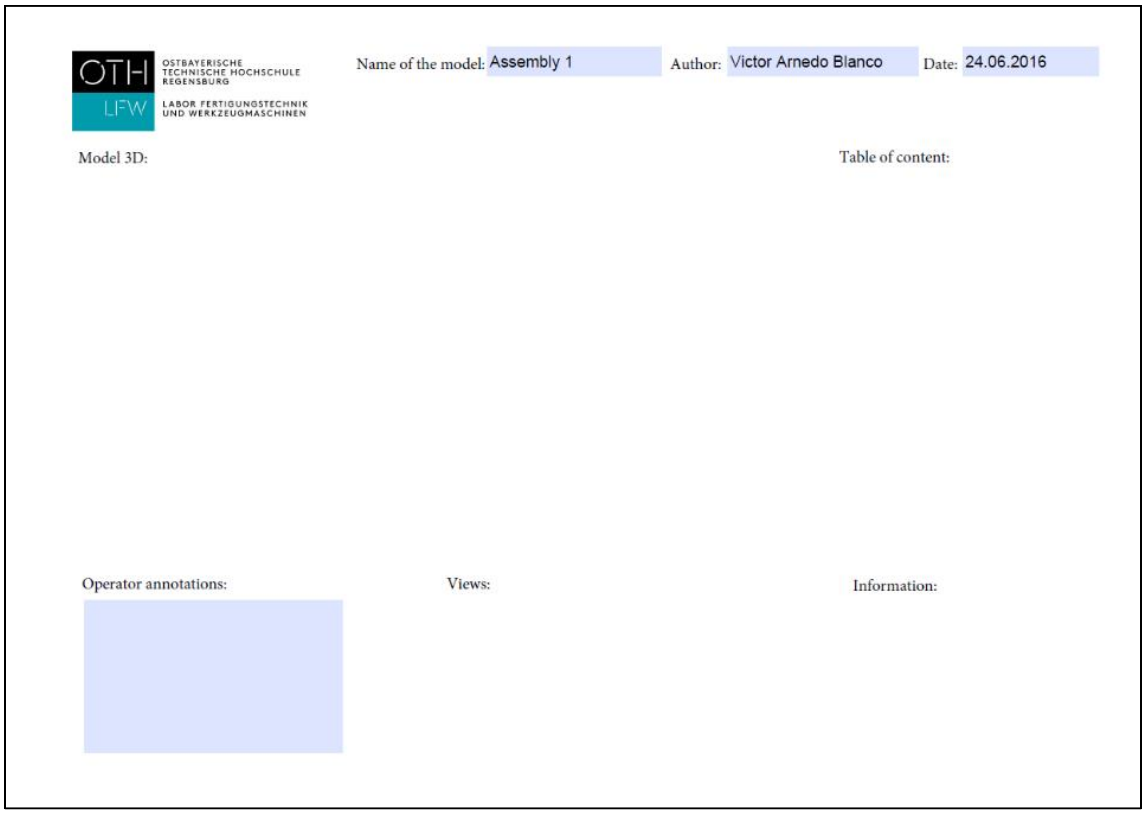
[3] Websites:< *https://acrobat.adobe.com/us/en/free-trial-download.html>,* *<http://www.tetra4d.com/tetra4d-enrich/>*

data or further information. Before inserting the interactive content, the template would look like in figure 3.15.


OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
LABOR FERTIGUNGSTECHNIK UND WERKZEUGMASCHINEN

Name of the model: Assembly 1 Author: Victor Arnedo Blanco Date: 24.06.2016

Figure 3.14. *Head of the created template.*


OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
LABOR FERTIGUNGSTECHNIK UND WERKZEUGMASCHINEN

Name of the model: Assembly 1 Author: Victor Arnedo Blanco Date: 24.06.2016

Model 3D:

Table of content:

Operator annotations:

Views:

Information:

Figure 3.15. *Created template.*

Obviously this is just an example, a model of template we can use. Tetra4D Enrich is capable of provide us endless kind of designs and configuration for templates. Once this is done, the next step will be the embedding of the 3D model in the document.

3.2.2. Embedding 3d model in the document

This task supposes one of the easiest parts in this workflow. Tetra4D Enrich, as said before, is capable of adding not only U3D files in the 3D PDF documents but also converting STEP files in U3D or even embedding STEP files in the templates with just one click, since it includes all the Tetra4D Converter features. If our template is done, we can add the 3D model going to the tools menu of Converter in Acrobat Pro and clicking in Add 3D, as we can see in figure 3.16.

We select our 3D model file and the area of the template we want to take up for it. We will have to select some quality options as the colour of the background, depending on our necessities. The result is shown in figure 3.17.

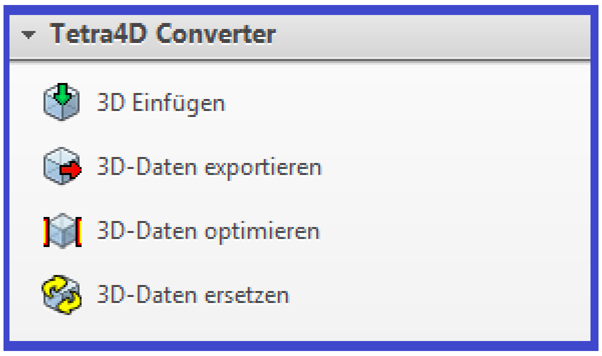
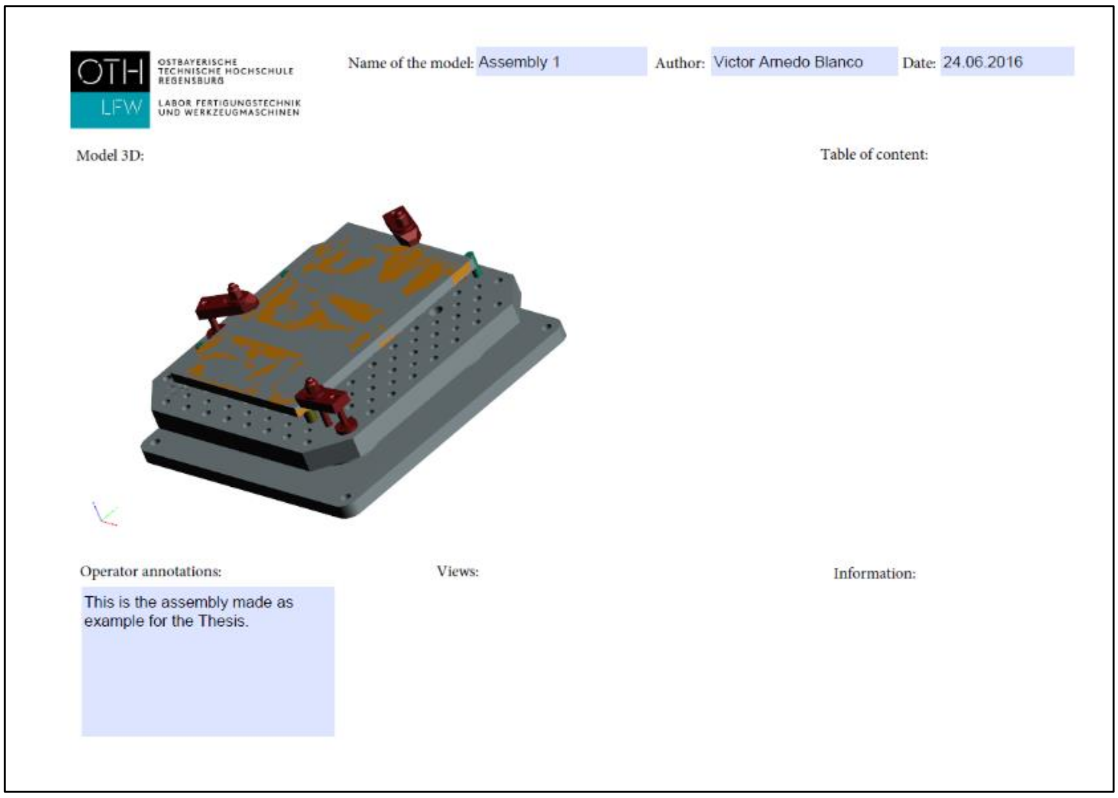


Figure 3.16. *Tetra4D Converter features.*


OTH OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
LFW LABOR FERTIGUNGSTECHNIK UND WERKZEUGMASCHINEN
Model 3D:
Table of content:
Views:
Information:

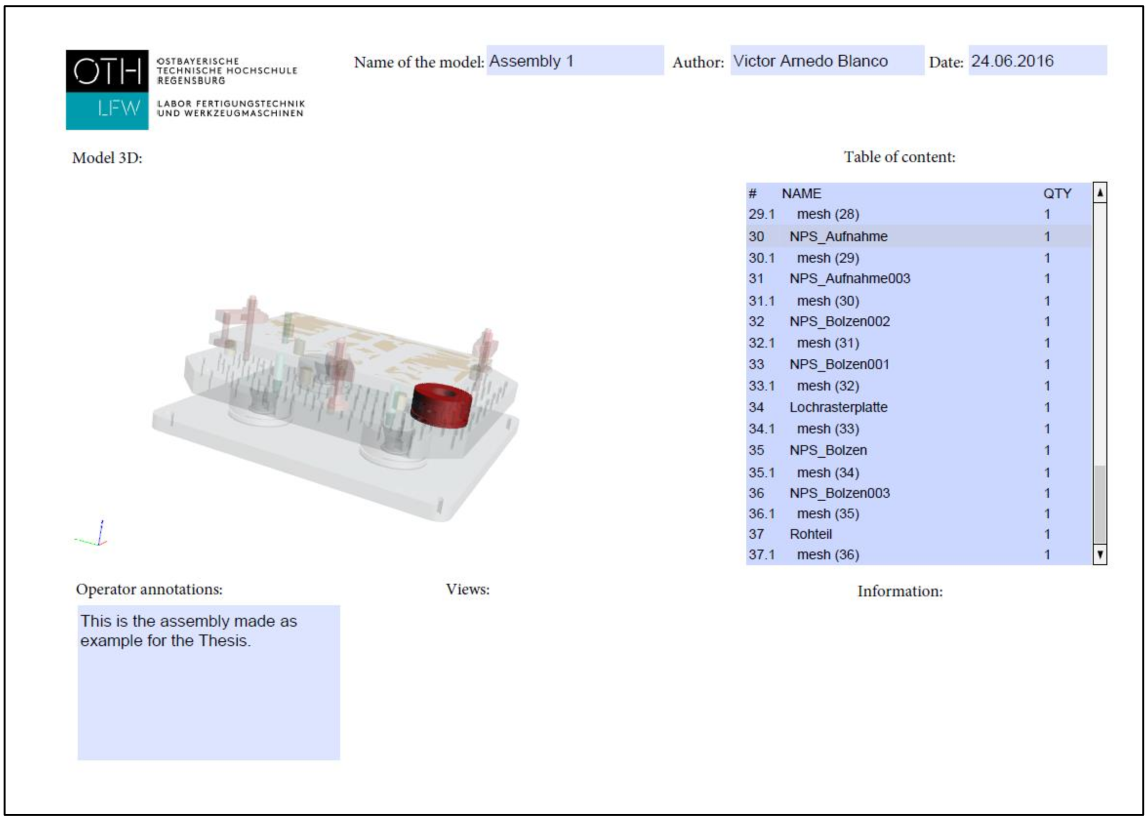
Name of the model: Assembly 1
Author: Victor Amedo Blanco
Date: 24.06.2016
Operator annotations: This is the assembly made as example for the Thesis.

Figure 3.17. *Template with 3D model embedded.*

3.2.3. Embedding further required information into the document

Tetra4D Enrich also gives us a lot of facilities for adding in our 3D PDF documents information about the 3D model and other features are possible to combine for compounding a very interactive document. The first thing we can add is a table of the components of the assembly. We have different ways to create this table. The software itself can generate the part lists or alternatively import it from external data like a CSV file or from a clipboard generated in a spreadsheet created in an editor like Microsoft Excel.

If we want the software to create it, there is the possibility to make a flat list, in which every part acquires a number but they have no relation on any other part. Additionally, we could design a hierarchical list, in which the hierarchy of the 3D information is reflected with sub-assemblies generated. The composition of the number of the list changes in this way. We can see an example of a table with the highlighted component in figure 3.18. We only have to click on this option in the menu of Tetra4D Enrich, shown in figure 3.19.


OTH
LFW
OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
LABOR FERTIGUNGSTECHNIK UND WERKZEUGMASCHINEN

Name of the model: Assembly 1 Author: Victor Amedo Blanco Date: 24.06.2016

Model 3D:

Table of content:

#	NAME	QTY
29.1	mesh (28)	1
30	NPS_Aufnahme	1
30.1	mesh (29)	1
31	NPS_Aufnahme003	1
31.1	mesh (30)	1
32	NPS_Bolzen002	1
32.1	mesh (31)	1
33	NPS_Bolzen001	1
33.1	mesh (32)	1
34	Lochrasterplatte	1
34.1	mesh (33)	1
35	NPS_Bolzen	1
35.1	mesh (34)	1
36	NPS_Bolzen003	1
36.1	mesh (35)	1
37	Rohteil	1
37.1	mesh (36)	1

Operator annotations: This is the assembly made as example for the Thesis.

Views:

Information:

Figure 3.18. *Template with hierarchical table inserted and piece highlighted.*

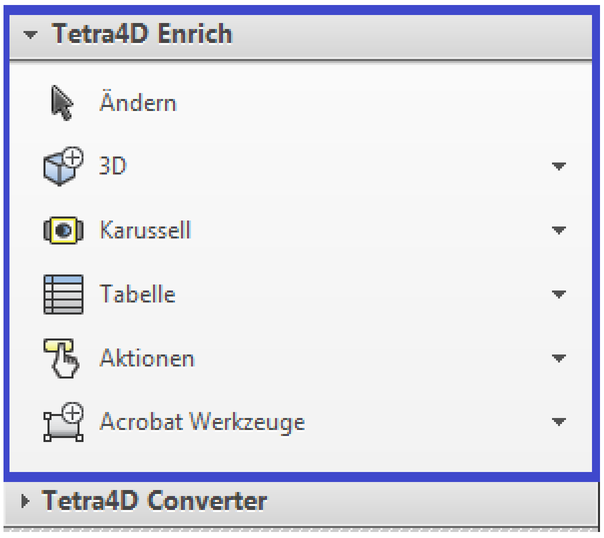
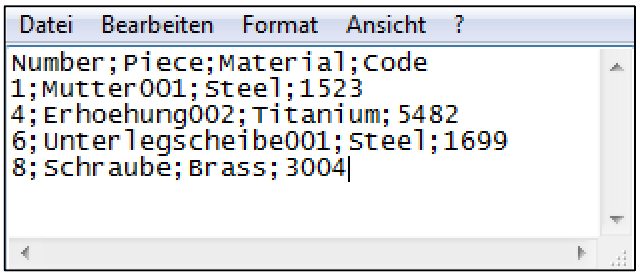
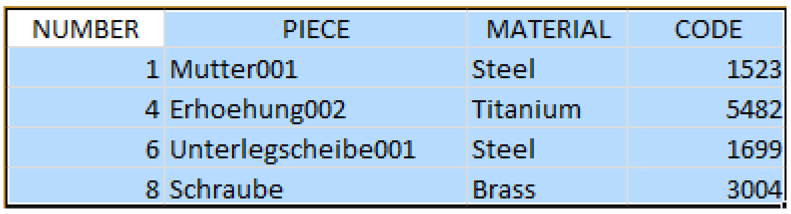


Figure 3.19. *Tetra4D Enrich features.*



```
Number;Piece;Material;Code
1;Mutter001;Steel;1523
4;Erhoehung002;Titanium;5482
6;Unterlegscheibe001;Steel;1699
8;Schraube;Brass;3004
```

Figure 3.20. *Example CSV table.*



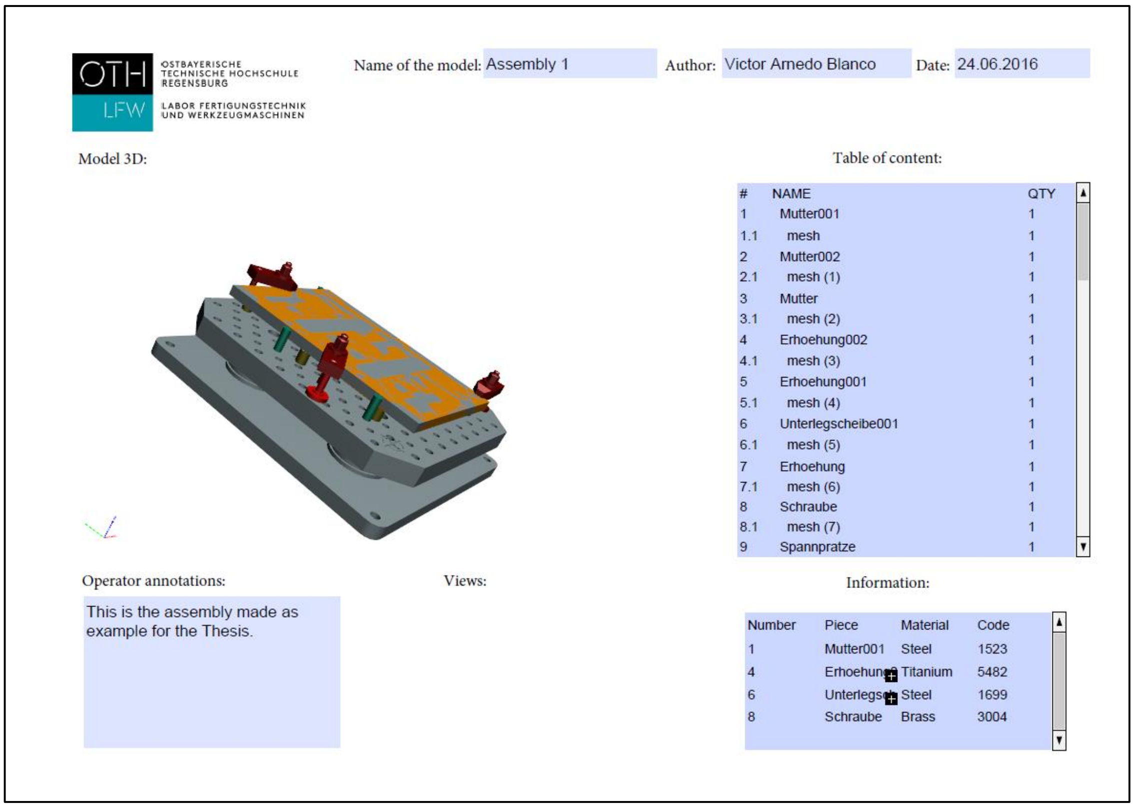
NUMBER	PIECE	MATERIAL	CODE
1	Mutter001	Steel	1523
4	Erhoehung002	Titanium	5482
6	Unterlegscheibe001	Steel	1699
8	Schraube	Brass	3004

Figure 3.21. *Example clipboard table.*

If we want to generate an extern data table, we should generate a table like in the example of figure 3.20, in the case of CSV format, with the same structure or a table in a spreadsheet and copy it in the case of the clipboard table addition, as in figure 3.21. Both options can be found in the same menu.

Once the tables are embedded we can manipulate them and give format to them, inserting colour, fonts or other options. In figure 3.22 we can see how the result will be after inserting the corresponding tables in the document. After that, we will insert a carrousel, in which we will be able to select the views of the model and change them just pressing the buttons. The final result of the document is shown in figure 3.23.

There are sometimes, although, difficulties to generate a table because the file has not charged correctly the model tree. In this case we can fix the problem importing and exporting the STEP file with another CAD/CAM software as NX Siemens, so the model tree will be correctly generated and the table perfectly implemented in our document.


OTH
OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
LFW
LABOR FERTIGUNGSTECHNIK UND WERKZEUGMASCHINEN
Model 3D:
Views:

Name of the model: Assembly 1 Author: Victor Arnedo Blanco Date: 24.06.2016

Table of content:

#	NAME	QTY
1	Mutter001	1
1.1	mesh	1
2	Mutter002	1
2.1	mesh (1)	1
3	Mutter	1
3.1	mesh (2)	1
4	Erhoehung002	1
4.1	mesh (3)	1
5	Erhoehung001	1
5.1	mesh (4)	1
6	Unterlegscheibe001	1
6.1	mesh (5)	1
7	Erhoehung	1
7.1	mesh (6)	1
8	Schraube	1
8.1	mesh (7)	1
9	Spannpratze	1

Operator annotations:

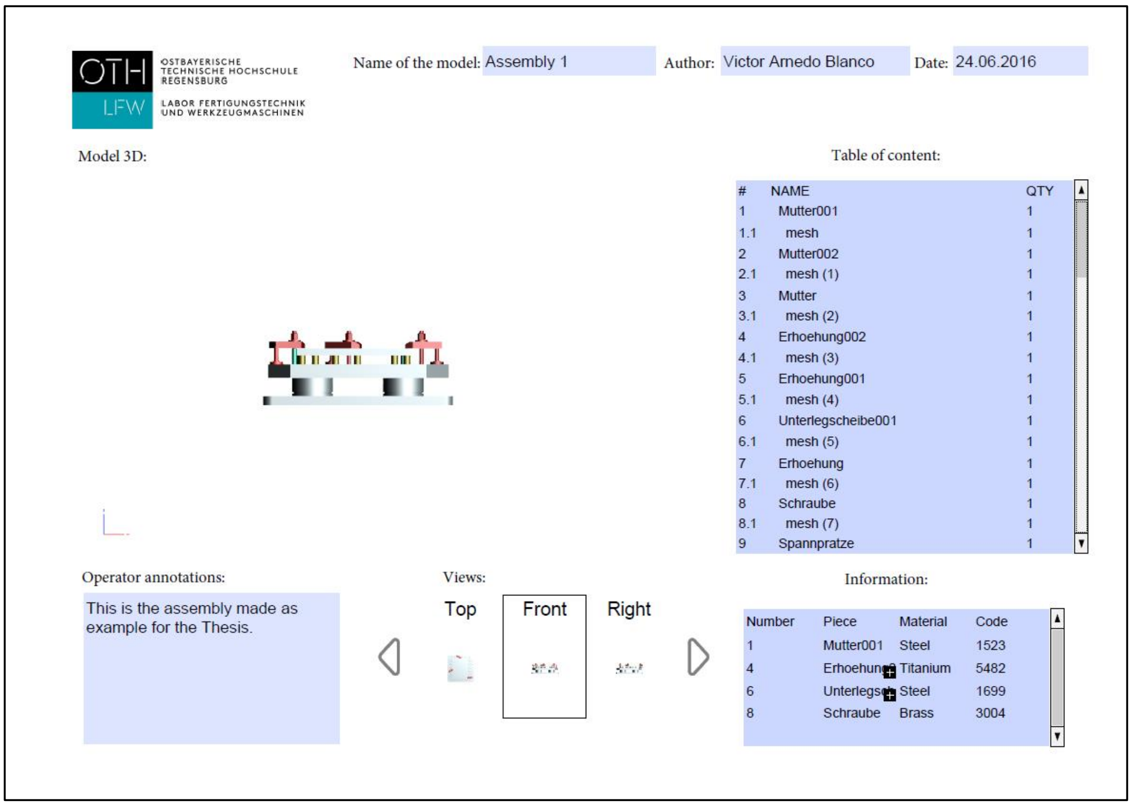
This is the assembly made as example for the Thesis.

Information:

Number	Piece	Material	Code
1	Mutter001	Steel	1523
4	Erhoehung	Titanium	5482
6	Unterlegsch	Steel	1699
8	Schraube	Brass	3004

Figure 3.22. *Template with information table inserted.*

All and all, this would be an example of document we can use for our purposes but, if wished, there is the possibility to insert more elements, such as buttons or 3D attributes to the model. Anyway, with this example we will have spent just a few minutes, since he template could be used for more than one assembly, and generating the elements is quite simple and fast to learn and implement.



OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
LABOR FERTIGUNGSTECHNIK UND WERKZEUGMASCHINEN

Name of the model: Assembly 1 Author: Victor Arnedo Blanco Date: 24.06.2016

Model 3D:

Table of content:

#	NAME	QTY
1	Mutter001	1
1.1	mesh	1
2	Mutter002	1
2.1	mesh (1)	1
3	Mutter	1
3.1	mesh (2)	1
4	Erhoehung002	1
4.1	mesh (3)	1
5	Erhoehung001	1
5.1	mesh (4)	1
6	Unterlegscheibe001	1
6.1	mesh (5)	1
7	Erhoehung	1
7.1	mesh (6)	1
8	Schraube	1
8.1	mesh (7)	1
9	Spannpratze	1

Operator annotations:

This is the assembly made as example for the Thesis.

Views:

Information:

Number	Piece	Material	Code
1	Mutter001	Steel	1523
4	Erhoehung	Titanium	5482
6	Unterlegsch	Steel	1699
8	Schraube	Brass	3004

Figure 3.23. *Final document after carrousel implementation.*

3.3. Creation of a 3d pdf document with 3d tool and latex

3D Tool and LaTeX will provide us with the necessaries mechanisms for converting an STEP model into an U3D file and create interactive 3D PDF documents. It has been the solution in which we have been focused because its necessity to write some code when we want to change the 3D model to embed in the template, but the final document will include helpful elements that will make the comprehension of the assembly to work a very easy task apart from the 3D model view and the model tree, that will allow us also to highlight the selected part on it. For this solution, the software needed will be:

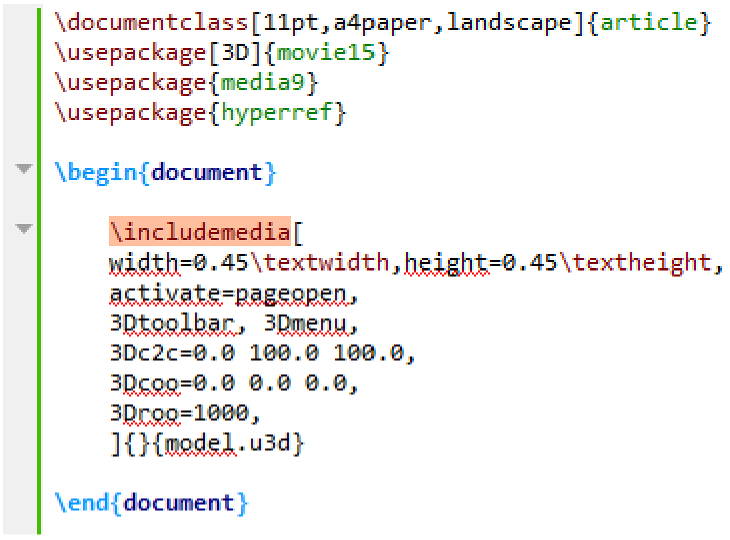
- 3D Tool Advanced
- LaTeX's TeXstudio

3.3.1. Generation of set-up document and embedding of the 3d model

The generation of the 3D PDF document has been different in this case. The way the design has been done is based in the trial and error method and started with the inclusion of the U3D file in the LaTeX document.

First of all, we had to convert the STEP model into U3D file. For this purpose, we used the software 3D Tool. The only problem this software has is that it is not capable of converting the PMI data, so it will not be available for us in order to include it in the 3D PDF document. Knowing this, we create the U3D file and we save it in the same folder in which we will save all the programs we will create with LaTeX.

Once done this, we should start working with the LaTeX environment in TeXstudio. LaTeX is a language based on libraries, so if we want to implement some objects or different features we will have to call this libraries including packages at the beginning of the main program. Our document will be composed of a main program that will import all the tables and information from other complementary smaller programs and the elements will be implemented manually. After that, our goal should be try to import all the necessary information automatically from the CAD/CAM software and visualize it in the document. The first thing to do is import the 3D model from the U3D file with the package movie15 or media9. In this case we will include both. Then, as always in LaTeX, we have to create a new document, giving it a format. For this, we will use the command `\documentclass` with which we will establish the size of the font, the sheet and the kind of document, in this case an article. After that we write the `\begin{document}` command, necessary every time we want to create a PDF file and include between this command and the `\end{document}` one the code for implementing the 3D model, basic for starting with the creation of our template and work basis as well. We can see the example code of our first program in figure 3.24.



```
\documentclass[11pt,a4paper,landscape]{article}
\usepackage[3D]{movie15}
\usepackage{media9}
\usepackage{hyperref}

\begin{document}

    \includemedia[
    width=0.45\textwidth,height=0.45\textheight,
    activate=pageopen,
    3Dtoolbar, 3Dmenu,
    3Dc2c=0.0 100.0 100.0,
    3Dcoo=0.0 0.0 0.0,
    3Droo=1000,
    ]{}{model.u3d}

\end{document}
```

Figure 3.24. *Code for embedding 3D model.*

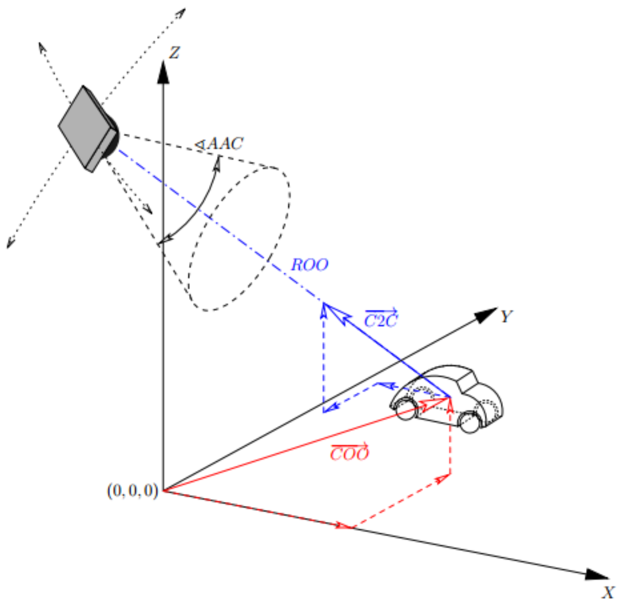


Figure 3.25. *Graphic of the camera and object in a coordinates XYZ system [31].*

In this example we can see we have included another package as `hyperref`, necessary for the inclusion of most of the features we will utilize for make the document more interactive and intuitive. We use the `includemedia` command for the embedding of the 3D model, and inside the keys we set all the options for the visualization of it, from the width and height of the box in which it will be include, to the center point of the orbit, the center of the camera vector and the orbital radius of the viewer with `3Dcoo`, `3Dc2c` and `3Droo` respectively. We can see it graphically in figure 3.25. We also activate the toolbar, the menu and finally add the 3D model with the name model.u3d in order that the designer can change the assembly to publish just saving the new one with this name and changing the old one in the corresponding folder.

3.3.2. Embedding further required information into the document

With this initial process done, we will obtain a simple PDF document in which we will have the 3D model and we will be able to interact with it as in a CAD/CAM viewer software.

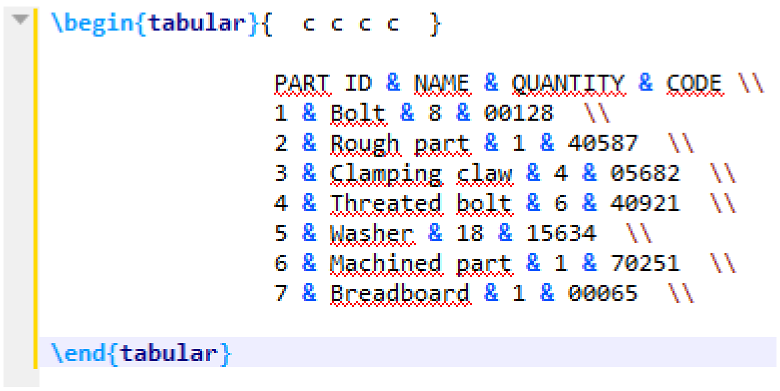
Before starting programming, the template, a structure of it has to be designed. Knowing where to collocate each element is necessary to ease the work and do it in an organized way. The interactive aspects we want to add to our example template are basically:

- Name of the model

- Author
- Table of the assembly's components
- Further information and annotations box
- Operator comments or feedback box

Working with all this kind of items results difficult in LaTeX, above all positioning them in the document. For this issue, the solution though has been base the structure in a table, so each component will be located in a specific cell and they will not superimpose one over the other.

The implementation of a table requires `\begin{tabular}` and for our type of structure. We can also `\begin{center}` to set the collocation. The attributes to include will be the position of the element in each column. This only can be done testing in which place fits better everything. We have to include the first letter of the side we want them to be in each column like `{l c}` for left and center, for instance. If we want to include elements on it, we should separate columns with the symbol "`&`" and rows with "`\\`", being possible, also, to give a specific height to the row writing `[20 ex]` at the end of each line. The number of the example can be changed depending on our necessities. One of the elements implemented in our table will be, as said before, a table of the 3D model components. This table will be created in another program like in figure 3.26, so we will need to use the package `inputenc` for the importing of elements from other programs. All the elements including external data will be imported, so the command `\input` will be used more than once in our main code.



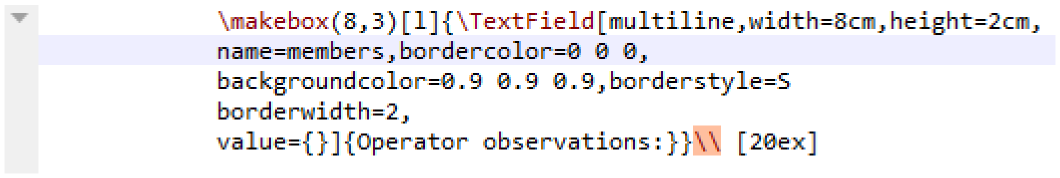
```
\begin{tabular}{  c c c c  }

                PART ID & NAME & QUANTITY & CODE \\
                1 & Bolt & 8 & 00128  \\
                2 & Rough part & 1 & 40587  \\
                3 & Clamping claw & 4 & 05682  \\
                4 & Threated bolt & 6 & 40921  \\
                5 & Washer & 18 & 15634  \\
                6 & Machined part & 1 & 70251  \\
                7 & Breadboard & 1 & 00065  \\

\end{tabular}
```

Figure 3.26. *Code for a table.*

When we create this table, we have to save it in the same folder as the main code and use the command giving as attribute the name of the file, like `\input{table_test}` as we will see in our main program.

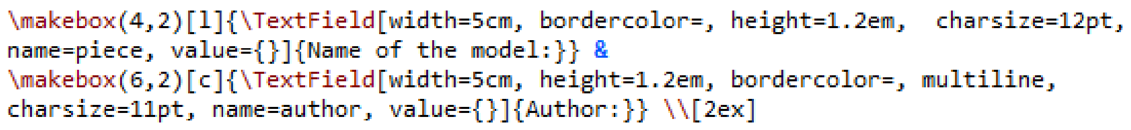
Another element we want to implement is a text field with which the operator will be able to annotate whatever he considers necessary. To make it real, we will need the command `\TextField` and set its options up. We can establish its height, weight, the size of the font, the background colour, the value or the name of the text field, an important fact in future elements. In this case, we also give it the possibility of having a multiline format and a border, as well as include it inside a `\makebox`, that will provide the field a specific space so it will not be modified by the changes of other elements in the table. In figure 3.27 we can see the code of this text field. The make box attributes are the weight, height and the position of the element inside. The text field, furthermore, can have a title, determined at the end of the declaration in between keys.

Another element that could be implemented with text field would be the name and the author. We will insert them also in `\makebox` and collocate them on the top of the document. Figure 3.28 shows us both parts of the code.



```
\makebox(8,3)[l]{\TextField[multiline,width=8cm,height=2cm,
name=members,bordercolor=0 0 0,
backgroundcolor=0.9 0.9 0.9,borderstyle=S
borderwidth=2,
value={}]{Operator observations:}}\\ [20ex]
```

Figure 3.27. *Code for the Operator annotations.*

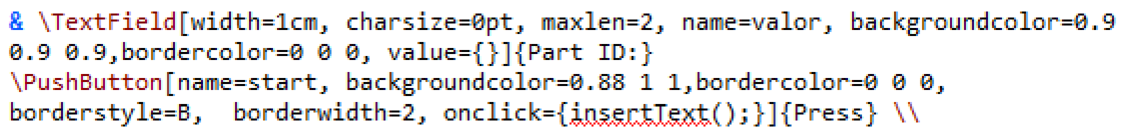


```
\makebox(4,2)[l]{\TextField[width=5cm, bordercolor=, height=1.2em,  charsize=12pt,
name=piece, value={}]{Name of the model:}} &
\makebox(6,2)[c]{\TextField[width=5cm, height=1.2em, bordercolor=, multiline,
charsize=11pt, name=author, value={}]{Author:}} \\[2ex]
```

Figure 3.28. *Code for name of the model and author.*

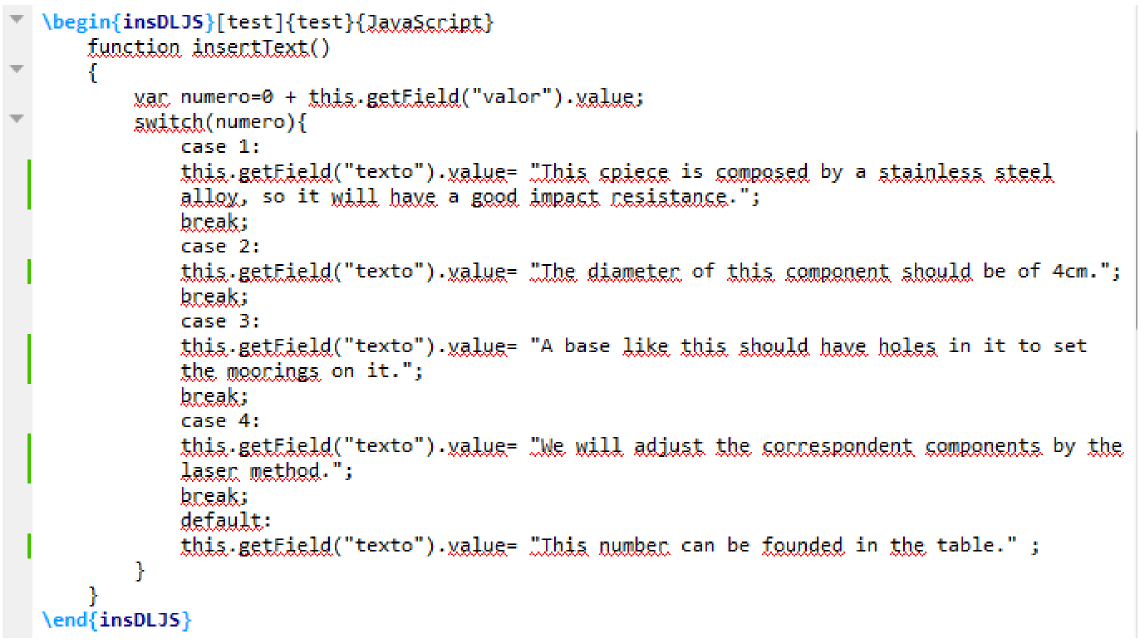
The setting of all the options can be consulted in the libraries documents. A lot of information is available in the Internet about it and it does not suppose a huge effort to find it out.

The last thing it will be implemented is the information section. As the name indicates, in this table we will create there will be all the data the CAD software can give us about a part of the assembly. The idea is to select one of the components writing down its number and the table will return all the information available about it. The code will be divided in three parts, seen in figures 3.29, 3.30 and 3.31.



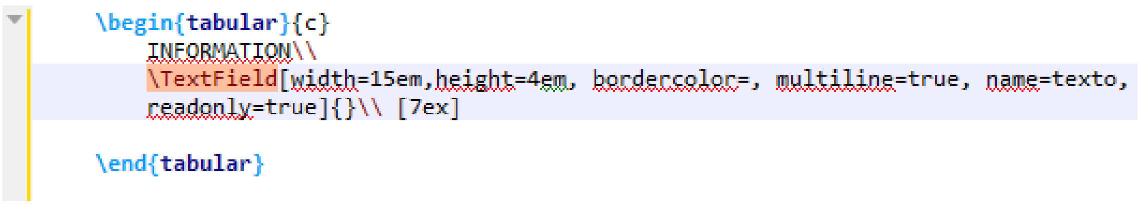
```
& \TextField[width=1cm, charsize=0pt, maxlen=2, name=valor, backgroundcolor=0.9
0.9 0.9,bordercolor=0 0 0, value={}]{Part ID:}
\PushButton[name=start, backgroundcolor=0.88 1 1,bordercolor=0 0 0,
borderstyle=B,  borderwidth=2, onclick={insertText();}]{Press} \\
```

Figure 3.29. *Code for text field and button for digits of component.*



```
\begin{insDLJS}[test]{test}{JavaScript}
    function insertText()
    {
        var numero=0 + this.getField("valor").value;
        switch(numero){
            case 1:
            this.getField("texto").value= "This cpiece is composed by a stainless steel
            alloy, so it will have a good impact resistance.";
            break;
            case 2:
            this.getField("texto").value= "The diameter of this component should be of 4cm.";
            break;
            case 3:
            this.getField("texto").value= "A base like this should have holes in it to set
            the moorings on it.";
            break;
            case 4:
            this.getField("texto").value= "We will adjust the correspondent components by the
            laser method.";
            break;
            default:
            this.getField("texto").value= "This number can be founded in the table." ;
        }
    }
\end{insDLJS}
```

Figure 3.30. *Code for JavaScript function.*



```
    \begin{tabular}{c}
        INFORMATION\\
        \TextField[width=15em,height=4em, bordercolor=, multiline=true, name=texto,
        readonly=true]{}\\ [7ex]

    \end{tabular}
```

Figure 3.31. *Code for table of further information.*

It is appropriated to explain the important aspects of the code embedded in this part. The first thing to implement is a system, seen in figure 3.29, with which the program is capable to identify the part of the assembly we require the information. It is done with a text field in which the operator has to introduce a two digits, limited by the option `maxlen`, and we will call it Part ID.

After that, we will implement one of the interactive elements LaTeX offers us, a button with \PushButton. This button has many ways to be used, but we will utilize the simple one, pressing it. The attributes are similar to a text field, giving it colour or border as well, and at the end we implement a function in it and give a name. The function has to be declared before the beginning of the document and it will be called `insertText`.

In figure 3.30 we can see the code of the function used in the button. It is based on JavaScript, as we can see, and it is started with the command `\begin{insDLJS}`, whose attributes include the function, the base name of it, and the script name. Inside it, we can observe that a variable is declared as a number. The value of the number, obtained from the one written in the text field by the operator, will be caught with `this.getField("valor").value`. The name in between the parenthesis is the name of the text field used for the number of the component, as we can check in figure 3.29, and it will be used for the switch below this declaration, setting the information pertaining to every component. As noticed in the image of the code, there are four cases used as example, all finished with a `brake` order but it can be modified whenever we want with the information wished. The only thing we have to do is write in between the quotation marks the text we want and add, if necessary, more cases in their corresponding number. Compulsorily, we have to write a default option in which we can write a message such as "This number can be founded in the table" to make know the operator that the digits noted are wrong. To get the information we will utilize again `this.getField` and `value`, but this time we will give it the name of the text field used to show the facts, in figure 3.31. In this case, the name will be `texto` and the only differences respect the other text field will be the inclusion of an option called `readonly`, to not be able to modify the information in the PDF document, and its insertion in a table, that will provide a better format with a tittle above the field. This text field and table will be written in another program and included with `\input` in the main program.

Once all the elements are established, we will have to structure them into the document using, as said before, a table of elements. The thought format has been similar to the one shown in figure 3.32. To design the table, we have based our code on the example shown before but improving it with the `multirow` package. This library allows us to use more than one row in a specific column while in another one there is just one. That means that, while in the left column there is only the 3D model situated, we can position the table of content and the further information table with the same height occupied by the 3D model but in the right column.

We will leave a little space at the bottom of the right column so it is possible to incorporate new features that the designer considers necessary or suitable. We can see the code for the main table in figure 3.33.

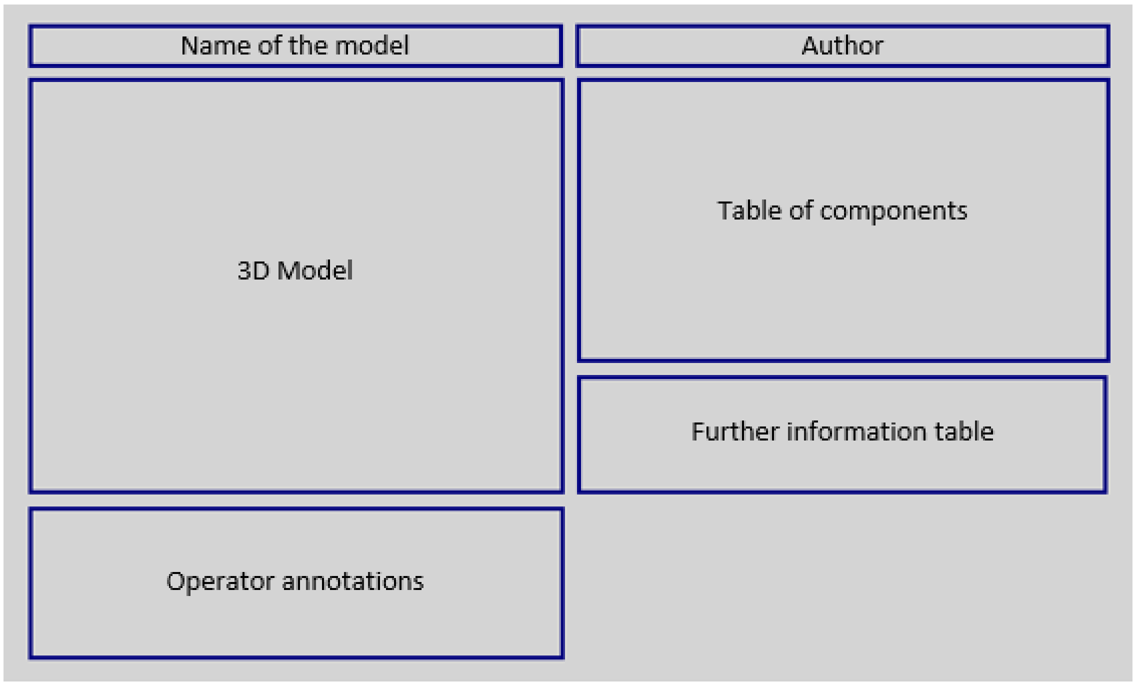
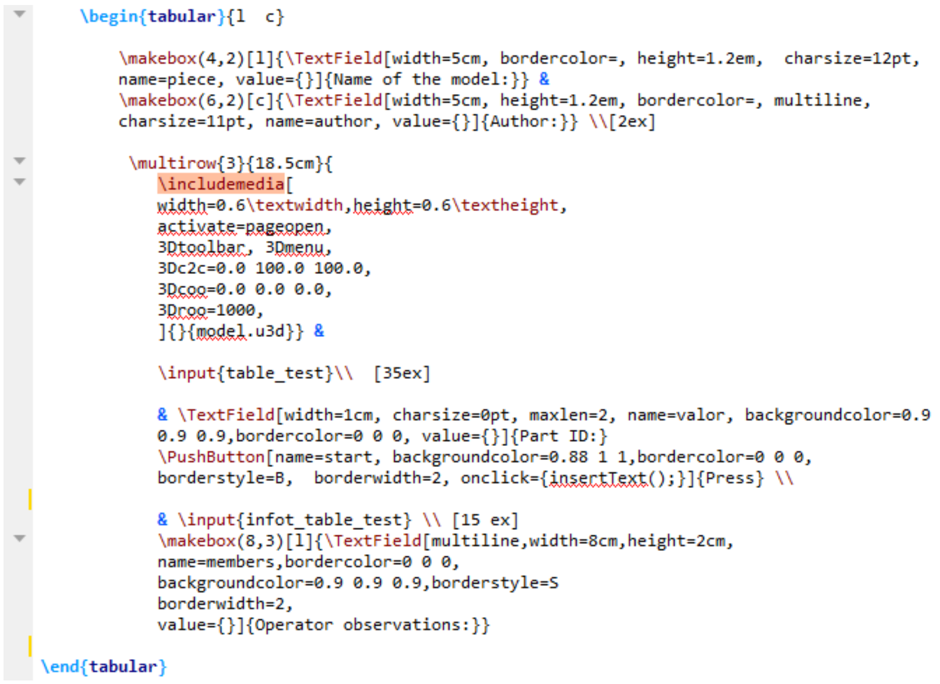


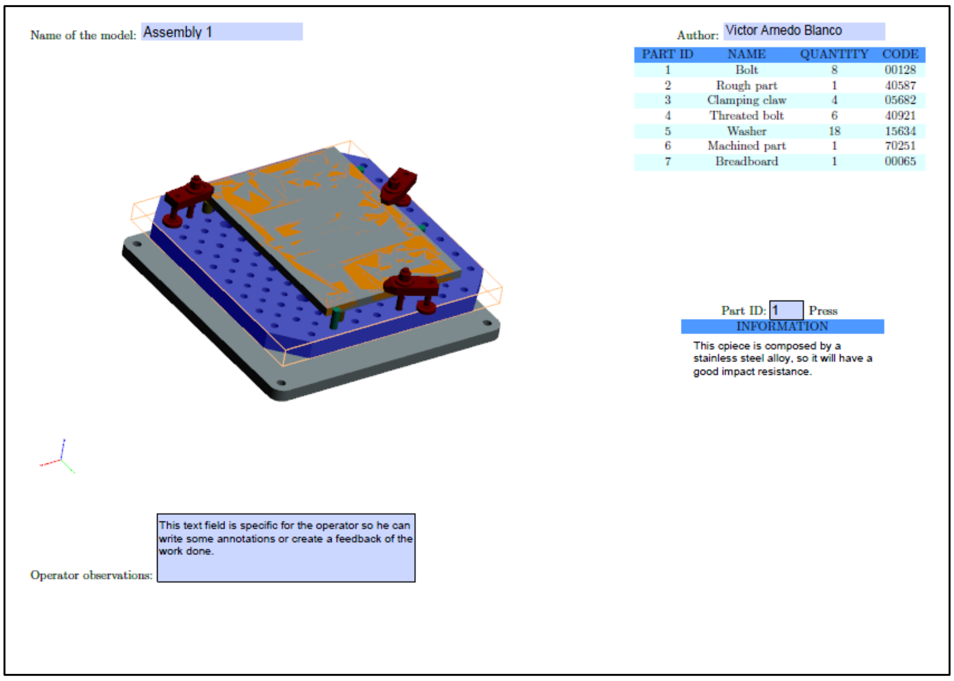
Figure 3.32. *Structure of the document.*

In the code it is noticed that the `multirow` includes three rows in the right column. This is due to the fact that the button and text field for the lecture of the component number are positioned above the table of further information, so we structure that in two different rows to make it clearer. We can see the imported files as well in the program. All and all, the PDF document will look like in figure 3.34.



```
    \begin{tabular}{l  c}

        \makebox(4,2)[l]{\TextField[width=5cm, bordercolor=, height=1.2em,  charsize=12pt,
        name=piece, value={}]{Name of the model:}} &
        \makebox(6,2)[c]{\TextField[width=5cm, height=1.2em, bordercolor=, multiline,
        charsize=11pt, name=author, value={}]{Author:}} \\[2ex]

         \multirow{3}{18.5cm}{
            \includemedia[
            width=0.6\textwidth,height=0.6\textheight,
            activate=pageopen,
            3Dtoolbar, 3Dmenu,
            3Dc2c=0.0 100.0 100.0,
            3Dcoo=0.0 0.0 0.0,
            3Droo=1000,
            ]{}{model.u3d}} &

            \input{table_test}\\  [35ex]

            & \TextField[width=1cm, charsize=0pt, maxlen=2, name=valor, backgroundcolor=0.9
            0.9 0.9,bordercolor=0 0 0, value={}]{Part ID:}
            \PushButton[name=start, backgroundcolor=0.88 1 1,bordercolor=0 0 0,
            borderstyle=B,  borderwidth=2, onclick={insertText();}]{Press} \\

            & \input{infot_table_test} \\ [15 ex]
            \makebox(8,3)[l]{\TextField[multiline,width=8cm,height=2cm,
            name=members,bordercolor=0 0 0,
            backgroundcolor=0.9 0.9 0.9,borderstyle=S
            borderwidth=2,
            value={}]{Operator observations:}}

\end{tabular}
```

Figure 3.33. *Code for the JavaScript function.*



Name of the model: Assembly 1

Author: Victor Amedo Blanco

PART ID	NAME	QUANTITY	CODE
1	Bolt	8	00128
2	Rough part	1	40587
3	Clamping claw	4	05682
4	Threated bolt	6	40921
5	Washer	18	15634
6	Machined part	1	70251
7	Breadboard	1	00065

Part ID: 1 Press

INFORMATION

This cpiece is composed by a stainless steel alloy, so it will have a good impact resistance.

Operator observations: This text field is specific for the operator so he can write some annotations or create a feedback of the work done.

Figure 3.34. *Final manually implemented 3D PDF document.*

As we can see in the figure above, the tables have been formatted, colouring its rows. This is possible with the packages `xcolor` and `colortbl` and defining colours like in figure 3.35, where we can see they are defined with the RGB code between 0 and 1. We can find all the combinations in the Internet. After that, we will have to implement the command `\rowcolor{colour}` with the already defined colour.

Other aspects of the code to comment are the use of other packages. It has been implemented the `geometry` package, which provides us with some options for the management of the margins and also the `array` package, for the formatting of the elements inside the tables.

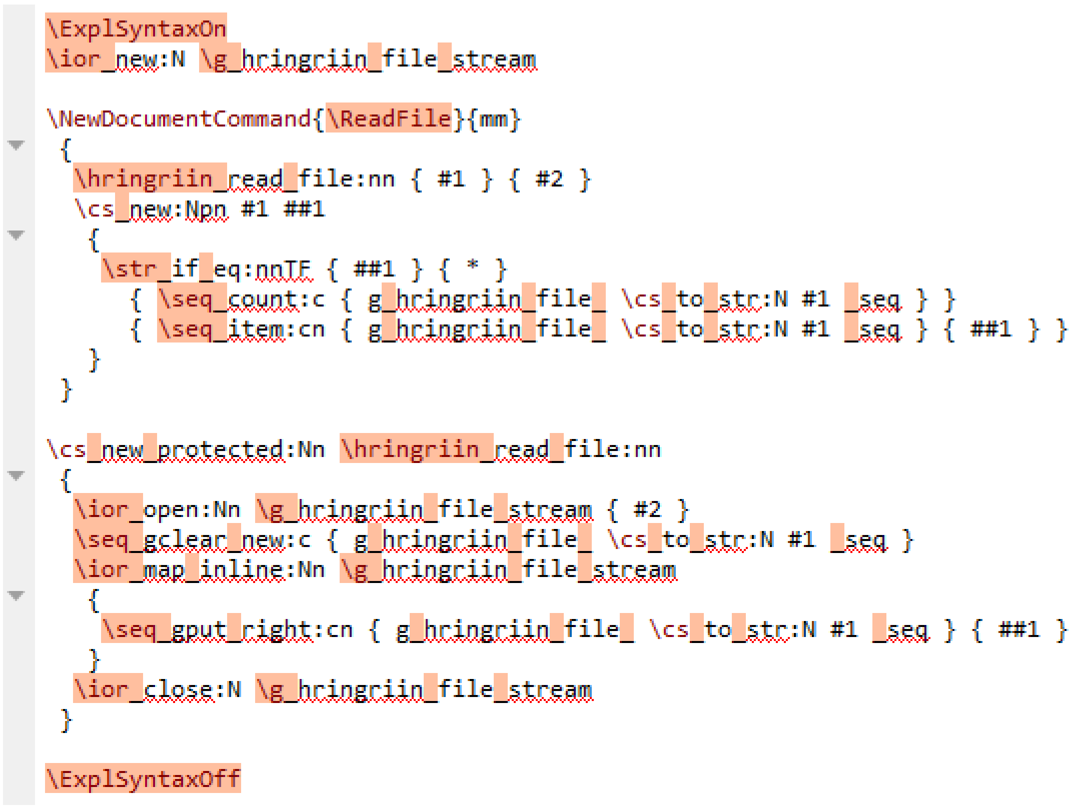
```
\definecolor{white}{rgb}{1,1,1}
\definecolor{Blue1}{rgb}{0.3,0.6,1}
\definecolor{Blue2}{rgb}{0.88,1,1}
```

Figure 3.35. *Code for the declaration of the colours.*

3.3.3. *Possible improvements*

One of the aims of the thesis, as explained before, is to find or create a fast and efficient workflow between the designer and the operator. In this section, the different possibilities to make this purpose real will be explained, as well as possible ways to continue the research in this field.

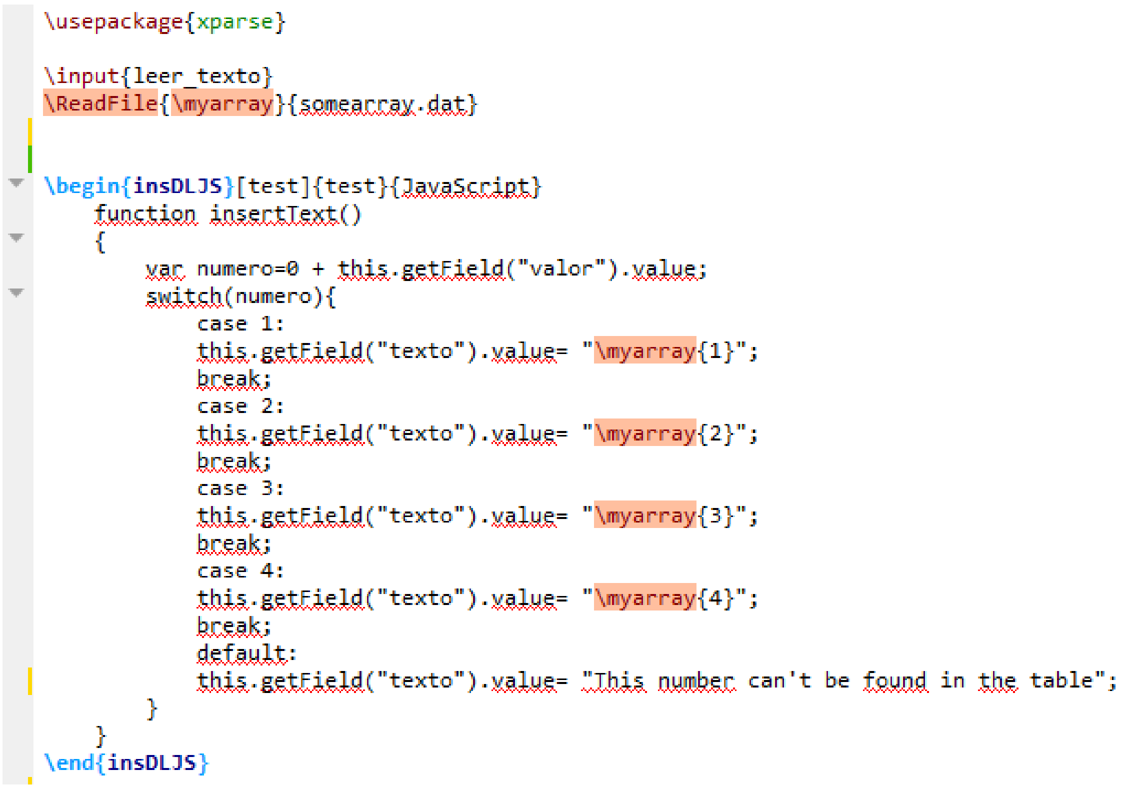
A little betterment that could be implemented fall to the further information table. In the example shown in the section above, the data corresponding to each component was directly written in the main program. We can, actually, import this information from an external DAT file made from the CAD software, only needing the required format so our code is capable to interpret the text and process it in the wished way and save it in the same folder as the programs used. Searching in the web we find a code like in figure 3.36, an example that enables us to read a file and store in different arrays each row of the text. The idea would be to give the information of the specific component in its correspondent row, so, for instance, if we want to show the information of a bolt numbered as the piece 6 of the assembly, its information should be written down on the sixth row of the document.



```
\ExplSyntaxOn
\ior_new:N \g_hringriin_file_stream

\NewDocumentCommand{\ReadFile}{mm}
 {
  \hringriin_read_file:nn { #1 } { #2 }
  \cs_new:Npn #1 ##1
   {
    \str_if_eq:nnTF { ##1 } { * }
      { \seq_count:c { g_hringriin_file_ \cs_to_str:N #1 _seq } }
      { \seq_item:cn { g_hringriin_file_ \cs_to_str:N #1 _seq } { ##1 } }
   }
 }

\cs_new_protected:Nn \hringriin_read_file:nn
 {
  \ior_open:Nn \g_hringriin_file_stream { #2 }
  \seq_gclear_new:c { g_hringriin_file_ \cs_to_str:N #1 _seq }
  \ior_map_inline:Nn \g_hringriin_file_stream
   {
    \seq_gput_right:cn { g_hringriin_file_ \cs_to_str:N #1 _seq } { ##1 }
   }
  \ior_close:N \g_hringriin_file_stream
 }

\ExplSyntaxOff
```

Figure 3.36. *Code for the processing of DAT file.*

The code above is composed by very complex elements of the last LaTeX language version. The package `xparse` must be added for the good work of the function. In the main code we will call the already declared function of `\ReadFile`, that with the command `\myarray{}` and the name of the text, as `{somearray.dat}`, will read and call the arrays stored from the file. In our program, we will declare the function before the beginning of the document and the command inside the JavaScript function, in the switch. Every case will have the line `this.getField("texto").value="\myarray{1}"`, with the number inside the last command depending on the component whose information is pertaining, for sending the value of the array created to the text field of the further information table, already explained in `infot_table_test`. The commands used before are arbitraries, so we can create our own commands with the wished name. The problem all of this implies is that we will have to create as cases as pieces has the assembly, so the process is not completely automatized. However, we will earn a lot of time with this improvement if we are able to create this kind of file with our designing software. We can see the code created in figure 3.37 and the result in figure 3.38.

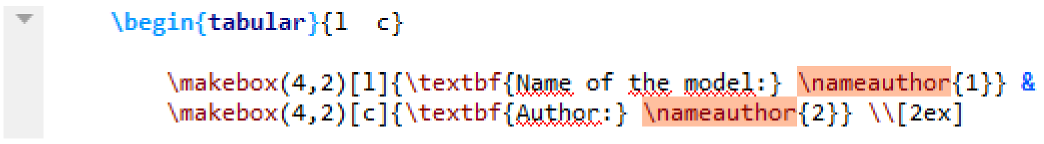


```
\usepackage{xparse}

\input{leer_texto}
\ReadFile{\myarray}{somearray.dat}


\begin{insDLJS}[test]{test}{JavaScript}
    function insertText()
    {
        var numero=0 + this.getField("valor").value;
        switch(numero){
            case 1:
            this.getField("texto").value= "\myarray{1}";
            break;
            case 2:
            this.getField("texto").value= "\myarray{2}";
            break;
            case 3:
            this.getField("texto").value= "\myarray{3}";
            break;
            case 4:
            this.getField("texto").value= "\myarray{4}";
            break;
            default:
            this.getField("texto").value= "This number can't be found in the table";
        }
    }
\end{insDLJS}
```

Figure 3.37. *Code for the improved table of information.*

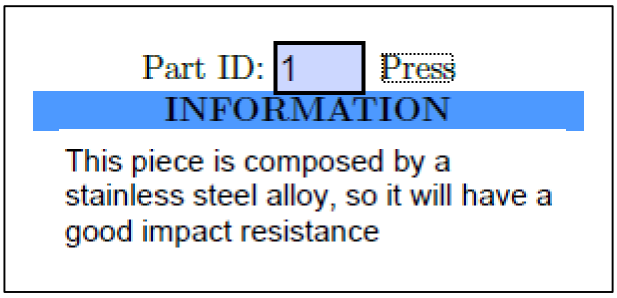
In addition, we can apply the same system for the automatization of the name of the 3D model and author implementation. We will need, though, our CAD/CAM software exporting this information in a DAT file in different rows. Once obtained the file, and saved in the same folder we have to write a similar code line than for the last solution, but changing `\myarray` by another name as `\nameauthor`, so the array stored in that variable will be called every time we write it, and the name of the file at the end of the line.

Finally, we will substitute the text fields used for that issue with our variables, so the elements will be implemented automatically every time we change the file with a new 3D model. This part of the code, in which we can observe the command `\textbf` that sets the letters in bold, will look like in figure 3.38 and the result is shown in image 3.39.



```
    \begin{tabular}{l  c}

        \makebox(4,2)[l]{\textbf{Name of the model:} \nameauthor{1}} &
        \makebox(4,2)[c]{\textbf{Author:} \nameauthor{2}} \\[2ex]
```

Figure 3.38. *Code for the automatic name of the model and author.*

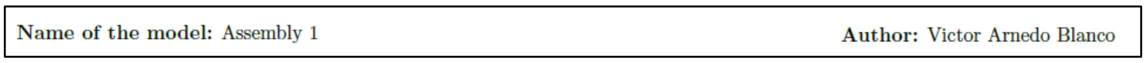


Part ID: 1 Press

INFORMATION

This piece is composed by a stainless steel alloy, so it will have a good impact resistance

Figure 3.39. *View of the improved table of information.*



Name of the model: Assembly 1 **Author:** Victor Arnedo Blanco

Figure 3.40. *Result of the automatic name of the model and author.*

At the end we will have a similar template as in figure 3.34, but improved and more efficient. There is, however, some more things that can be changed so all the information is reproduced automatically in our document. And the way to do it would be following the steps done in this issue, with the arrays systems. One of the solutions could be improved may be the automatic read of all the cases for the information table, so the designer does not have to write as cases as pieces the assembly has in the switch of the JavaScript function. Implementing a loop in and comparing the number got in the text field with the array wished and show it might be the goal in this situation.

In the same way would be the betterment of the creation of the components table. The problem here is that the format explained before for the table has to be written in the code, so we cannot just import and insert it as an array variable but we have to write in the program. However, the method might be quite similar to the used for the information table, reading an extern file and process it so the program can show the data automatically, with loops to give more flexibility to the account of elements in it. The way to continue is already started, and more effort and work will be required to obtain a completely automatic generation of a 3D PDF document.

CHAPTER 4: EXAMPLES OF IMPLEMENTATION

In this section we will display a sequence of pictures that will show all the processes step by step graphically of how the solution is implemented, not entering in the background part, since it has been explained in the previous chapter. We will use two assemblies so the good functioning with independence of the 3D model can be demonstrated. Before starting with each solution we should have both pieces saved in STEP file from the CAM software, and they look like in figures 4.1 and 4.2.

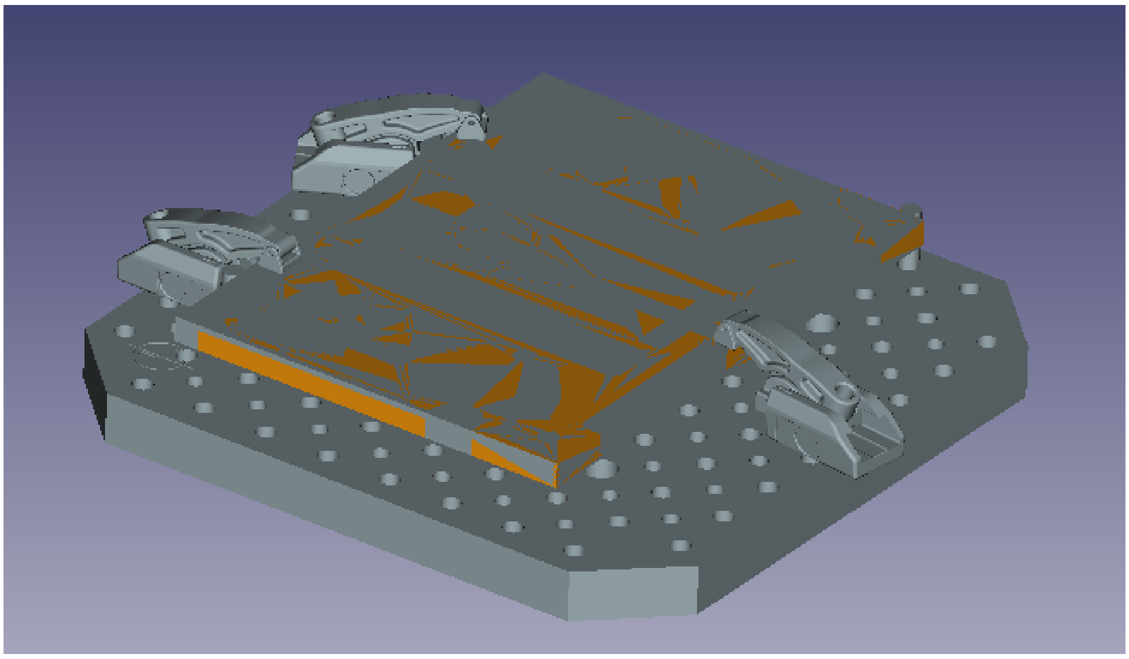


Figure 4.1. *View of the Assembly 1 in the CAM software.*

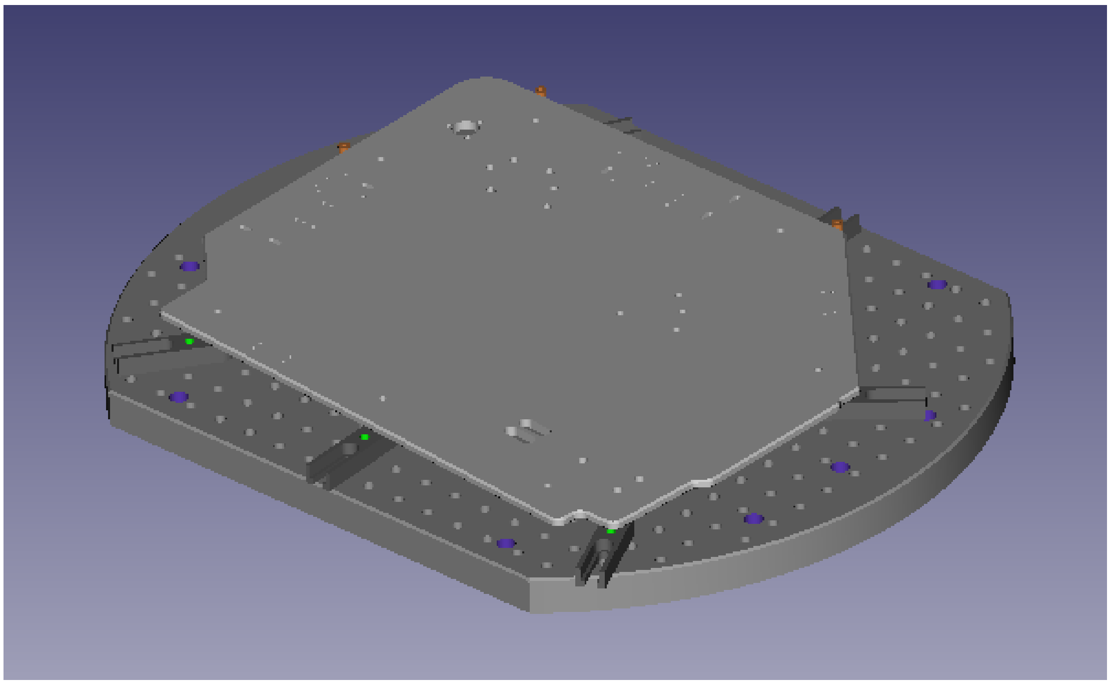
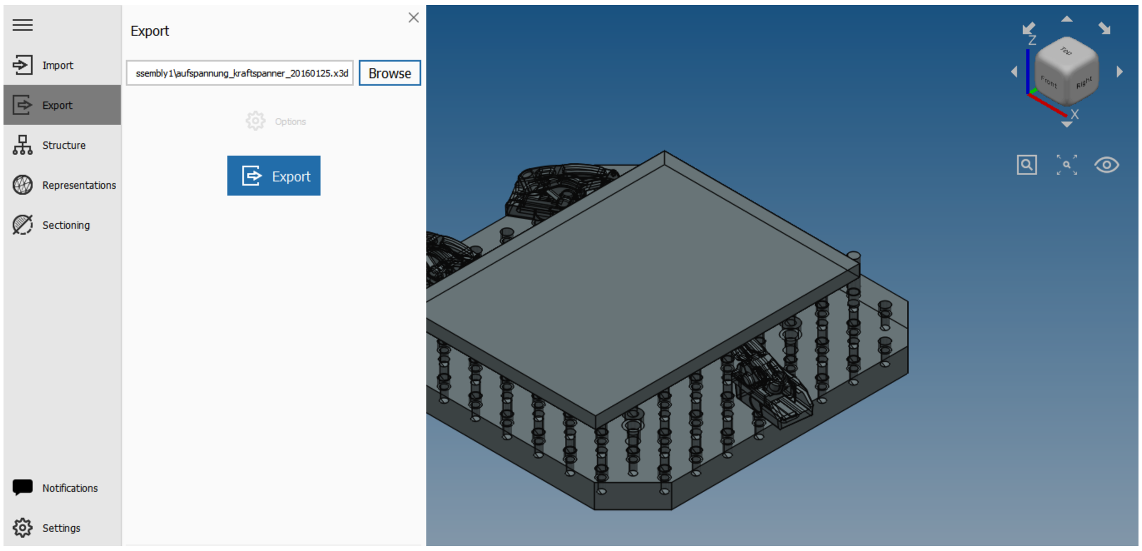


Figure 4.2. *View of the Assembly 2 in the CAM software.*

4. 1. Web Application in the browser

This is the only solution in which we will need to convert our STEP files into X3D files, so we will need the software CAD Exchanger, the chosen one because of its relation quality-price, to generate the desired files:



ssembly1\aufspannung_kraftspanner_20160125.x3d

Figure 4.3. *View of the Assembly 1 in the converter software.*

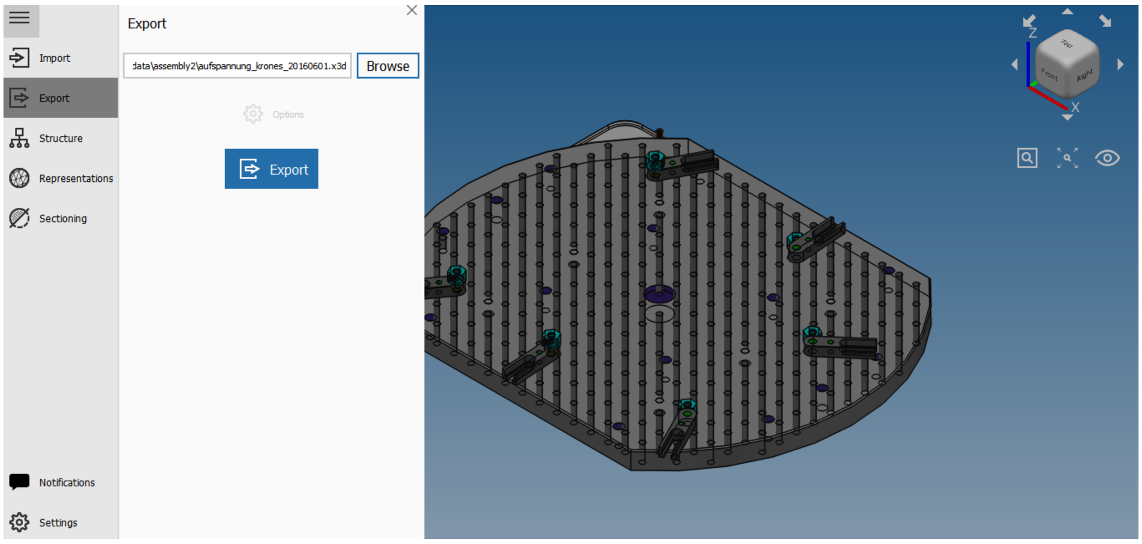


Figure 4.4. *View of the Assembly 2 in the converter software.*

After the exporting them as X3D files and send them to individual folders with each name inside the DATA folder where all the other files are, we should rename the wished assembly to model.x3d and change the name of `MYAPP.model` in the main.js file. The name we have to write is how the folder of the assembly is called. As an example, if the folder of assembly one is called in the same way, Assembly1, we should write `MYAPP.model = "Assembly1"` in main.js code and call our file model.x3d. Once everything is done, we can open the index.html file in Mozilla Firefox browser, as it is the best to support all the features of the 3D model.

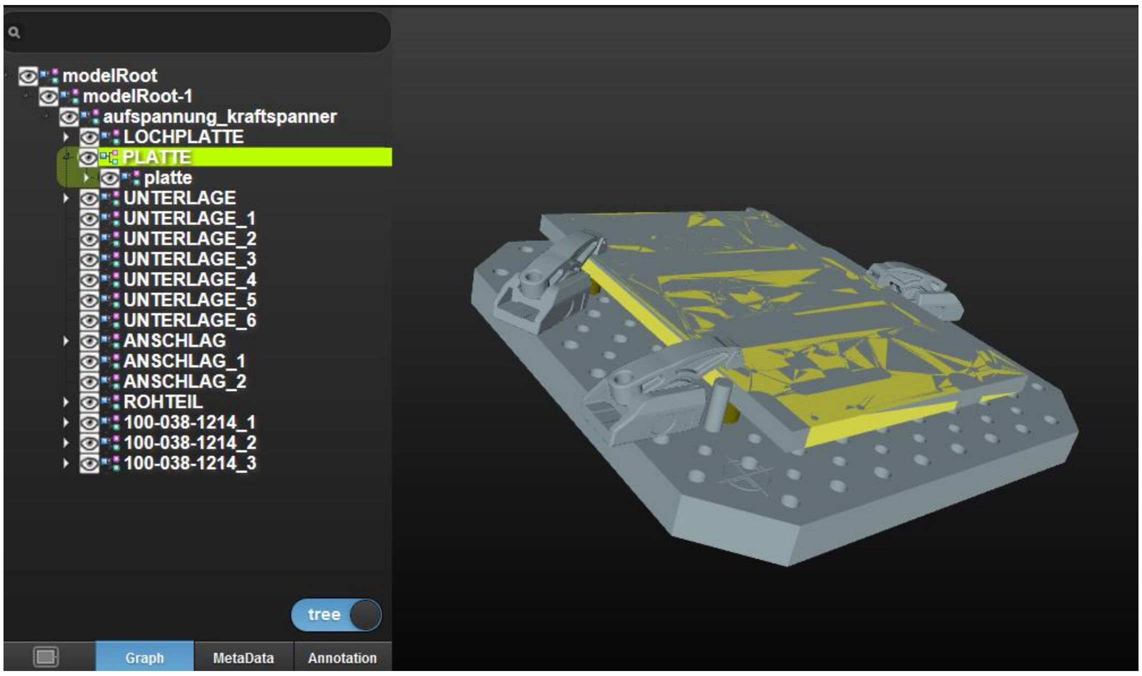


Figure 4.5. *View of the Assembly 1 in the web application.*

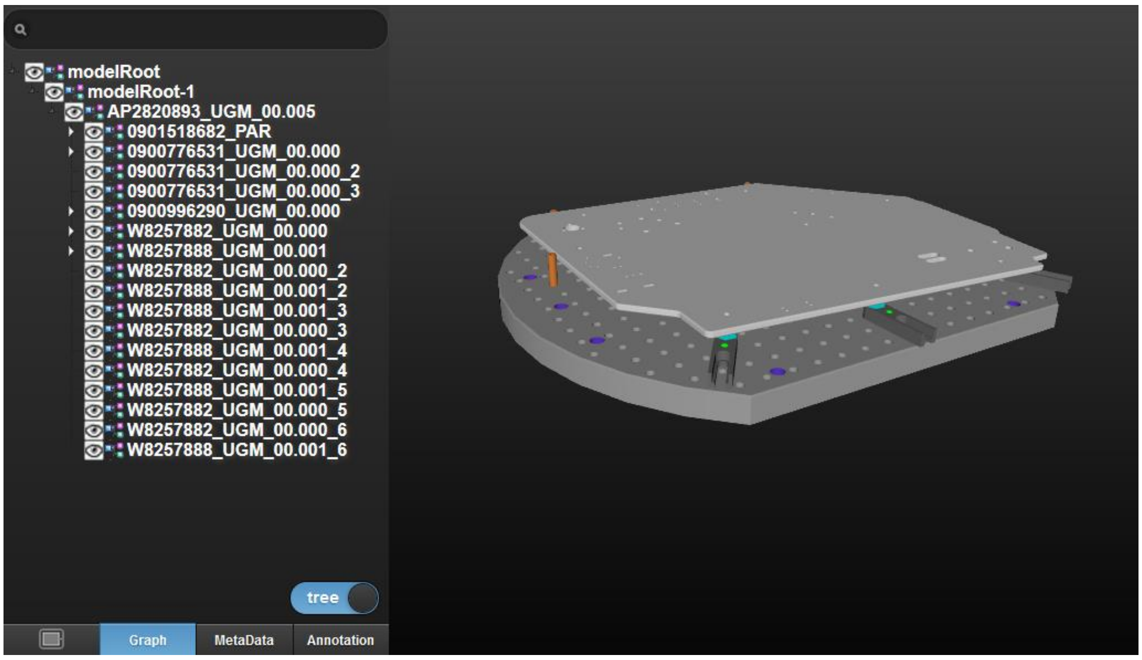


Figure 4.6. *View of the Assembly 2 in the web application.*

We also can add information in MetaData and Annotations sections, changing the notepads as shown in implementation chapter before. This is an example for Assembly 2:

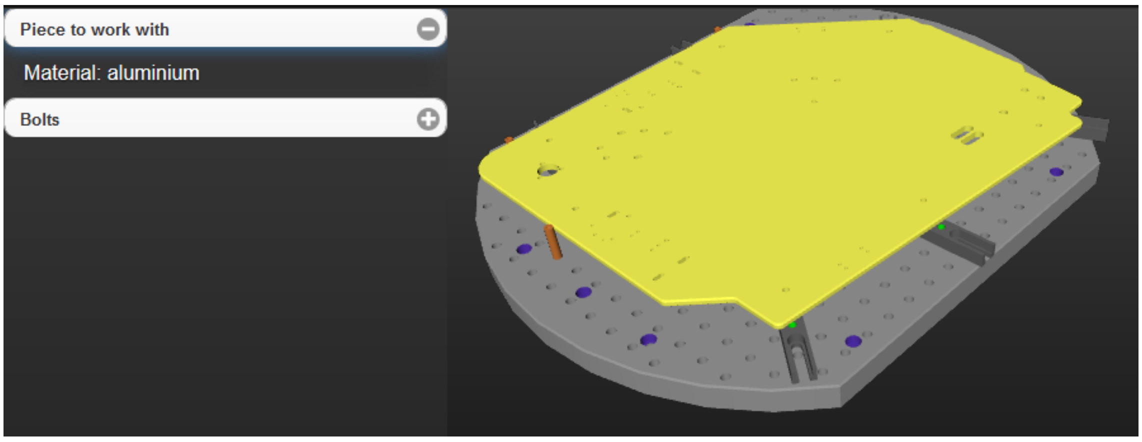
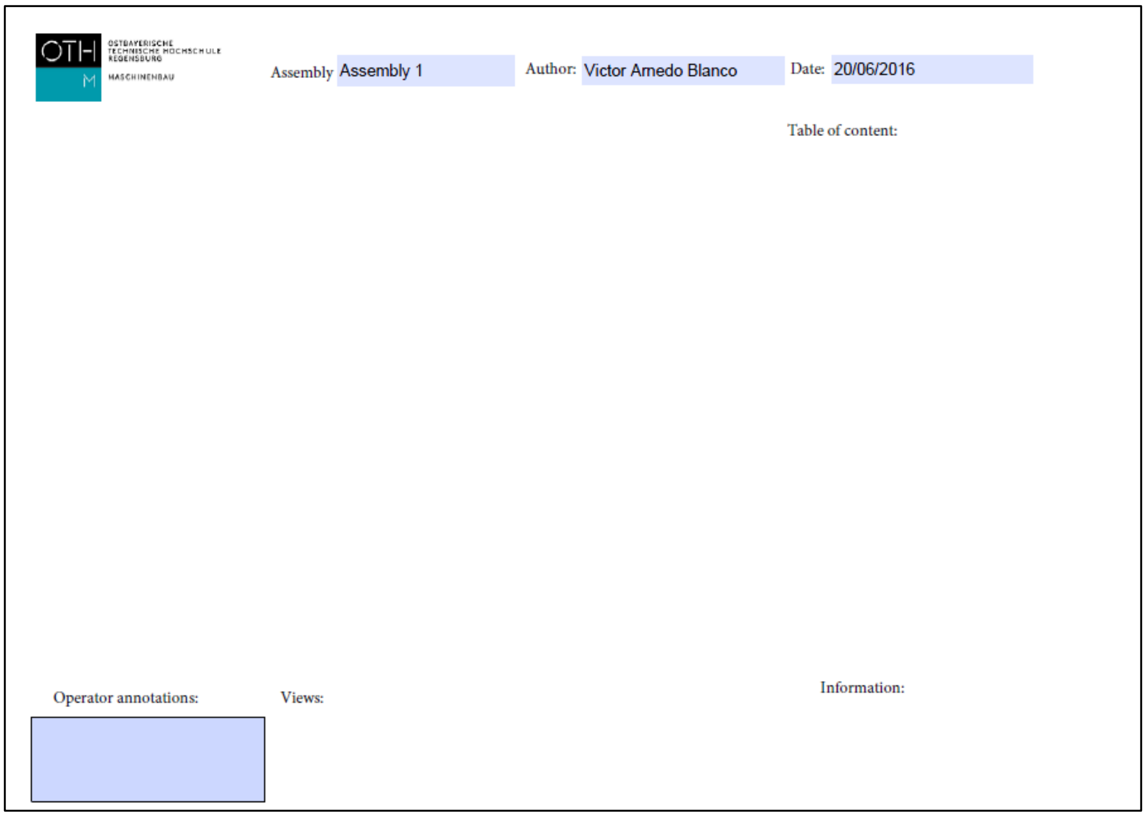


Figure 4.7. *Assembly 2 annotations.*

4.2. Tetra4D

This solution is the easiest one in terms of functionality. We only need to create a template as in figure 4.8 and add in it the interactive objects, as the 3D model, the table of components and the views of the assembly.



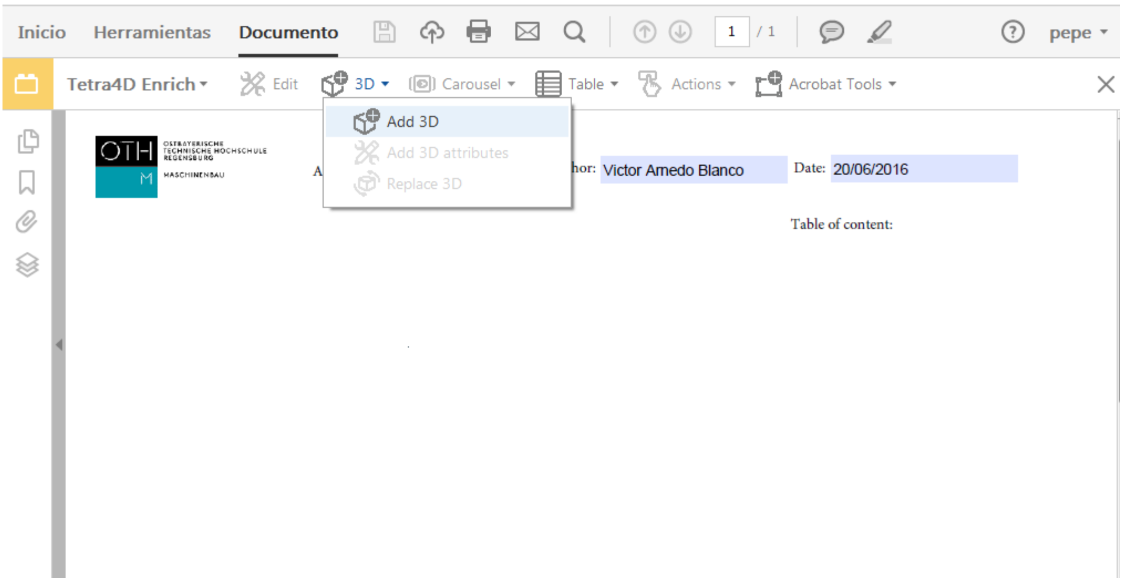
Assembly: Assembly 1
Author: Victor Amedo Blanco
Date: 20/06/2016
Table of content:
Operator annotations:
Views:
Information:

Figure 4.8. *Tetra4D created template.*

In this case, no converter is needed since Tetra4D Enrich itself englobes Tetra4D Converter and it is able to import STEP files in the PDF files, so we only need this software to carry this solution on.

It results a very fast process, with which in five minutes we can have an interactive 3D PDF document ready to share with the operator and very easy to manipulate and intuitive. We can see the process in the next figures.

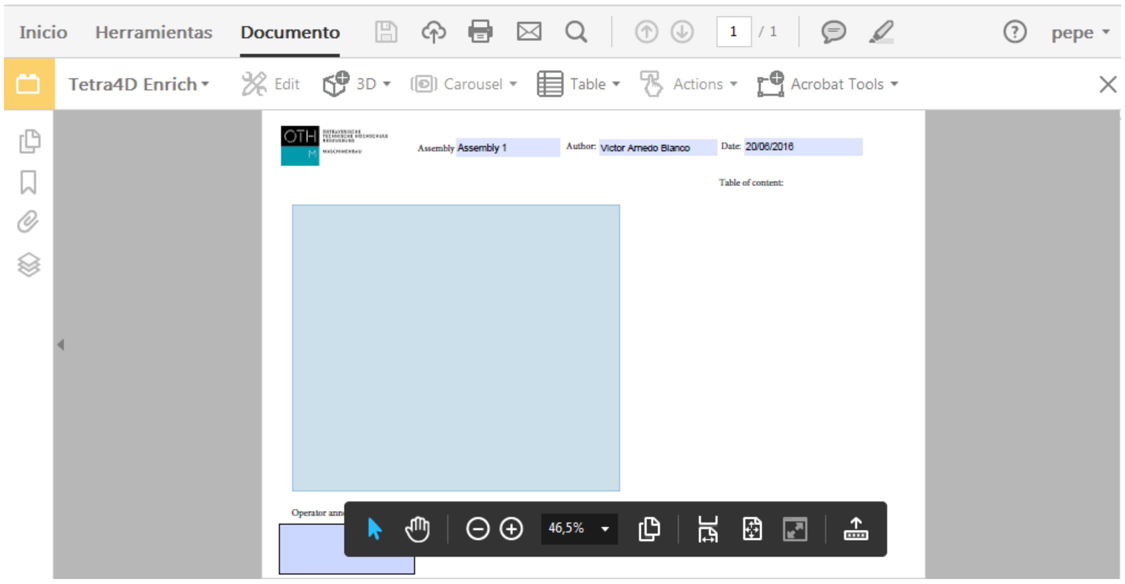
First of all, we add the 3D model in the template with the tool provided by the software, as we can see in figures 4.9 and 4.10.



hor: Victor Amedo Blanco Date: 20/06/2016

Table of content:

Figure 4.9. *Adding the 3D model on Tetra4D created template.*



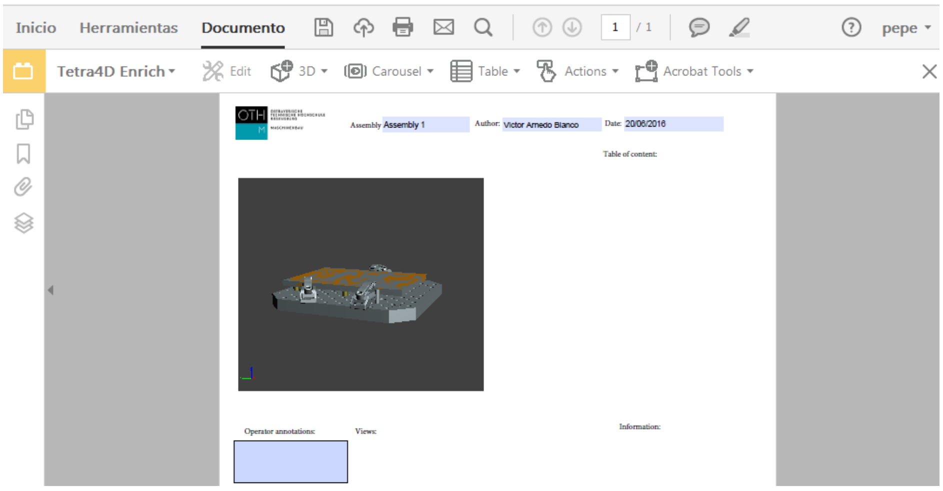
Assembly Assembly 1 Author: Victor Amedo Blanco Date: 20/06/2016

Table of content:

Operator ann

Figure 4.10. *Adding the 3D model on Tetra4D created template.*

After that, we should add the table of components, also with the tools, and the views and further information. The latest can be implemented by a CSV file or a clipboard, done in a spreadsheet for example. We can see the result of the two assemblies interactive documents in figures 4.11, 4.12 and 4.13.

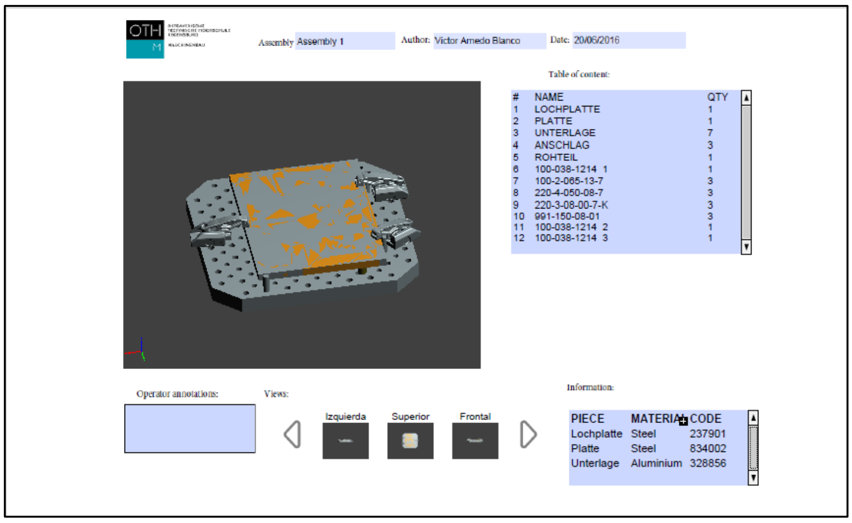


Assembly: Assembly 1 Author: Victor Amedo Blanco Date: 20/06/2016

Table of content:

Operator annotations: Views: Information:

Figure 4.11. 3D model on Tetra4D created template.



Assembly: Assembly 1 Author: Victor Amedo Blanco Date: 20/06/2016

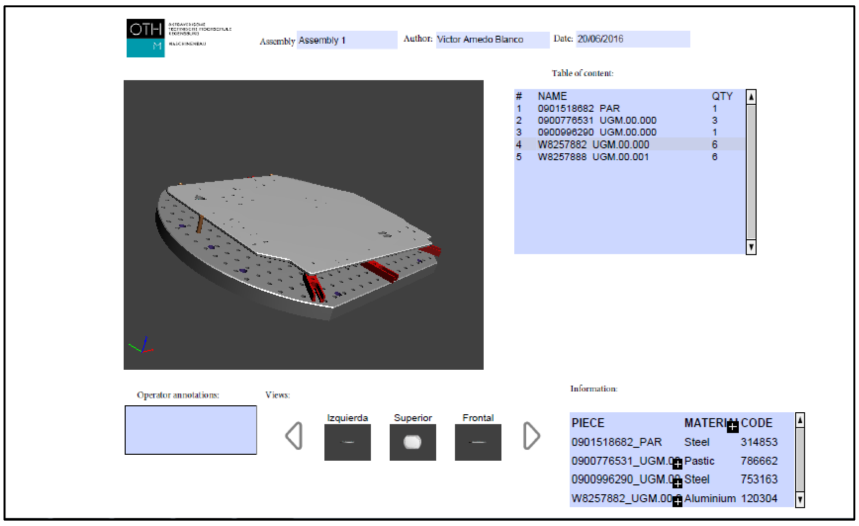
Table of content:

#	NAME	QTY
1	LOCHPLATTE	1
2	PLATTE	1
3	UNTERLAGE	7
4	ANSCHLAG	3
5	ROHTEIL	1
6	100-038-1214 1	1
7	100-2-065-13-7	3
8	220-4-050-08-7	3
9	220-3-08-00-7-K	3
10	991-150-08-01	3
11	100-038-1214 2	1
12	100-038-1214 3	1

Operator annotations: Views: Information:

PIECE	MATERIAL	CODE
Lochplatte	Steel	237901
Platte	Steel	834002
Unterlage	Aluminium	328856

Figure 4.12. Assembly 1 generated document.



Assembly: Assembly 1 Author: Victor Amedo Blanco Date: 20/06/2016

Table of content:

#	NAME	QTY
1	0901518682 PAR	1
2	0900776531 UGM.00.000	3
3	0900996290 UGM.00.000	1
4	W8257882 UGM.00.000	6
5	W8257888 UGM.00.001	6

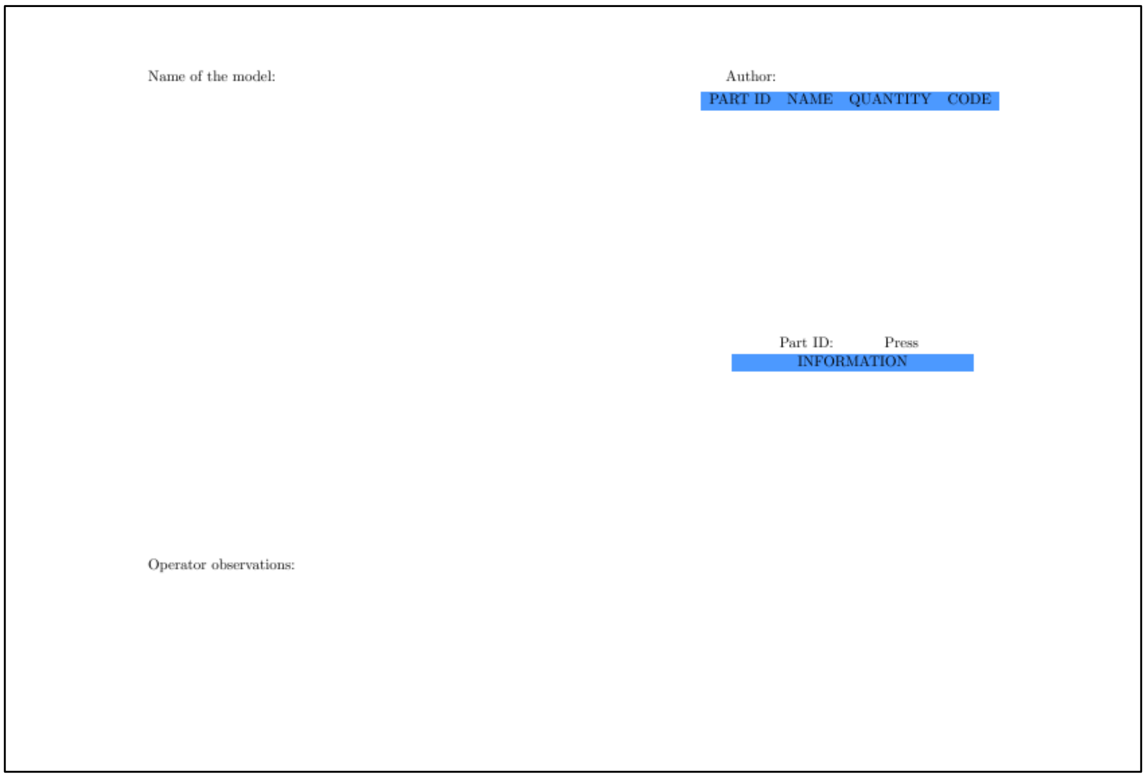
Operator annotations: Views: Information:

PIECE	MATERI	CODE
0901518682_PAR	Steel	314853
0900776531_UGM.0	Pastic	786662
0900996290_UGM.0	Steel	753163
W8257882_UGM.00	Aluminium	120304

Figure 4.13. Assembly 2 generated document.

4.3. 3D Tool and LaTeX

For this solution, two software will be necessary. First, 3D Tool will act as the STEP to U3D converter, generating the necessary file to add, coding, in the second software, LaTeX's TeXstudio. With the latest, we can create a template that can be used for our assemblies. We have designed one, and looks like figure 4.14.



Name of the model:

Author:

PART ID	NAME	QUANTITY	CODE

Part ID: Press

INFORMATION

Operator observations:

Figure 4.14. *LaTeX created template.*

After the creation of the template, we should add all the objects of the document, as the 3D model, the text fields for the information and the observations of the operator and the table of components.

The programming of the rest of the parts is explained in chapter 3, so we only have to write the components, add the figure in the corresponding part and creating the TXT file to make the interactive document read all the information and showing it in the corresponding room of the template.

First of all, we have to open the documents in 3D Tool and export them as U3D files, like in figures 4.15 and 4.16.

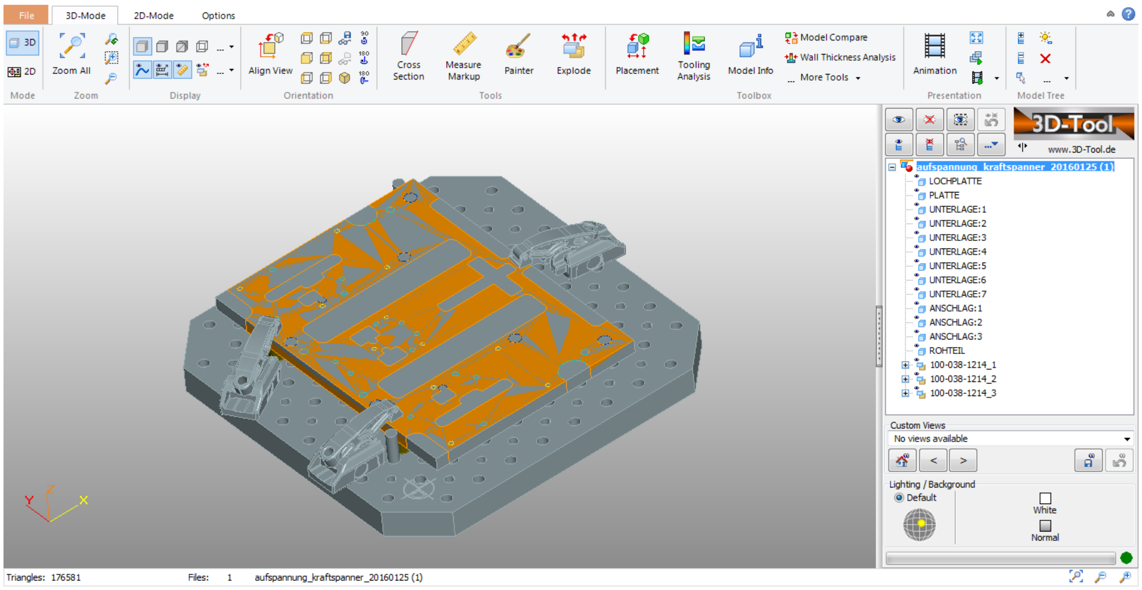


Figure 4.15. *Exporting Assembly 1 as U3D file in 3D Tool.*

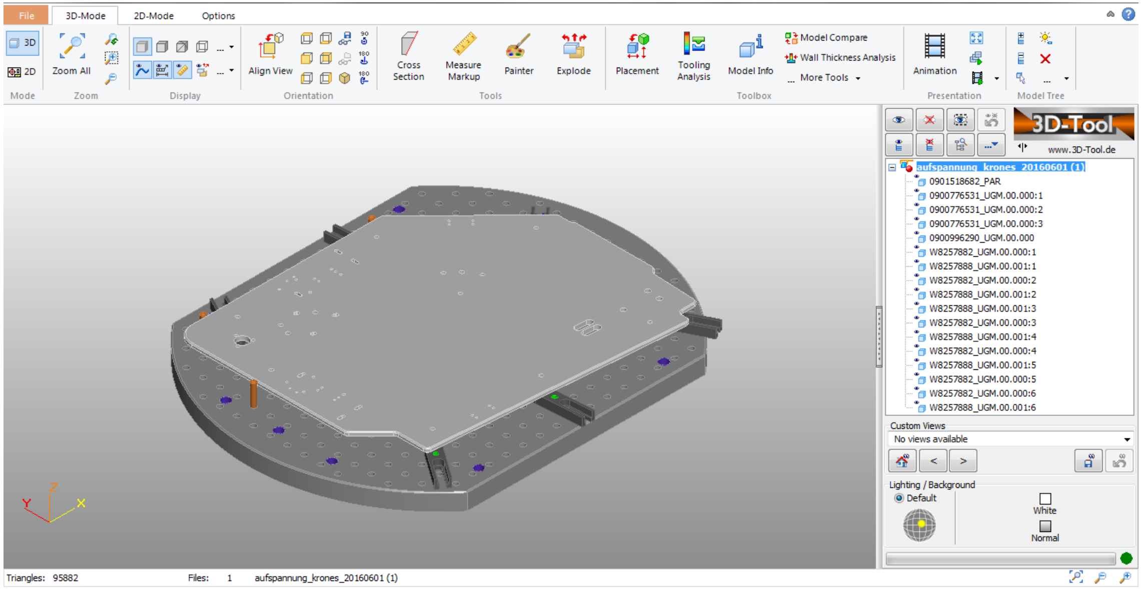
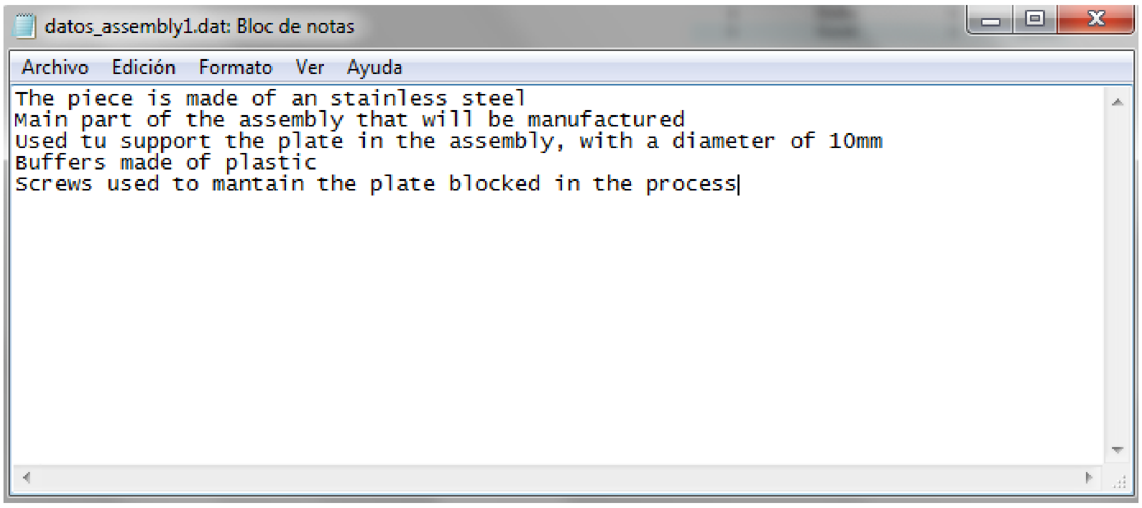


Figure 4.16. *Exporting Assembly 2 as U3D file in 3D Tool.*

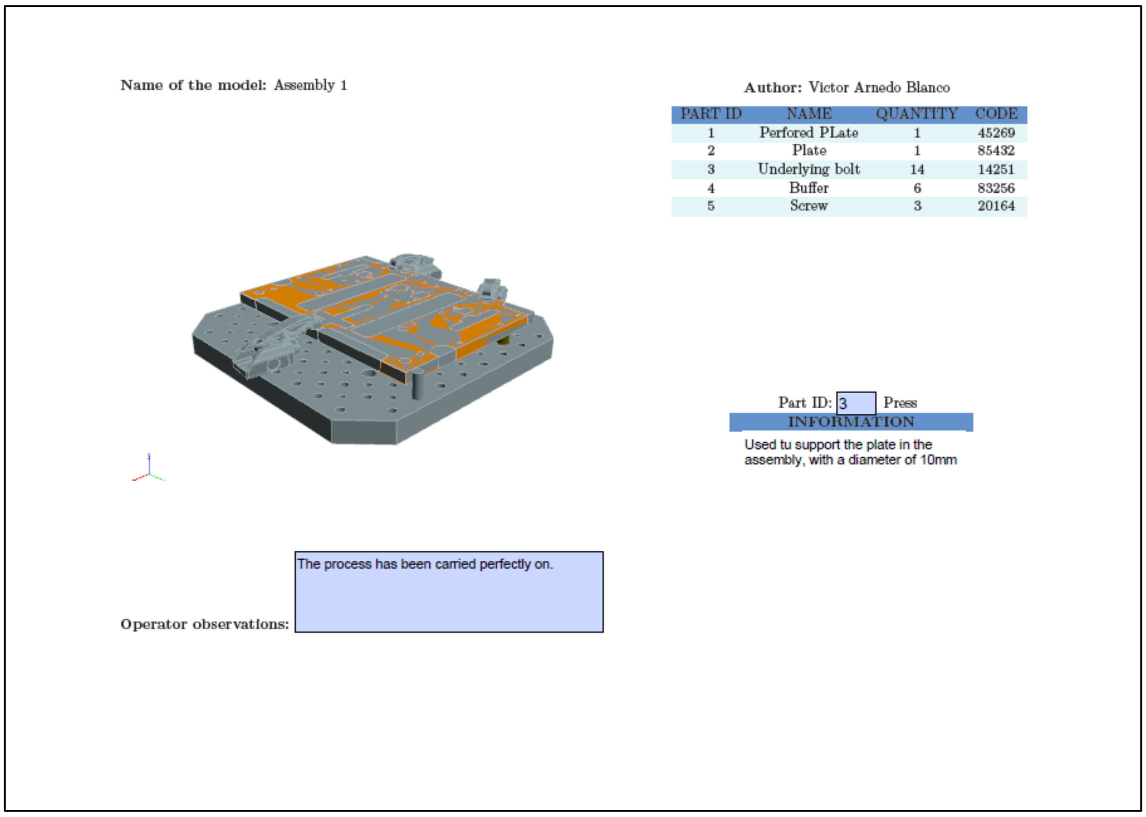
The next step in both cases is opening the coded program in the LaTeX editor and change the name of the U3D 3D model to import for the document, Assembly1 and Assembly2 in our case. Then, we would have to write the name, quantity and code of each component in the assembly in the *table_test.tex* file to be able to view it in the main document. Furthermore, we have to write a text pad document like in figure 4.17, in the same way, so the main program can read and show it in the information table.


datos_assembly1.dat: Bloc de notas
Archivo Edición Formato Ver Ayuda

```
The piece is made of an stainless steel
Main part of the assembly that will be manufactured
Used tu support the plate in the assembly, with a diameter of 10mm
Buffers made of plastic
Screws used to mantain the plate blocked in the process
```

Figure 4.17. *Text pad document for the information table.*

In figures 4.18 and 4.19 we can see the resulted document for both assemblies.



Name of the model: Assembly 1

Author: Victor Arnedo Blanco

PART ID	NAME	QUANTITY	CODE
1	Perfored PLate	1	45269
2	Plate	1	85432
3	Underlying bolt	14	14251
4	Buffer	6	83256
5	Screw	3	20164

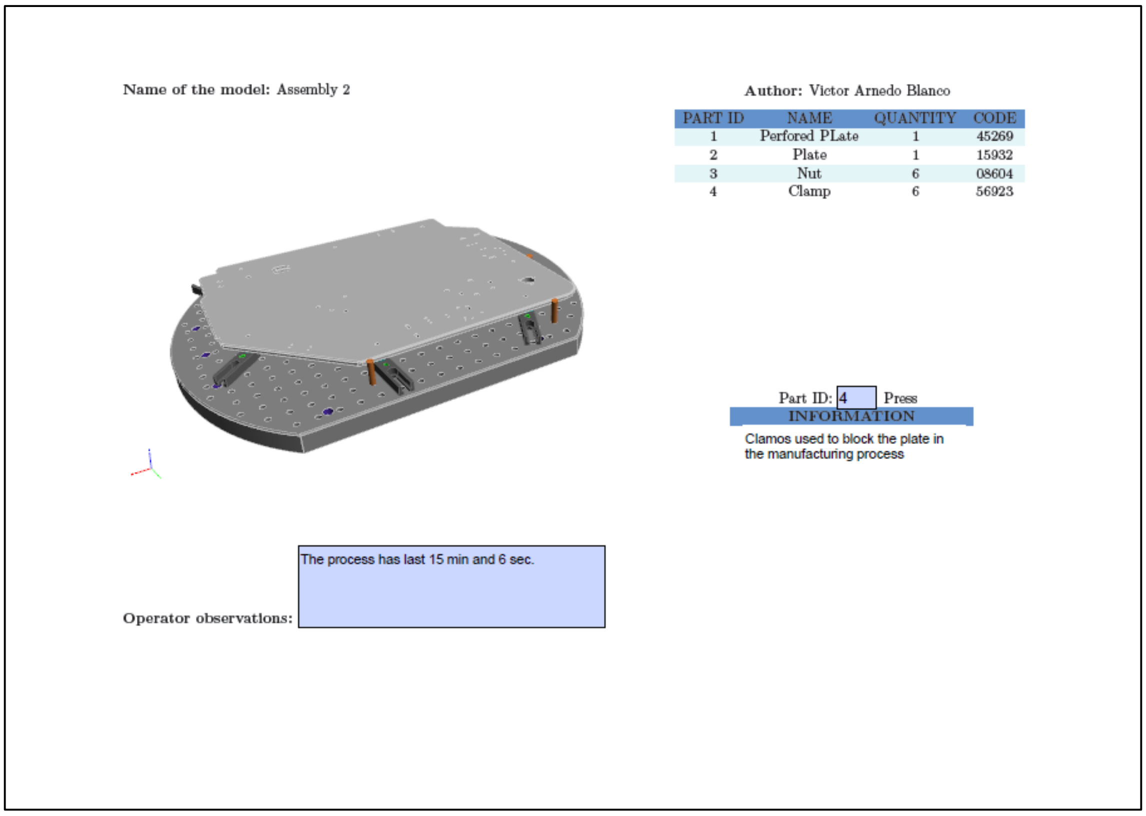
Part ID: 3 Press

INFORMATION

Used tu support the plate in the assembly, with a diameter of 10mm

Operator observations: The process has been carried perfectly on.

Figure 4.18. *Resulted document for assembly 1.*



Name of the model: Assembly 2

Author: Victor Arnedo Blanco

PART ID	NAME	QUANTITY	CODE
1	Perfored PLate	1	45269
2	Plate	1	15932
3	Nut	6	08604
4	Clamp	6	56923

Part ID: 4 Press

INFORMATION

Clamos used to block the plate in the manufacturing process

Operator observations: The process has last 15 min and 6 sec.

Figure 4.19. Resulted document for assembly 2.

CHAPTER 5: CONCLUSIONS

In this thesis, three solutions for the implementation of a workflow for the creation of 3D PDF documents from CAD/CAM software have been exposed. After working exhaustively with all of them we can conclude in some facts and opinions.

The carrying out of the Web application with HTML and X3D may be a good choice for its versatility, ease of creation and interactivity in different kind of devices. It supposes a distinct solution that also could be combined with the other two and the economic costs involved on it, attached to the conversion from STEP file to X3D, are quite low. Additionally, there is no necessity to HTML programming knowledge for its carrying out.

The 3D PDF creation has two different options to be put into practice. Each one has a process to be implemented, variating its difficulty, the time employed and the costs, but being objective, its results are really similar. Although Tetra4D offers a more intuitive way of work, automatic processes without programming and the no necessity of converting in U3D files the 3D model, the template and final document created with 3D Tool and LaTeX can reach an analog appearance and with a much lower price. It is true that we will have to learn TeX language in order to set our workflow up, but its results can be pretty satisfactory.

Therefore, the utilization of the last two tools will be the most suitable way to solve our initial issues. A little code will have to be written in order to design our own template and insert our components table, as said in the implementation section, but with the automation performed, the earning of time in the creation of a new document with a new 3D model embedded will be noticed. And this is one of the reasons why we will recommend this solution after the research done. They will be all exposed below:

- 3D Tool and LaTeX will suppose a slight economic investment compared with all the capabilities and features possible to be embedded in our templates and documents.

- The quality of the 3D PDF created is excellent and the elements can produce a very useful interactivity with the manufacturing operator.
- There is chance for further investigation and researching in the field so an amelioration of the documents created can be carried out.

- The efficiency concerning to the earning of time is obtained since the change of the 3D model published requires some steps that last no longer than a few minutes if the way to do it is known.

All the tools detailed in the thesis constitute a possibility of implementation for a workflow in the project, but 3D Tool and LaTeX fulfil all the necessities stipulated and will finally be the choice selected for our purposes.

CHAPTER 6: BIBLIOGRAPHIC REFERENCES

[1] Computerhope.com. (2016). *What is HTML (HyperText Markup Language)?*. [online] Available at: http://www.computerhope.com/jargon/h/html.htm [Accessed 3 Jun. 2016].

[2] Webopedia.com. (2016). *What is HyperText Markup Language - HTML? Webopedia Definition*. [online] Available at: http://www.webopedia.com/TERM/H/HTML.html [Accessed 3 Jun. 2016].

[3] Computerhope.com. (2016). *What is HTML (HyperText Markup Language)? - What is HTML5?*. [online] Available at: http://www.computerhope.com/jargon/h/html.htm [Accessed 3 Jun. 2016].

[4] Behr, J., Eschler, P., Jung, Y. and Zollner, M. (2009). X3DOM – A DOM-based HTML5/ X3D Integration Model. [online] pp.1-3. Available at: http://www.web3d.org/wiki/images/3/30/X3dom-web3d2009-paper.pdf [Accessed 4 Jun. 2016].

[5] WhatIs.com. (2016). *What is framework? - Definition from WhatIs.com*. [online] Available at: http://whatis.techtarget.com/definition/framework [Accessed 4 Jun. 2016].

[6] Doc.x3dom.org. (2016). *X3DOM Documentation: Getting Started*. [online] Available at: http://doc.x3dom.org/gettingStarted/background/index.html [Accessed 4 Jun. 2016].

[7] Wikipedia - La enciclopedia libre. (2016). *PDF*. [online] Available at: https://es.wikipedia.org/wiki/PDF [Accessed 6 Jun. 2016].

[8] Prepressure.com. (2013). *The history of PDF | How the file format and Acrobat evolved*. [online] Available at: http://www.prepressure.com/pdf/basics/history [Accessed 6 Jun. 2016].

[9] Wikipedia. (2016). *Portable Document Format*. [online] Available at: https://en.wikipedia.org/wiki/Portable_Document_Format [Accessed 6 Jun. 2016].

[10] Wikipedia. (2016). *PostScript*. [online] Available at: https://en.wikipedia.org/wiki/PostScript [Accessed 6 Jun. 2016].

[11] Ecma-international.org. (2016). *Welcome to Ecma International*. [online] Available at: http://www.ecma-international.org/memento/index.html [Accessed 6 Jun. 2016].

[12] Pdf3d.com. (n.d.). *About U3D Format | Create 3D PDF | 3D Conversion | Universal 3D - Universal 3D (U3D) Format*. [online] Available at: http://www.pdf3d.com/u3d/ [Accessed 6 Jun. 2016].

[13] Pdf3d.com. (n.d.). *About U3D Format | Create 3D PDF | 3D Conversion | Universal 3D - U3D is Used For*. [online] Available at: http://www.pdf3d.com/u3d/ [Accessed 6 Jun. 2016].

[14] Pdf3d.com. (n.d.). *About U3D Format | Create 3D PDF | 3D Conversion | Universal 3D - History of the U3D File Format*. [online] Available at: http://www.pdf3d.com/u3d/ [Accessed 6 Jun. 2016].

[15] Wikipedia. (n.d.). *X3D*. [online] Available at: https://en.wikipedia.org/wiki/X3D [Accessed 10 Jul. 2016].

[16] Sabia.tic.udc.es. (n.d.). *Conociendo X3D*. [online] Available at: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/x3d/Conociendo%20X3D.htm [Accessed 10 Jun. 2016].

[17] Sabia.tic.udc.es. (n.d.). *WebGL*. [online] Available at: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Tutoriales/TutorialWebGL/index.htm [Accessed 10 Jun. 2016].

[18] Fritz, R. (2015). *Three Reasons Tetra4D and Tech Soft 3D Fit Together | Tetra4D*. [online] Tetra4d.com. Available at: http://www.tetra4d.com/three-reasons-tetra4d-and-tech-soft-3d-fit-together/ [Accessed 12 Jun. 2016].

[19] Tetra4d.com. (n.d.). *Tetra4D Converter | Tetra4D*. [online] Available at: http://www.tetra4d.com/tetra-4d-converter/ [Accessed 12 Jun. 2016].

[20] Tetra4d.com. (n.d.). *Tetra4D Enrich | Tetra4D*. [online] Available at: http://www.tetra4d.com/tetra4d-enrich/ [Accessed 12 Jun. 2016].

[21] Unwalla, M. (2016). LaTeX: an introduction. *Communicator*, [online] Spring 2016. Available at: http://www.techscribe.co.uk/ta/latex-introduction.pdf [Accessed 16 Jun. 2016].

[22] Ctan.org. (n.d.). *CTAN: TeX*. [online] Available at: https://www.ctan.org/tex/ [Accessed 16 Jul. 2016].

[23] Wikipedia ES. (2016). *LaTeX*. [online] Available at: https://es.wikipedia.org/wiki/LaTeX [Accessed 16 Jun. 2016].

[24] Wikipedia. (2016). *LaTeX*. [online] Available at: https://en.wikipedia.org/wiki/LaTeX [Accessed 16 Jun. 2016].

[25] Isicad.net. (2016). *isicad: CAD Exchanger: The easiest way to convert 3D data - Introduction*. [online] Available at: http://isicad.net/articles.php?article_num=18432 [Accessed 24 Jun. 2016].

[26] Isicad.net. (2016). *isicad: CAD Exchanger: The easiest way to convert 3D data - The Beginning*. [online] Available at: http://isicad.net/articles.php?article_num=18432 [Accessed 24 Jun. 2016].

[27] Isicad.net. (2016). *isicad: CAD Exchanger: The easiest way to convert 3D data - Key reasons that make CAD interoperability challenging*. [online] Available at: http://isicad.net/articles.php?article_num=18432 [Accessed 24 Jun. 2016].

[28] Arquitectura de Computadoras. (2013). *COMPUTACIÓN PARALELA*. [online] Available at: https://conceptosarquitecturadecomputadoras.wordpress.com/computacion-paralela/ [Accessed 24 Jun. 2016].

[29] GitHub. (2015). *x3dom/cad-viewer*. [online] Available at: https://github.com/x3dom/cad-viewer [Accessed 17 Jun. 2016].

[30] GitHub. (2015). *x3dom/cad-viewer*. [online] Available at: https://github.com/x3dom/cad-viewer [Accessed 17 Jun. 2016].

[31] A. Grahn. (2012). *The movie15 Package.* [Online]. Available: http://ctan.math.utah.edu/ctan/tex-archive/macros/latex/contrib/movie15/doc/movie15.pdf [Accessed on: Jun. 6, 2016].

CHAPTER 7: COMPLEMENTARY BIBLIOGRAPHY

Dictionary.com. (2016). *the definition of HTML*. [online] Available at: http://www.dictionary.com/browse/html [Accessed 14 May 2016].

Wikipedia. (n.d.). *HTML*. [online] Available at: https://en.wikipedia.org/wiki/HTML [Accessed 14 May 2016].

Rouse, M. (2005). *What is HTML (Hypertext Markup Language)? - Definition from WhatIs.com*. [online] SearchSOA. Available at: http://searchsoa.techtarget.com/definition/HTML [Accessed 14 May 2016].

Es.wikipedia.org. (n.d.). *HTML*. [online] Available at: https://es.wikipedia.org/wiki/HTML [Accessed 15 May 2016].

Web3d.org. (2016). *What is X3D | Web3D Consortium*. [online] Available at: http://www.web3d.org/x3d/what-x3d [Accessed 16 May 2016].

Web3d.org. (n.d.). *Getting Started with X3D | Web3D Consortium*. [online] Available at: http://www.web3d.org/getting-started-x3d [Accessed 16 May 2016].

Álvarez, M. (2011). *¿Qué fue de VRML? Conoces X3D?* [online] DesarrolloWeb.com. Available at: http://www.desarrolloweb.com/articulos/vrml-x3d.html [Accessed 18 May 2016].

Doc.x3dom.org. (n.d.). *X3DOM Documentation: Tutorials*. [online] Available at: http://doc.x3dom.org/tutorials/index.html [Accessed 25 May 2016].

CoffeeCup Software. (2016). *Free HTML Editor*. [online] Available at: http://www.coffeecup.com/free-editor/ [Accessed 27 May 2016].

Acrobat.adobe.com. (n.d.). *PDF files, Adobe Portable Document Format | Adobe Acrobat DC*. [online] Available at: https://acrobat.adobe.com/us/en/why-adobe/about-adobe-pdf.html [Accessed 2 Jun. 2016].

Wikipedia. (n.d.). *Portable Document Format*. [online] Available at: https://en.wikipedia.org/wiki/Portable_Document_Format [Accessed 2 Jun. 2016].

Manuel, F. (2013). *Los mejores lectores PDF para Windows*. [online] Xatakawindows.com. Available at: http://www.xatakawindows.com/aplicaciones-windows/los-mejores-lectores-pdf-para-windows [Accessed 2 Jun. 2016].

Evans, D. (n.d.). *Print Center Features - Adobe PostScript vs. Adobe PDF*. [online] Adobe.com. Available at: https://www.adobe.com/print/features/psvspdf/ [Accessed 2 Jun. 2016].

Taft, E., Chernicoff, S. and Rose, C. (1999). *PostScript language reference*. 3rd ed. [ebook] Adobe Systems Incorporated, pp.1-15. Available at: http://partners.adobe.com/public/developer/en/ps/PLRM.pdf [Accessed 2 Jun. 2016].

Es.wikipedia.org. (2016). *PostScript*. [online] Available at: https://es.wikipedia.org/wiki/PostScript [Accessed 2 Jun. 2016].

File-extensions.org. (n.d.). *How to open and convert files with U3D file extension*. [online] Available at: http://www.file-extensions.org/u3d-file-extension [Accessed 3 Jun. 2016].

3D PDF Consortium. (n.d.). *U3D - 3D PDF Consortium*. [online] Available at: http://www.3dpdfconsortium.org/u3d/ [Accessed 3 Jun. 2016].

Systems, 3. (2015). *More on Universal 3D (U3D) Format - 3DA Systems*. [online] 3DA Systems. Available at: http://www.3dasystems.com/blog/more-on-universal-3d-u3d-format/ [Accessed 2 Jun. 2016].

Wikipedia. (n.d.). *Universal 3D*. [online] Available at: https://en.wikipedia.org/wiki/Universal_3D [Accessed 3 Jun. 2016].

Elsevier.com. (n.d.). *Interactive U3D models*. [online] Available at: https://www.elsevier.com/books-and-journals/content-innovation/interactive-u3d-models [Accessed 3 Jun. 2016].

Thoma, M. (2012). *Creating pdf-forms with LaTeX*. [online] Martin Thoma. Available at: https://martin-thoma.com/creating-pdf-forms-with-latex/ [Accessed 6 Jun. 2016].

Rainiero, N. (2012). *Embed 3Ds into PDF with LaTeX & U3D | Rainnic in the Clouds*. [online] Rainnic.altervista.org. Available at: http://rainnic.altervista.org/content/embed-3ds-pdf-latex-u3d [Accessed 6 Jun. 2016].

En.wikibooks.org. (n.d.). *LaTeX/Installing Extra Packages - Wikibooks, open books for an open world*. [online] Available at: https://en.wikibooks.org/wiki/LaTeX/Installing_Extra_Packages [Accessed 6 Jun. 2016].

LaTeX Project Team. (2001). *LaTeX 2E for authors.* [Online]. Available on: https://latex-project.org/guides/usrguide.pdf, [Accessed on: Jun. 6, 2016].

Wikipedia. (n.d.). *LaTeX*. [online] Available at: https://en.wikipedia.org/wiki/LaTeX [Accessed 6 Jun. 2016].

A. Grahn. (2015). *The media9 Package, v0.49.* [Online]. Available: http://texdoc.net/texmf-dist/doc/latex/media9/media9.pdf, [Accessed on: Jun. 6, 2016].

Sharelatex.com. (n.d.). *Positioning images and tables - ShareLaTeX, Online LaTeX Editor*. [online] Available at: https://www.sharelatex.com/learn/Positioning_images_and_tables [Accessed 7 Jun. 2016].

Texblog.org. (2012). *Multi-column and multi-row cells in LaTeX tables – texblog*. [online] Available at: http://texblog.org/2012/12/21/multi-column-and-multi-row-cells-in-latex-tables/ [Accessed 7 Jun. 2016].

T. Feuerstack, Bremen, Germany. (2003). *Dokumente einfach interaktiv.* [Online]. Available on: ftp://ftp.fernuni-hagen.de/pub/pdf/urz-broschueren/unplugged/upl007.pdf, [Accessed on: Jun. 10, 2016.]

Filippov, A. (2015). *Elphel Development Blog » X3D assemblies from any CAD*. [online] Blog.elphel.com. Available at: http://blog.elphel.com/2015/12/x3d-assemblies-from-any-cad/ [Accessed 10 Jun. 2016].

Brutzman, D. and Daly, L. (2007). *X3D*. Amsterdam: Elsevier/Morgan Kaufmann.

Techsoft3d.com. (2016). *Tetra4D® Products | Tech Soft 3D*. [online] Available at: http://www.techsoft3d.com/products/tetra-4d-applications/3d-pdf-converter/ [Accessed 12 Jun. 2016].

Tex.stackexchange.com. (n.d.). *TeX - LaTeX Stack Exchange*. [online] Available at: http://tex.stackexchange.com/ [Accessed 15 Jun. 2016].

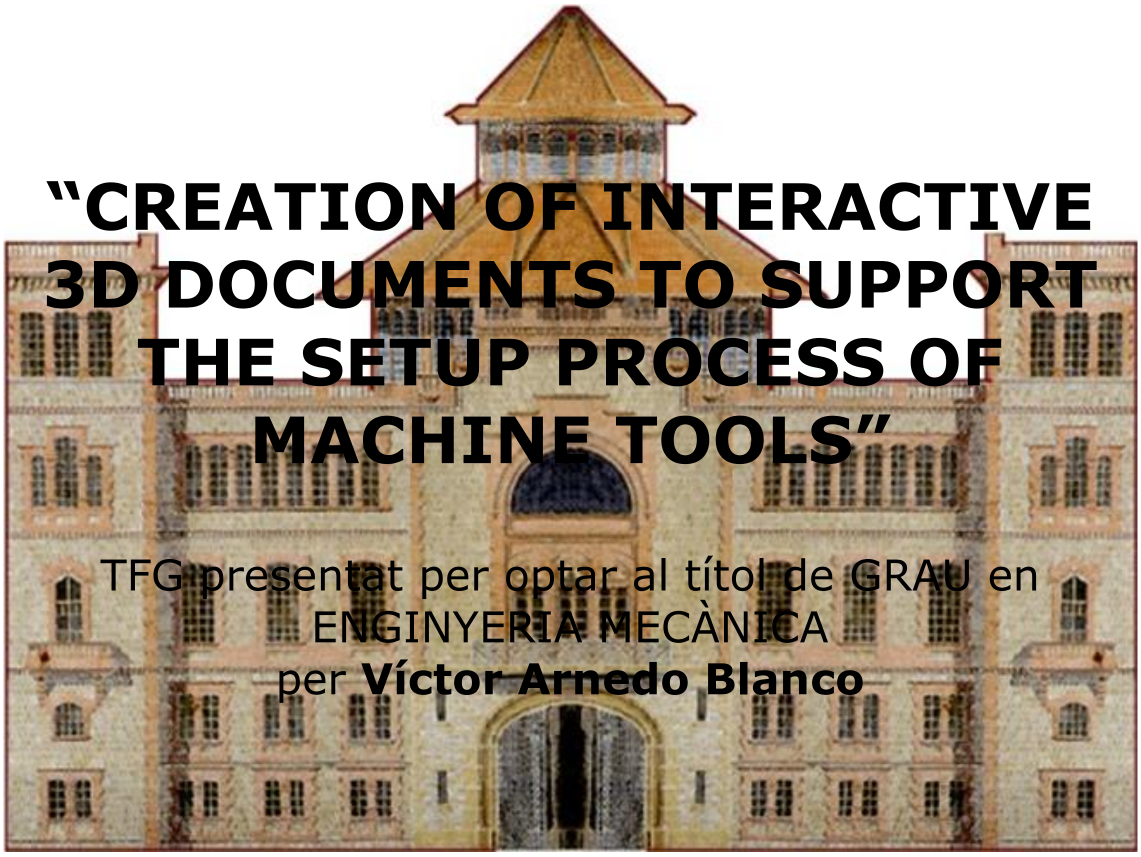
Cadexchanger.com. (2016). *CAD converter available in SDK, GUI and CLI - CAD Exchanger*. [online] Available at: http://cadexchanger.com/products.html [Accessed 17 Jun. 2016].

Dahl, C. (2004). *Planet PDF - Example Acrobat JavaScripts*. [online] Planetpdf.com. Available at: http://www.planetpdf.com/creative/article.asp?ContentID=6828 [Accessed 20 Jun. 2016].

Sharelatex.com. (n.d.). *Commands - ShareLaTeX, Online LaTeX Editor*. [online] Available at: https://www.sharelatex.com/learn/Commands [Accessed 2 Jul. 2016].

Escola Universitària d'Enginyeria Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Budget



"CREATION OF INTERACTIVE 3D DOCUMENTS TO SUPPORT THE SETUP PROCESS OF MACHINE TOOLS"

TFG presentat per optar al títol de GRAU en ENGINYERIA MECÀNICA
per **Víctor Arnedo Blanco**

Barcelona, 11 d'octubre de 2016

Director: José Antonio Travieso Rodríguez
Departament d'Enginyeria Mecànica (DEM)
Universitat Politècnica de Catalunya (UPC)

BUDGET

In this section the different costs will be exposed. First of all, we will carry a budget of the software to use on, and after selecting a solution, we will make a total costs budget. It is an aspect to have in consideration if we are thinking in adopting a workflow process for this issue. Once all the features and the implementation have been explained, we have to decide which is the most suitable solution, looking for a balance between quality and price. Here, in table 7.1, there is a comparative table in which we can see the different prices of the software and the cost will suppose every solution.

SOFTWARE	Solution 1 (€)	Solution 2 (€)	Solution 3 (€)
CAD Exchanger	548		
CoffeeCup HTML	0		
Tetra4D Enrich		1748	
Acrobat Pro		676.39	
3D Tool			300
LaTeX's TeXStudio			0
TOTAL	548	2424.39	300

Figure 7.1. *Tools budget of the software and solutions.*

It is appropriated to put into context this budget. The licences exposed, given by the companies, are for one year but might be offers and discount if the user or firma want to use more than one computer license or extend it for more than one year. This happens with CAD Exchanger for instance, that reduce the price of the software if the licenses are demanded for more than one year in approximately a 50% of the original price. Knowing this, we could compare the different options thinking about the savings of money.

One of the options stands out from the others. Tetra4D Enrich solution is a relatively expensive option for a single user and an object of discussion for a firma accounting department, since its price exceeds the amount of 2000€ per license. In this case, the client is paying for the quality and the facilities the product offers him, due to the fact he has not the necessity of programming any code and the software does all the conversions necessaries. The problem may fall to the necessity of acquiring Adobe's Acrobat Pro apart from Tetra4D Enrich, so it increases the cost in approximately a 40%. There is no necessity to have this software for the display of a 3D PDF, but Tetra4D Enrich only works in the Pro version environment, so it is a pack that must be bought.

If we do not want to spend so much money, it is preferable for us to waste some time in coding and creating our templates, the option would be 3D Tool and LaTeX. In an economical view, it is the best option to choose if our goal is to obtain PDF documents. We will need the 3D Tool Advanced version for the conversion and a LaTeX editor and compiler as TeXstudio for free. It is a high savings of money respect from the first solution, but it is also slower, so it is important to assess these issues.

Furthermore, if we prefer creating a Web application instead of a PDF document, we will have to obtain CAD Exchanger, the cheapest X3D file converter. If the option of the created and shared Web app exposed in the thesis is enough good, it will not suppose any cost, otherwise we will have to develop our own Web environment in order to implement the idea.

If we make an overview of all we have seen it is noticed that, except for Tetra4D Enrich, we have to pay for converter software. One thing that can be found out in the market analyse is the fact that there was a huge difficulty to find U3D and X3D converters, and the ones capable of making this conversion possible for free did not allow the import of STEP files. Under the circumstances, it was necessary to move to commercial software, and the cheapest ones are reflected in this work. If we want to have the quality of U3D and X3D files from STEP 3D models, we will have to spend money on it. This is the main reason why there are not non-cost solutions in the budget.

As in the report, we will consider the option of using 3D Tool and LaTeX so we will carry an engineering and documentation budget and add it to the tools budget so we can create a total costs budget.

Task	Cost of junior engineer (€/h)	Hours	Total cost
Initial information research	15	150	3000
Implementing solutions	15	80	1200
Programming	15	180	2700
Typing and revising	15	190	2850
TOTAL		600	9750

Figure 7.2. *Engineering and documentation costs budget.*

Concept	Cost (€)
Tools and software	300
Engineering and documentation	9750
IVA (21%)	2110.50
TOTAL	12160.50

Figure 7.3. *Total costs budget.*

As we can see in the tables, the total cost of the project carried on by a junior engineering will be of *TWELVE THOUSAND ONE HUNDRED SIXTY* euros and *FIFTY* cents.