

7-23-2010

A CAD INTERFACE FOR PRODUCT CUSTOMIZATION

Eddy Efendy

Purdue University - Main Campus, eefendy@purdue.edu

Follow this and additional works at: <http://docs.lib.purdue.edu/techdirproj>

Efendy, Eddy, "A CAD INTERFACE FOR PRODUCT CUSTOMIZATION" (2010). *College of Technology Directed Projects*. Paper 24.
<http://docs.lib.purdue.edu/techdirproj/24>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.



College of Technology

A CAD INTERFACE FOR PRODUCT CUSTOMIZATION

In partial fulfillment of the requirements for the
Degree of Master of Science in Technology

A Directed Project Proposal

By

Eddy Efendy

Committee Member

Approval Signature

Date

Richard M. French , Co-Chair _____

Bradley C. Harriger , Co-Chair _____

Henry W. Kraebber , Member _____

TABLE OF CONTENTS

Executive Summary	3
Introduction.....	4
Statement of the Problem.....	6
Significance of the Problem.....	7
Statement of the Purpose	9
Definitions.....	11
Assumptions.....	11
Delimitations.....	12
Limitations	12
Literature Review.....	14
Methodology	22
<i>Time Action Plan</i>	22
Programming Approach.....	23
<i>Code Development</i>	31
<i>Establishing Base Model</i>	37
Result.....	50
Conclusion and Future Improvement.....	52
References	55
Appendices	58
Appendix A – Single Cutaway Customization Form.....	59
Appendix B – Double Cutaway Customization Form	60
Appendix C – Visual Basic Code	61

Executive Summary

Product customization has become more prevalent in today's manufacturing industry. In fact, given a choice in today's society and taking into consideration the present technology, customers would prefer a product that can be built according to their specific needs and desires (Kumar, 2008). Customization of a product is typically done by an individual possessing the necessary product knowledge and design skills and applying them to the interpretation of the customer's desired requirements and specifications. This process is generally costly and time-consuming. As computer-aided design (CAD) software grows in terms of control and functionality, the potential exists to place common customization requests in the hands of the customer. This capability will permit even customers with little to no CAD skills to customize a product within a defined range of specifications.

This project has created an intelligent interface to allow a consumer with minimal CAD knowledge to interact with the software to make desired, common customizations to an existing product design. The user-interface was developed using the Visual Basic programming language and the CAD software's built in application programming interface (API) command structure. This process allows the automation of select embedded CAD productivity tools whereby the customer is able to modify specific parameters to manipulate the shape of their product and automatically generate a 3-dimensional computer-aided model reflecting their specific modifications.

This project utilized a solid body electric guitar body as the product example for this study. The methodology for this project is described, as well as specific limitation ranges and/or constraints that were placed on the parametric parameters of the guitar body. Due to the complex shape of the product and the unpredictable desires of the customer for customizing the product,

the parametric sketch was constrained to limit the kind of transformation the parametric model was able to do.

The result of this project was the development of an interactive design tool that prevents design engineers from needing to design the same product family repeatedly to suit the needs of the customer, thereby reducing the design time and mistakes while enhancing the consistency of the product.

Introduction

The concept of mass customization has become increasingly popular since the 1990's (Pine, 1993). This is due to customization offerings a competitive advantage to companies with increased customer value. Furthermore, in keeping with the evolving paradigm of mass customization, Meyer and Utterback (1992) also introduced the concept of product family design, where standardized products can be replaced with specific features and functionality according to customers' specific needs and desires. One may conclude by these studies that by automating the design process to allow the customer more range of direct, interactive control with the design, companies could experience a significant reduction in operation costs.

With today's emerging markets and product variety, it is very important for industrial companies to explore product customization to capture customer attention and deliver true customer value. The challenges are formidable, especially with today's customers who consistently demand "a product with the highest quality, fastest delivery, and highest level of product customization" (Kumar, 2008). Unfortunately, most of today's product customization falls under the category of *customized standardization* (Lampel & Mintzberg, 1996), where the customers are not involved in the design and manufacturing process, as depicted in Figure 1 below.

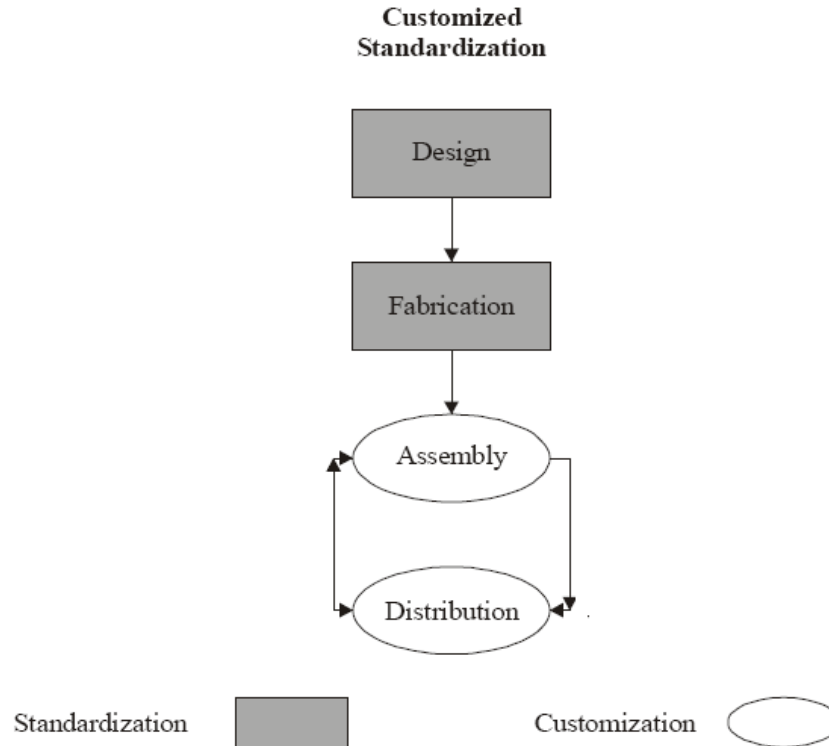


Figure 1. Customized Standardization Strategy (Lampel & Mintzberg, 1996)

Customization is a widely studied subject. Kumar (2008) and Siddique and Boddu (2003) concurred that with today's expanding World Wide Web, product customization has made tremendous progress. However, according to Kumar (2008), there is a notable lack of movement towards product customization for manufacturers.

According to the survey result by Wilson (2007), "CAD is the primary tool used to support the customization process (92%). The implication is that the customization process is primarily drawing-driven based on tribal knowledge with heavy engineering involvement in the specification process." Unfortunately, the survey also implied that "There is very little integration of tools within the customization process...The lack of integration implies that there is a significant amount of manual intervention within the customization process requiring time and resources, and leaving opportunity for errors."

The study performed by Wilson (2007) shows a need to increase customer involvement in customizing a product during design and manufacturing process. Therefore, the overall nature of the project was to develop an application for producing customized products for customers with poor or no skill in CAD software. This project focused on developing an interface for the customers to use to modify product appearance by seamlessly integrating several software applications. In other words, the customer is now able to shape a product to their satisfaction without needing to know how to use CAD software. This participatory design method is the newest step in customer satisfaction.

Statement of the Problem

Traditionally, individuals seeking to purchase products tend to shop at stores or over the Internet. Unfortunately, most of the product's appearance and features have been designed and produced by the company. The only customization that can be made to the product by the customer is no more than a cosmetic change, such as texture and color (Dauner, Launder, Stimpfig, & Reuter, 1998).

According to Kumar (2008), this type of product customization is becoming inadequate for today's society. However, most manufacturing companies produce products in bulk is because they are trying to lower production costs. Unfortunately, this focus allows the company to lose sight of customer's unique requirements (Holweg & Pil, 2001).

Therefore, one of the growing challenges for twenty-first century manufacturing companies is to keep up with the unique demand from customers and competitive pressure to reduce cost. Build-to-Order (BTO) is the dominant approach used in today's manufacturing industries to solve those problems. BTO refers to products that are custom build according to

what customers want. Yet, only 14% have embraced mass customization, which is “ironic given that BTO is a “customer driven” strategy.” as confirmed in Figure 2 below (Wilson, 2007).

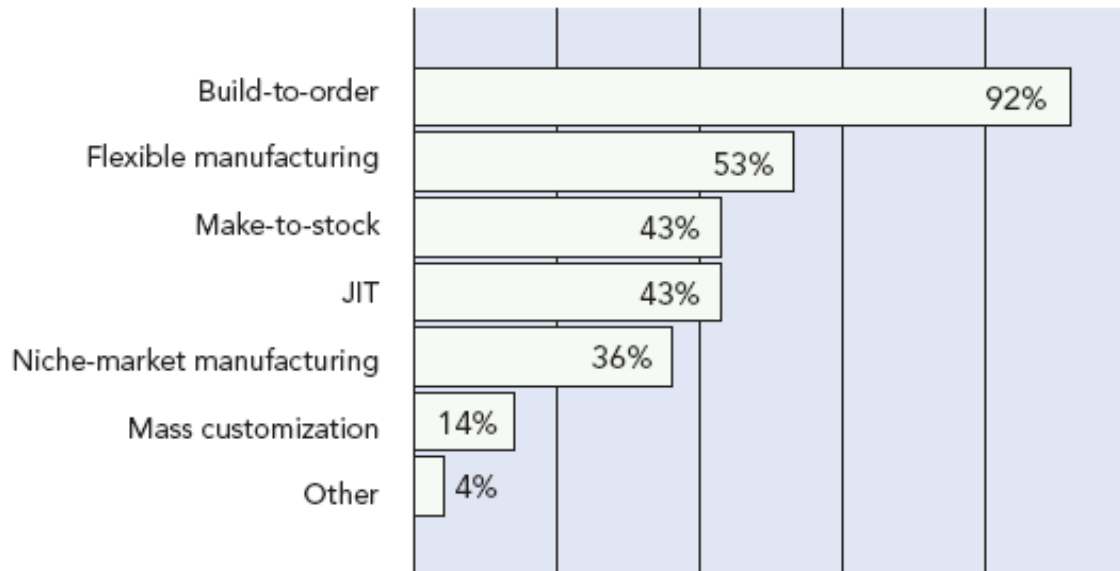


Figure2. Demand strategies in which most companies employ (Wilson, 2007)

Moreover, to deliver customer requests, design engineering spends most of their time on the drawing board, designing customized products to the customers’ liking. That is due to the fact that “engineering is uniquely positioned to optimize the fit between a customer’s needs and manufacturing, this is time very well spent. Unfortunately, much of the time spent is low-value activity” (Wilson, 2007).

Unfortunately, customers have limited knowledge on the design process of a product. Therefore, to bridge the gap between design engineering and the end user, this research focused on developing an interface through which potential buyers could customize the appearance of any given product, without being an expert in the design software.

Significance of the Problem

One fundamental factor in product customization is communication concerning customer’s specific requirement. Sadly enough, one of the difficulties of product customization

is the lack of understanding between the design engineer and the customer (Åhlström & Westbrook, 1999). If the design engineers misinterpret the customer's request, worthless work could occur. As previously mentioned, product customization is a time consuming process for the design engineers. Many companies cannot respond quickly enough, and for this reason, most manufacturing companies produce products in bulk, which often results in an oversupply of inventory and diminishing profits if they cannot find a customer. Therefore, there was a definite perceived need to undertake this project. This project provides the customer with a direct, interactive tool to modify a product's design without the need for skills or knowledge of any specific CAD system, only knowledge of the product. Once the product is designed and documented according to customer's desired specifications, the manufacturing process can then be initiated.

This project provides a tool to assist the efficiency of the design process as it applies to the customization of manufactured products. Failure to explore this area only contributes to the continued inefficiencies of the design process whereby a customer can only acquire predominantly standard products that have been predesigned and manufactured by a company, with little to no opportunity for customer input on possible product design modifications that would better serve the customer's needs or desires.

Customers place a great value on personalized products because they are able to form a bond with the product that will consequently have an impact on the brand and the company. Customization not only increases customer satisfaction, it can also increase market share according to Åhlström and Westbrook (1999). Furthermore, according to Wilson (2007), manufacturing companies can also reap a benefit by charging a higher premium price for customized products.

A project in this area is timely; according to Wilson (2007), “Market demand for customized products is increasing and expected to continue to grow. 63% of respondents have seen an increase in demand in the last five years, and 26% anticipate that the growth rate will be between 25% and 50% in the next two years.”

According to Piller (2007), in order to survive in today’s market; many manufacturing companies are offering the customers the ability to customize a product appearance to their liking. Therefore, product customizations are significant for today’s manufacturing industry, in order to gain a competitive edge among their competition. A better understanding regarding product customization and its impact on the customers can lead to a superior manufacturing process, which in turn will improve customer satisfaction.

Statement of Purpose

According to Wilson (2007), “one of the primary barriers to customization effort is the lack of knowledge the customers have on option...There are huge opportunities for improvement in sales and operational effectiveness to be gained by addressing this issue.” as revealed in Figure 3 below.

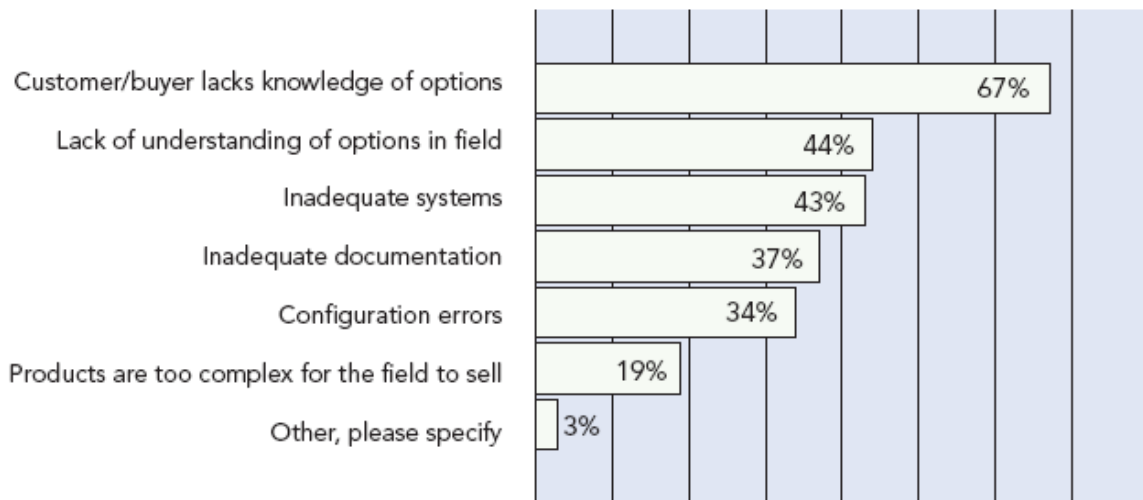


Figure 3. Barriers to customization (Wilson, 2007)

The purpose of this project was to create a tool to allow customers to be more involved in the design phase for product customization through the use of an interface that is seamlessly integrated with CAD's system API. Figure 4 displays the system architecture for this project.

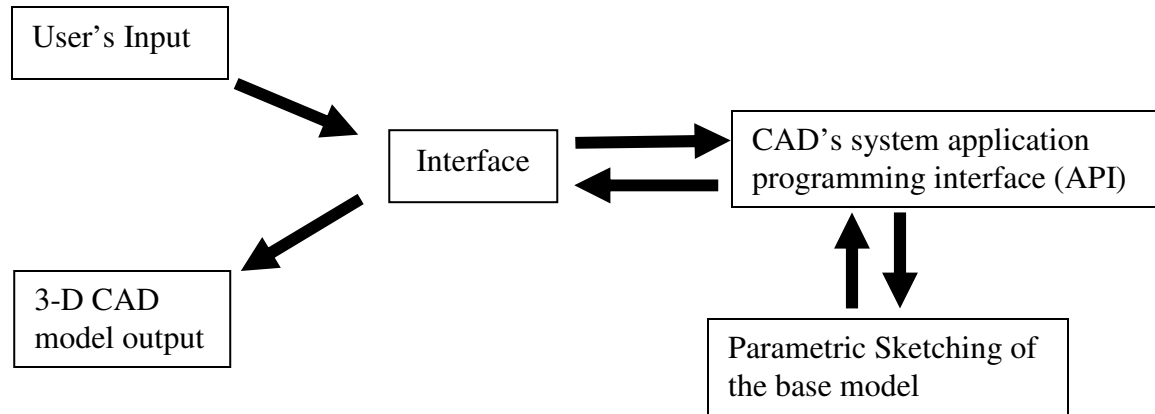


Figure 4. The system architecture

Basic activities and information flow of the system architecture are summarized by the following description:

1. A user-interface form was developed for the customers using Visual Basic language.
2. Customers input values for pre-identified feature dimensions for a product mode.
3. A new product model will then be design according to the customer's parameter values via CAD's system application programming interface (API).

In short, this technique provides customers with the ability to make design changes to the product without the need to possess design skills in CAD software. In essence, this method attempted to close the gap of misinterpretation in product customization between the design engineers and the customers. Most importantly, this project educates the customer about what options are available for them.

Definitions

API – Application Programming Interface refers to the accessibility of the software functions that can be called automatically/programmatically (Lombard, 2007). For instance, “you can use Autodesk Inventor’s API to write a program that will perform the same types of operations you can perform when using Autodesk Inventor interactively.” (Autodesk Inventor Object Library, 2009)

CAD – Computer Aided Design refers to computer software that aid in the design/drafting of a part/product.

VB – “Visual Basic is one of the software industry’s most popular development language for creating standalone software components, including executable programs, ActiveX controls and COM components.” (Autodesk Inventor Object Library, 2009)

VBA – Visual Basic for Applications is the subset of Visual Basic that is designed to provide development capabilities inside any other individual software application. It provides programming development tools required to customize application and integration solutions. (Keenan, 1999)

Assumptions

The following assumptions were made at the beginning of the project.

- 1) Since this project relied greatly on software integration, the most important assumption was that all the software used can be integrated together.
- 2) The CAD’s API needed to be able to provide the appropriate commands for communicating with the identified design tools within the Inventor.
- 3) Customers do not have to have CAD software knowledge.

- 4) The customers should be knowledgeable about specific product guitar families so it is easier to answer the questions provided by the user interface.

Delimitations

Some delimitations of this project were as follow:

1. SolidWorks software was chosen for this project over other Computer Aided Design (CAD) software because the personal investigator has more expertise in SolidWorks software than any other CAD software.
2. The framework for this project was focused on the customization of a family of electric guitars due to the personal investigator's involvement in the guitar workshop at Purdue University. This activity lays the ground work for the future of the guitar workshop.
3. This project focused on single-cutaway and double-cutaway guitar shapes because of their well-known fundamental basic shape.
4. Due to the shape complexity of the guitar, customers are limited to specific cutaway styles.
5. In order to maintain design integrity, customization ranges for the cutaway style was limited.

Limitation

The project was limited by several factors.

- 1) The CAD software used is an educational version. Therefore, there was no guarantee of full performance as compared with the industrial versions of the software.
- 2) Software compatibility between the selected software packages was limited due to software versions.

- 3) All aspects of the product design were limited to the design tools that were available in the CAD software.
- 4) The development of the user interface of this project was dependent on the commands available in the CAD's API structure.

Literature Review

Introduction

Product customization is not a new concept. The demand and the number of studies done in this area have grown exponentially since the beginning of the 1990's (Pine, 1993); nevertheless, there is much room for continued research (Kumar, 2008). For this particular project, the review of the literature was conducted in order to compare, contrast, and analyze previous practices in product customization.

Before conducting the review of literature, the project was thoroughly discussed with Professor Bradley Harriger, Professor of Mechanical Engineering Technology. This discussion provided some guidance as how to appropriately create a suitable outline of information for the review of literature.

Product customization is a very broad expression, including many specific classifications. Knowing the classifications of product customization was crucial for the continuation of this project in order to understand the background of the project's specific focus. Once the classifications of product customization were explained, prior methods according to these classifications were then gathered to be analyzed. Finally, since this project utilized programming language in customizing a product, further review of literature was searched and studied to see what had been done so far.

Therefore, to illustrate this, the review of literature was focused on the amalgamation of information concerning:

1. Classifications of product customization
2. Prior methods in product customization interfacing
3. The utilization of programming language in product customization

Methodology in Conducting the Review of Literature

Key terms used in product customization were identified so that a search for reference material could be conducted. These key terms were used individually and in combination and included such phrases as “Product Customization,” “Computer Aided Design (CAD) Custom Product,” “Visual Basic Application (VBA) CAD,” and “Application Programming Interface (API) CAD” Using these mentioned keywords, a search was conducted via the search engine on engineeringvillage.com, an online catalog subscribed to by Purdue University. This website contains engineering journal articles from a plethora of research databases such as Elsevier, Compendex, Pergamon, and Informaworld. Additional recourses such as internet search engine, Autodesk Inventor Object Library, and textbooks on the subject of Visual Basic were also used.

Results of Review of Literature

As mentioned, the review of literature was divided into three categories, in which journal articles were reviewed for relevant information.

1. Classifications of product customization. Organizations view customization in various ways. In order to have a clear idea about customizations, this review of literature began with a framework to identify and classify customizations. Coates and Wolff (1995), Lampel and Mintzberg (1996), and Gilmore and Pine (1997) have classified customizations according to the customer’s involvement. Although their perspectives are different from one another, they do overlap in some areas.

Coates and Wolff (1995) categorized customization in terms of manufacturing practice: soft customization and hard customization. Soft customization is when the customer does not

interfere during the design and fabrication processes, whereas in hard customization, the customer is fairly involve in those aforementioned processes.

Lampel and Mintzberg (1996) perceived customization in a more detailed way when compared to Coates and Wolff (1995). Lampel and Mintzberg (1996) perceived customization as particular strategy, depending on how much involvement the customer has in the value chain of a manufacturing firm. Lampel and Mintzberg (1996) have developed a manufacturing firm value chains into four structures: design, fabrication, assembly, and distribution. Therefore, depending on the customer’s involvement within the value chain, Lampel and Mintzberg (1996) came up with five different strategies as depicted in Figure 5 below.

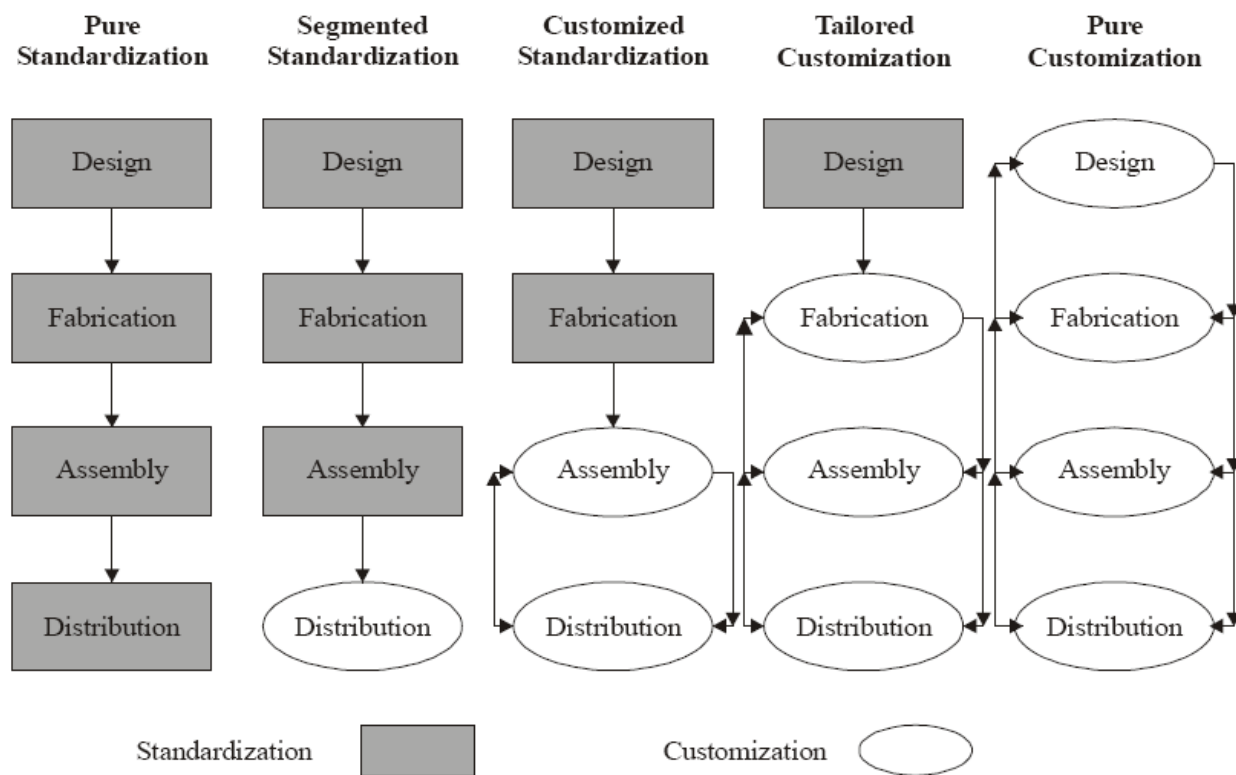


Figure 5. A Continuum of Strategies (Lampel & Mintzberg, 1996)

- Pure Standardization refers to a product that has already been completely built by the company, where the customer has no influence over the produced product.

- Segmented Standardization is where the company responds to the specific need of different group of customer, and therefore, the company makes different types of products according to that specific group of customer.
- Customized Standardization refers when the product is customized for the customers at the end of the production phase (at the assembly point).
- Tailored Customization is where the company has a basic design of a product and is able to tailor the product according to the customer's desire.
- Pure Customization is when the customers are able to customize the product from the beginning of the value chain.

Lastly, Gilmore and Pine (1997) have recognized four distinct approaches to mass customization: collaborative, adaptive, cosmetic, and transparent customization, as depicted in Figure 6 below. This approach is designed to facilitate managers to determine the type of customization they should pursue in their organization.

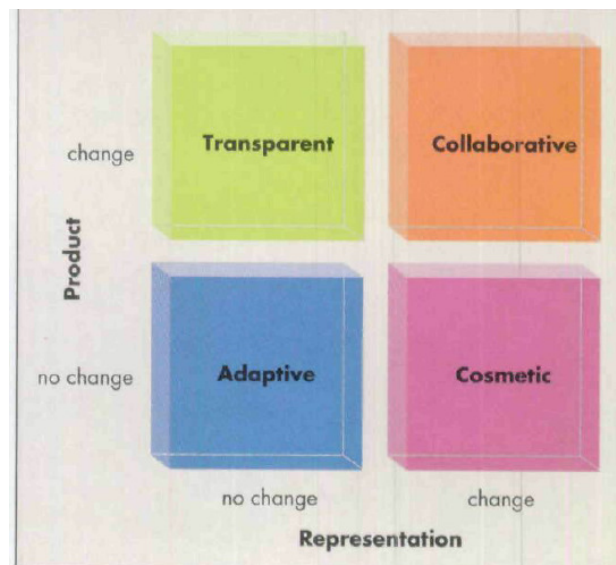


Figure 6. The Four Approaches to Customization (Gilmore & Pine, 1997)

- Collaborative customization is where the product is tailored according to the customer's need but the fabrication and assembly side are standardized.

- Adaptive customization is where the product is standard but designed to be customizable according to customer's need.
- Cosmetic customization is where the product remains standard but the product presentation is tailored to customer's need.
- Transparent customization is where the product is customized without the customer's explicit knowledge

These classifications further clarify the direction for this project. The following conclusion is made as a result of the aforementioned information. The objective of this project was classified towards hard customization according to Coates. More specifically, this project was classified towards tailored customization according to Lampel, and collaborative customization according to Gilmore.

2. Prior methods in product customization interfacing. The operation-strategy for product customization has evolved over the last four decades. Kumar (2008) summarized how the priority of competition has changed over time: “price until the mid 1980s, quality until the early 1990s, flexibility until the mid 1990s, and agility or responsive thereafter.” Aside from the customization strategies aforementioned, there have also been numerous methodologies and technologies that support the development of product customization as the customer continues to demand a product with the highest level of personal satisfaction. This section will review how technologies support information transfer from customers to manufacturers and what technologies makes product customization possible.

The following are some previously-made attempts at establishing an involved customer-manufacturer communication link. Researchers such as Siddique and Boddu (2003) and Yen and Ng (2000) have developed Web-based product customization systems. However, both Yen and

Ng (2000) and Siddique and Boddu (2003) present their work differently. Yen and Ng (2000) proposed an electronic catalog for custom products, which is stored on the World Wide Web. Meanwhile, Siddique and Boddu (2003) procedures on product customization are more detailed, using Web related tools to collect user specification and generate an automatic CAD model of a product according to customer requirements.

In both scenarios, the specifications from customers are fed into CAD systems. Clearly, it seems that CAD systems are the main enabling technologies that support product customization. This is expected because CAD systems allow design changes expeditiously. However, to communicate with the CAD system, a generic programming language must be utilized. Correspondingly, Siddique and Boddu (2003) used a C programming language to communicate with an Application Program Interface (API) for CAD software to generate the product model according to customer specification.

Therefore, narrowing the focus on the usage of programming language to enhance the customer's satisfaction, the following literature review analyzes how programming language is used to assist people who don't know anything about CAD.

3. The utilization of programming language in product customization. Programming language has been around since the beginning of the twentieth century. Since then, programming language has evolved tremendously, in both research and industry. One of the most popular programming languages is Visual Basic (VB) because of its simplicity, and therefore, it is used by many beginner programmers. Today, almost all software, including CAD software has Visual Basic for Application (VBA) built into their application. VBA is another version of VB that was designed to provide custom solutions in all aspects of the host application.

According to Keenen (1999), “Because VBA has been embraced within automation; a single common development language can now be used among multiple software products in a manufacturing application.” To this, Keenan (1999) also addresses that with VBA assistance, “Manufacturers also are starting to realize the benefits of connecting processes across factories, among software programs of all kinds. VBA allows this integration among software programs that are controlling processes for not only the device layer and control layer, but also the information layer.”

The utilization of VBA is abundance. Researchers such as Prince, Ryan, and Mincer (2005), Seppanen (2000), Gattamelata, Pezzuti, and Valentini (2006), and Sanson (2006) have successfully used Visual Basic to integrate and customize software to meet the needs of their application. Prince, Ryan, and Mincer (2005) are able to design, model, and analyze a simply supported shaft instantaneously by merely entering a few parameters in Visual Basic. Meanwhile, Seppanen (2000) used Visual Basic to modify a model data in Arena simulation software by changing some parameters in Microsoft Excel software. With similar technique, Gattamelata, Pezzuti, and Valentini (2006) used Application Programming Interface (API) in CAD system to re-construct a tessellated surface from a 3d laser scanner into an editable solid feature in CAD software. Lastly, Sanson (2006) utilized Application Programming Interface (API) to assist inexperienced non-optical designers to execute repetitive optical design tasks.

Conclusion

In today’s global society, product customization has become part of an important manufacturing strategy. This review of literature provided further clarifications for the project objectives in which theoretical aspects with reference to product customization concepts and classifications were discussed. The divisions made by Coates and Wolff (1995), Lampel and

Mintzberg (1996), and Gilmore and Pine (1997) were all suitable for this project. Furthermore, the literature review reveals that much product customization research and application has been done. Moreover, to implement product customization, programming language, particularly, Visual Basic plays an important role in integrating different manufacturing technologies.

1. Programming Approach. During this phase, each piece of software used in the project was researched to understand its basic functionality, compatibility and interoperability. Software design and familiarization was also examined as a part of this phase as well prior to starting the development of the code. The software design activity included the development of a strategy/plan for the creation of the software interface application that was the primary focus of this project. Once the strategy was determined, further research on software specific codes was necessary to become familiar with their coding commands, syntax structure and format. The result of the project provides customers with the ability to make design changes to a product via a user-interface form without the need to possess any type of CAD software skills. In the simplest sense, customers input values for pre-identified feature dimensions for a product model and the developed software program will automatically generate a new model based on the parameters. Figure 7 shows a basic block diagram for this procedure.

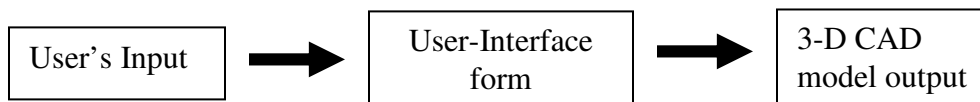


Figure 7. Basic system architecture of the project

While the basic concept of the application is relatively simple, the planning, layout and logic behind the development of the application interface was challenging and quite involved. Visual Basic 2008 Express and SolidWorks were the selected software packages for creating the user interface form and the 3D solid model respectively.

One key element needed for the success of this project was a thorough understanding of the CAD tools available in SolidWorks for generating multiple part configurations. SolidWorks allows the development of part configurations through the use of a design table. A design table uses an Excel spreadsheet to allow a user to enter part feature information to create a new

version or versions of a product by changing selected part dimensions or by suppressing part features. The Excel Spreadsheet can be automatically linked to the SolidWorks software, so when a person edits the data in Excel spreadsheet externally, SolidWorks will create the new model or models based on the entered data.

To make sure that these pieces of software would worked properly together, software compatibility was checked before going any further. It was found that each software uses a common programming structure called an application programming interface (API) which enables knowledgeable software users to create tools using some version of the Visual Basic programming language to interact with the software. It was discovered that one could accomplish this interaction using two basic approaches depending on the type of tool being developed and the type of user interaction desired. Those approaches included: VBA, Add-Ins (DLL or EXE), and Standalone EXE, as depicted in Figure 8.

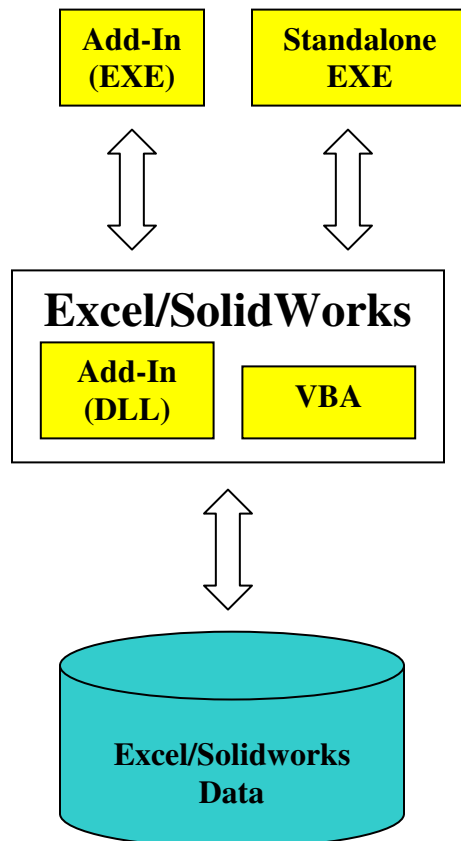


Figure 8. Accessing Excel/Solidworks API

The purpose of this project is to provide the customer the ability to make design changes to the product with little to no functional skills or knowledge of CAD software. There are basically two fundamental ways to accomplish this project, customize a model from within SolidWorks or customize the model outside of the SolidWorks software. Therefore, customizing the model from within the CAD software would defeat the purpose of this project since the user would need some level of functional knowledge of the CAD software.

Based on the criteria for this project, the Standalone EXE was selected as the method of choice for this project. A Standalone EXE is an independent program that is capable to control other software and has its own interface. In this case, Visual Basic software was used as the Standalone EXE to interactively work within the SolidWorks and Excel software to customize a model.

Therefore, the three main pieces of software and their basic purpose for this project are as the follows:

- Visual Basic – Creates the user-interface form and integrates SolidWorks and Excel together using each software's API command structure.
- SolidWorks – 3D solid model creation
- Microsoft Excel – Customer's parameter values are entered in.

1.1 Software Design. There are hundreds of published API commands for the software used in this project. Reading and comprehending each and every API code would have taken a very long time, creating unnecessary work and time. It was decided to break down the tasks required from each software and then focus on more specific API commands that would accomplish the projects goals.

Since the basic principle for each software had been realized, a more detailed description of the project's integration structure is depicted in Figure 9 and explained as follows: once the user has submitted parameter values into the user-interface form, the parameter values will automatically be entered in Microsoft Excel, which in turn updates the SolidWorks 3D model automatically, and the result will be displayed back to the user-interface form for the customer to compare.

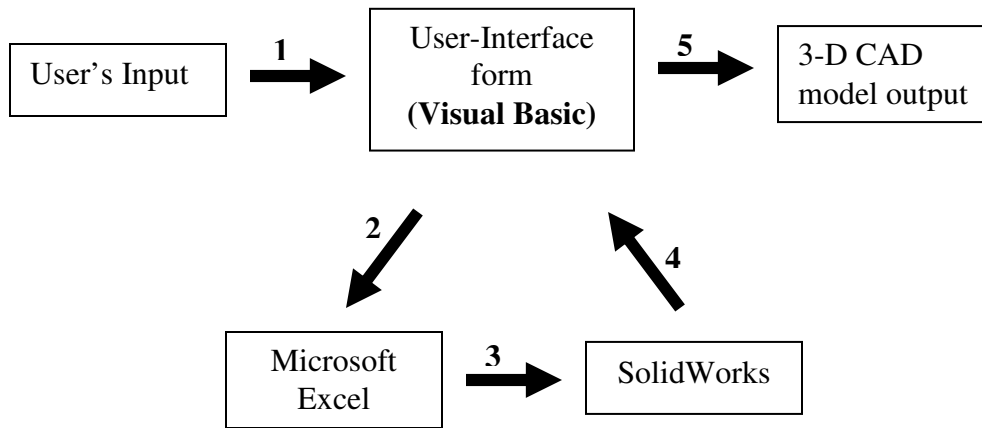


Figure 9. Detailed system architecture of the project

For the customer to automatically enter the parameter values into the Excel, several Excel API commands needed to be focused on, specifically opening an existing file, adding the data to the file, saving the file, and lastly closing the Excel file.

Since the Excel file is linked to a Solidworks file, the part model will be automatically updated once it is recognized or identified by the SolidWorks software. Similar to the Excel API commands, some important SolidWorks API commands required included: how to automatically open the Solidworks file, link the Excel file, select the updated part configuration, save and close the SolidWorks file, and display the newly created configuration.

Although the API commands for creating the user-interface form is moderately simple, the essence role for Visual Basic was to be able to manipulate/control Microsoft Excel and

SolidWorks using their API commands to work internally as if it were part of Visual Basic codes.

Based on the detailed system architecture of the project, the crucial API commands for each piece of software are depicted in Figure 10.

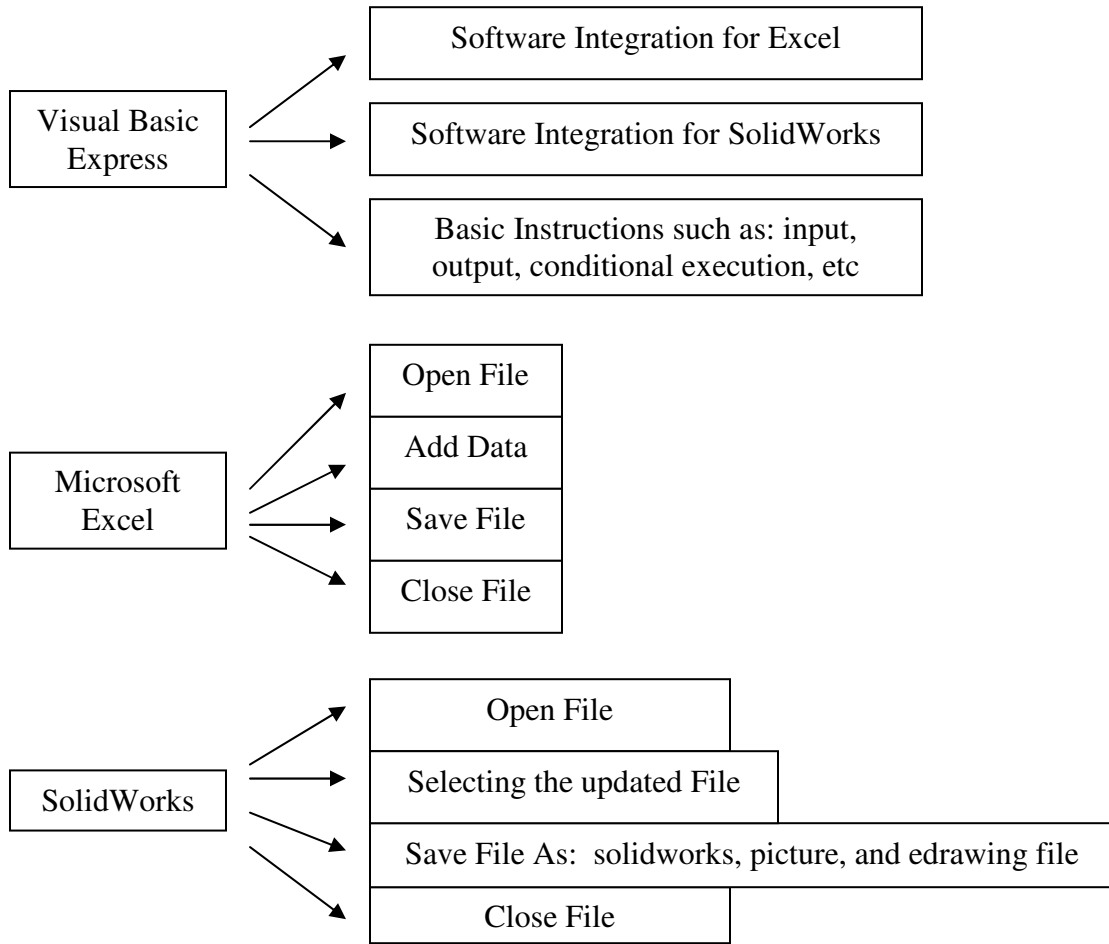


Figure 10. Crucial API commands

1.2 Software Familiarization. To become familiar with the software, essential API commands were studied in detail according to the software design. Manuals, books, and software websites were used to assist with software familiarization. It is during this phase that the detail of imperative codes mentioned above will be discuss and understand completely.

Using Visual Basic to establish the communication links between Microsoft Excel and SolidWorks was critical for the success of this project. To do this, both a SolidWorks and an Excel “type” library were referenced. A “type” library is where API commands are stored and referencing other software “type” libraries allows Visual Basic to access their API structure and control the software within Visual Basic. The location of the software type library is placed under the “Project Properties” as shown in Figure 11. For this project, Microsoft Excel Object Library 11.0 Object Library and SldWorks 2007 Type Library are then added at the “COM” tab as shown in Figure 12. With that, Visual Basic is now able to access SolidWorks’s and Microsoft Excel’s API commands.

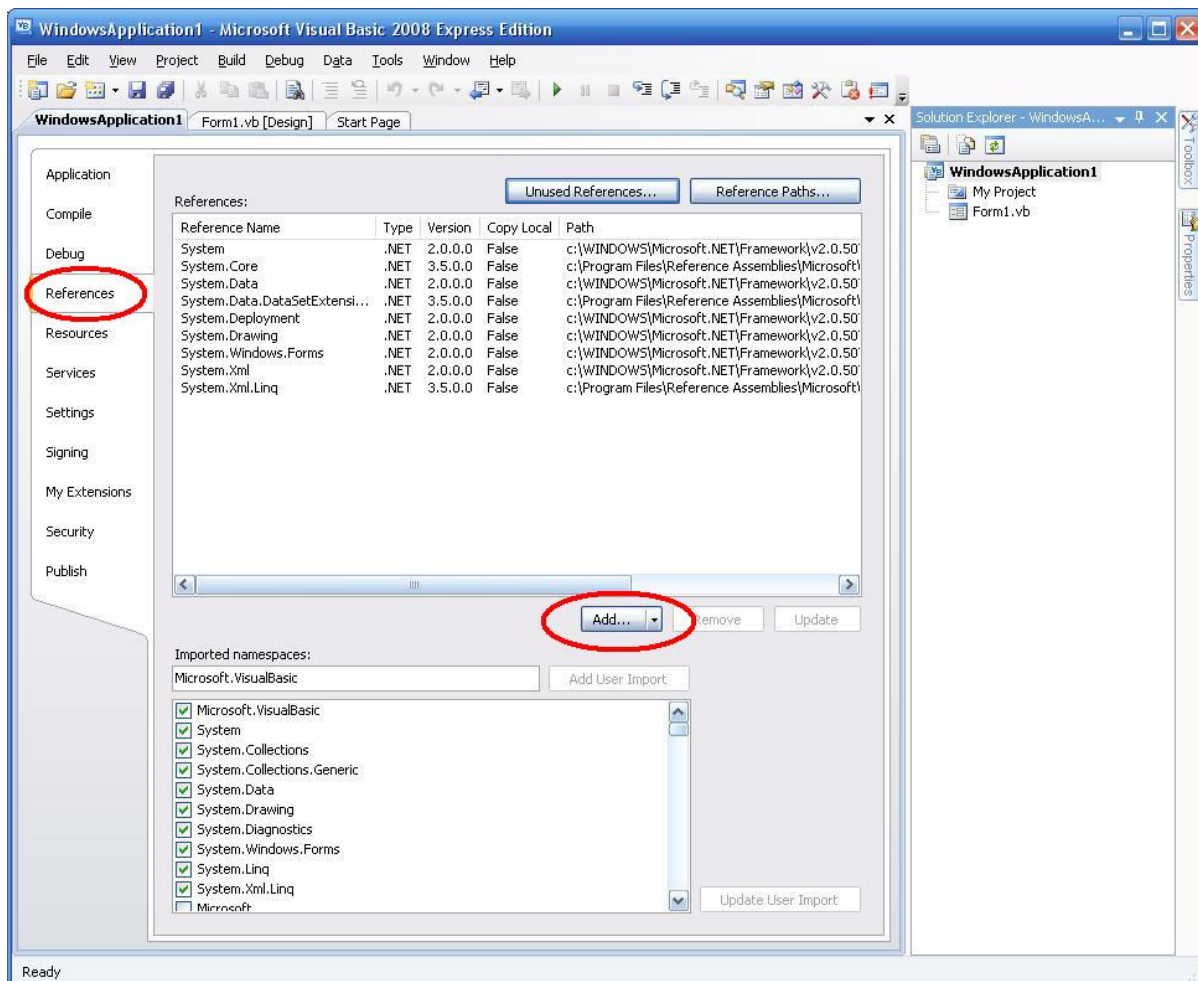


Figure 11. Project Properties in Visual Basic

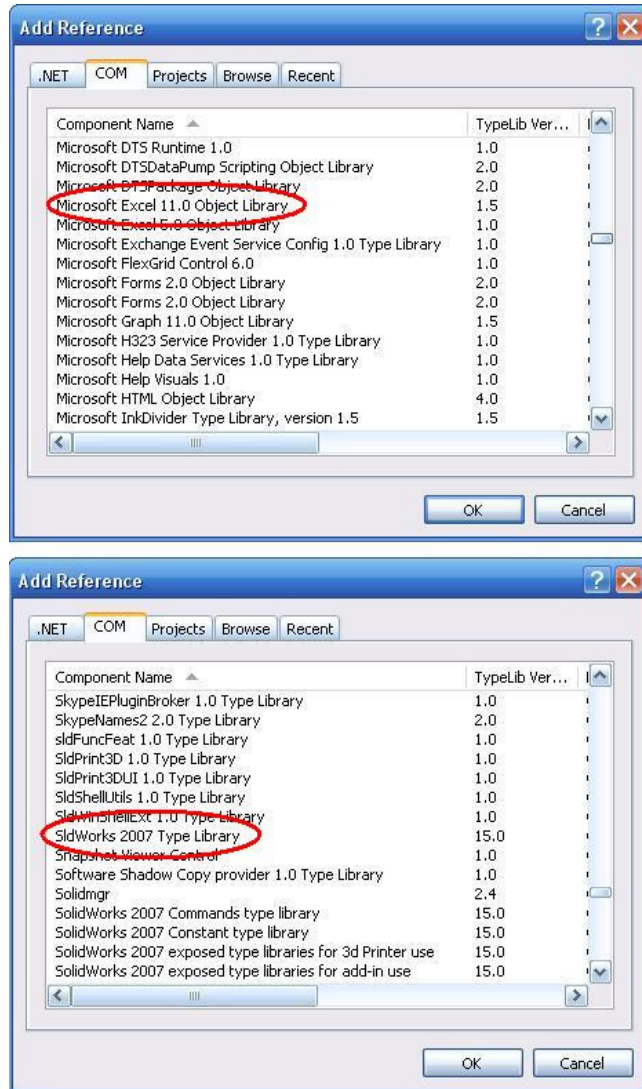


Figure 12. Adding both Excel and SolidWorks Type Library

To automatically enter the customer's parameter value from the user-interface form into the Excel, Visual Basic must use Excel API commands. The following are the techniques and codes used to enter the data automatically.

Since the Excel Workbook for the Design Table is saved somewhere in a folder, the following code was used to open a workbook file named "GuitarDesignTable.xls" located on drive D:

```
Workbooks.Open("D:\GuitarDesignTable.xls")
```

Once the Excel Workbook was opened, the parameter value can now be automatically entered by the following code:

```
Workbooks.Range("A1").Select ()  
Workbooks.ActiveCell.FormulaR1C1 = "53"
```

Entering the parameter value into a cell involves first selecting the desired cell and then passing the numbers into that cell. The example codes select the "A1" cell with a value of "53". Data can be entered with similar codes, with the exception of changing the cell selection and input value.

The Excel Workbook must now be saved and exited once the data has been entered so SolidWorks can use the updated Excel Spreadsheet. The codes are as follow:

```
Workbooks.Save()  
Workbooks.Close()
```

Since the Excel Spreadsheet is linked to SolidWorks, a new part model has been created according to the parameter values. To display the customized model back to the user-interface form, Visual Basic must use SolidWorks API commands.

When the new model is created, SolidWorks automatically displays a message box to inform the user that a new model was created, which requires an undesired human interaction. The following API commands were discovered to automatically skip the message box when opening the SolidWorks file.

```
SldWorks.OpenDoc6("C:\Telecaster Design Table.sldprt", 1, 1)
```

"OpenDoc6" is the code to open a SolidWorks file. The filename is located in the C drive with a filename of "Telecaster Design Table". The first numbers represent the type of document that is going to be opened as, which in this case is a *part* model. The second numbers represent the mode in which the document is opened as, in this case suppressing any dialog boxes.

Since the new part model is not automatically displayed in SolidWorks, API commands must be used to display the results of the user's interactions. The modified model needs to be selected with API command. The following are the codes to select the new configuration.

```
SldWorks.SelectByID2( "Filename", "CONFIGURATION")
```

"SelectByID2" is the code for selecting a specified entity. "Filename" is the name of the object that would like to be displayed. "CONFIGURATION" represents the type of object that needs to be presented, which in this case is a "configuration" option.

Once the new configuration model is displayed, the new model must now be saved as a picture file so it can be displayed on the user-interface form for the customer to see and compare. The following is the code to do that.

```
SldWorks.SaveBMP( "filename location", width, height)
```

Last is to close the SolidWorks file so whenever the customer decides to put in a new parameter value to the model, the SolidWorks file can be re-used. The code is as follows:

```
SldWorks.CloseDoc( "filename" )
```

Obviously there are many more codes involved in this project, but previously stated are the primary codes used for this project. For further information regarding the Excel and SolidWorks codes can be refer in Appendix C pg. 66 – pg.70.

2. Code Development. It is during this phase that the user-interface form and code implementation were created. Several important aspects must be carefully considered during this process. Some of these aspects are:

- Layout / Packaging – This aspect is mainly concerned with the user-interface form. User friendliness is the main criteria for the user-interface form. The

components and information of the form should be clear, easy to follow and understand.

- Reusability – The user-interface form components should be reusable in redesigning the model parts.
- Reliability – The software code should be able to perform according to the stated requirement.
- Robustness – The software code should operate and withstand an invalid input from the customers.

2.1 Designing user-interface layout. Visual Basic was used to establish a design layout for the user interface form. The basic layout of this project is shown in Figure 13 below.

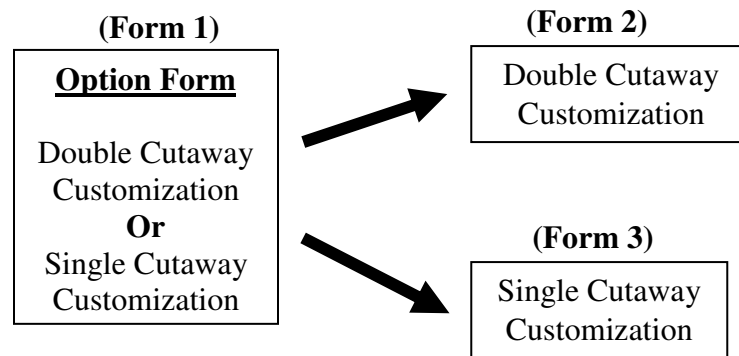


Figure 13. Basic Form Layout

The first form gives the user choice between customizing single cutaway or double cutaway guitar as shown in Figure 14. The layout of the first form is very straightforward. It shows two guitar models with two buttons from which the customer could choose to modify the model.



Figure 14. Welcome Form

Once the customer chooses which model they want to modify, they are brought to a different form. Both second and third layout forms are the same. They are divided into five sections as shown in Figure 15. The explanations are as follows:

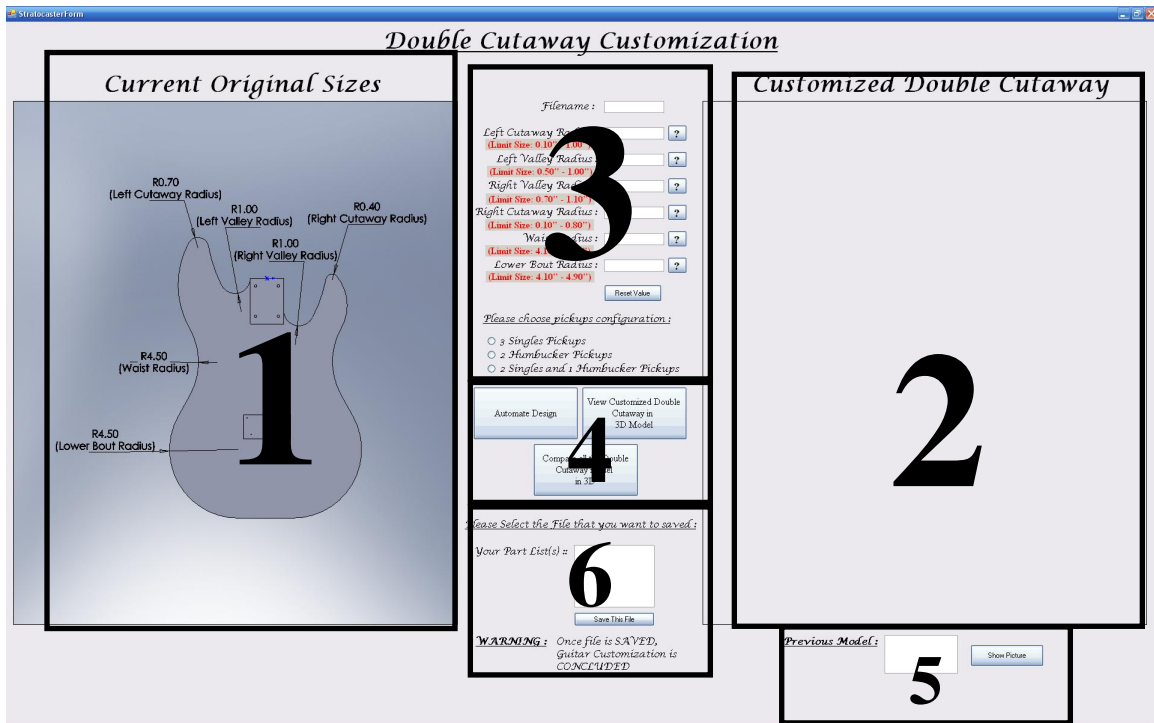


Figure 15. Customization layout form

The first section in the user-interface form is where the original model is displayed with the current dimension. This is to show the customer the original body shape. Labels on top of the picture box allow the customer to distinguish between the original and modified model. Moreover, the picture shows the specific types of customization that customers can make to specific areas of the guitar body. Once the model is customized, the second picture box will show the updated model with the current values of the customized model. These two sections purposefully comprise a large portion of the user-interface form to allow the customer to compare both models more thoroughly.

The screenshot shows a user interface for guitar customization. It features several input fields with labels and limit size information:

- Filename* :
- Left Cutaway Radius* : ?
(Limit Size: 0.10" - 1.00")
- Left Valley Radius* : ?
(Limit Size: 0.50" - 1.00")
- Right Valley Radius* : ?
(Limit Size: 0.70" - 1.10")
- Right Cutaway Radius* : ?
(Limit Size: 0.10" - 0.80")
- Waist Radius* : ?
(Limit Size: 4.10" - 4.90")
- Lower Bout Radius* : ?
(Limit Size: 4.10" - 4.90")

Below these fields is a button.

Please choose pickups configuration :

- 3 Singles Pickups
- 2 Humbucker Pickups
- 2 Singles and 1 Humbucker Pickups

Figure 16. Section 3 - Data input for the user-interface form

The third section of the form, shown in Figure 16, details the customer input section of the form. This section consists of labels, textbox, and radio button. The labels identify the parameters that can be changed by the user, as well as, the parameter ranges used for each

customization. The textbox is where the customer enters parameter values. A reset button is also provided if they decide to change all the parameters. Lastly, the customer is able to select guitar pickups configuration using a radio button input as well.

Figure 17 shows the fourth section of user-interface form consisting of three buttons that are available for providing additional programmed functionality.

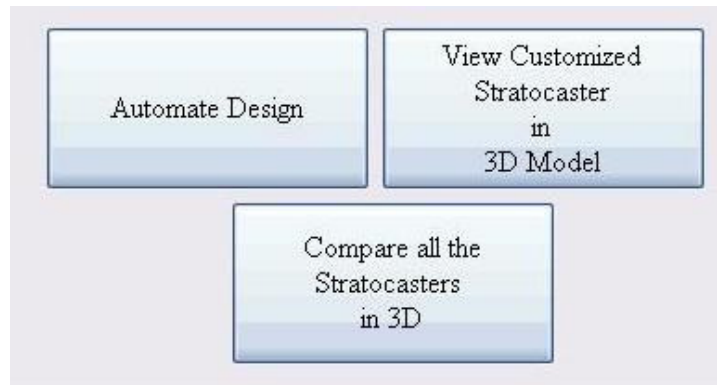


Figure 17. Section 4 - User-define form button

Once the customer is satisfied with his interactions and selections, he/she can select the “Automate Design” button to submit the values to the software application. The “Automate Design” button will transfer the customer’s parameter values into the Excel Spreadsheet, which in turn updates the model in the SolidWorks. This updated model will then be displayed in the customized picture box. If the customer would like to see the model in 3D, the “View customized model in 3D model” button can be selected. This will bring up a viewer window where the customer can inspect the part more thoroughly. If the customers are interested in comparing multiple configurations in 3D, they can select the “Compare all the models in 3D” button. This will bring up two viewer windows for the customer to compare the model side by side.

The fifth section of the user-interface form represents a list box and a button, as shown in Figure 18.



Figure 18. Section 5 - User-interface picture selection

Customers are allowed to do several iterations to customize the guitar. Each time the customer creates a new model, the model will be saved in the “Previous model” list box. When the customers are trying to remember the shape of their previous design, they can choose the previous model from the list box and press the “Show Picture” button and the model will appear on the above picture box for the customer to compare.

The last section of the user-interface form is shown in Figure 19.

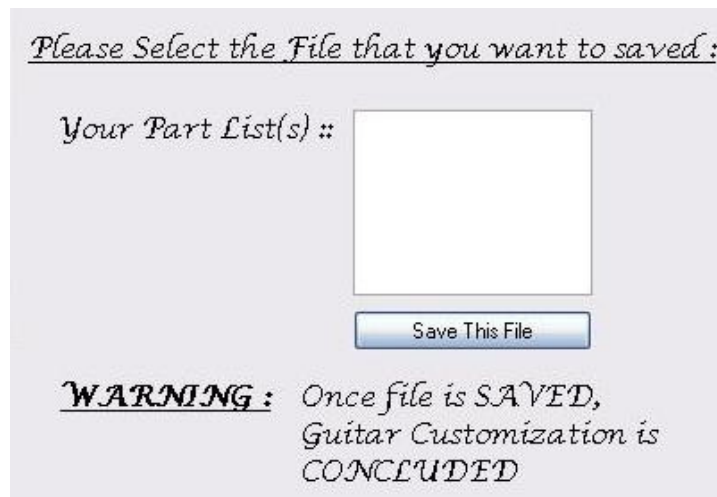


Figure 19. Section 6 - File Selection of the user-interface form

Similar to the previous section, the customer’s model is listed in the list box. The customers are only able to choose one model out of several iterations that they could have made. Once the customer selects the “Save this file” button, the file that they chose will be saved and the program will automatically close.

In general, designing these forms required a basic simplicity, where all the design elements are clear, easy to follow and understand by the customer.

2.2 Code Implementation and testing. During code implementation, many error-checking procedures were put in place to make sure that the codes are reliable and robust according to the expectation. Basic reasoning is implemented for error checking codes. Some of the implemented error checking will be discussed in this section.

All the textboxes in the user-interface form are protected with error checking codes to make sure that the customer cannot enter any invalid input. The customers are only allowed to enter numeric and decimal points into the textbox parametric values. Additionally, textboxes are not allowed to be left unfilled.

Since multiple users will access this software, the customers are allowed to enter a unique filename that will be placed in a special folder. The folder of this filename is routinely checked so if other users entered the same filename, it will be denied. Every user will have a specific filename that they can use.

Once the error checking codes were in place, evaluation and testing of the codes were able to take place to see if any additional error checking needed to be added. Selected volunteers were used to evaluate the reliability and robustness of the software and user-interface form and provide informal feedback.

3. Establishing Base Model. During the design procedure, a single and double cutaway guitar model was favored due to their well-known basic shape. Mottola (2009) used several parameters to fully describe an outline for the standard guitar model as depicted in Figure 20 below.

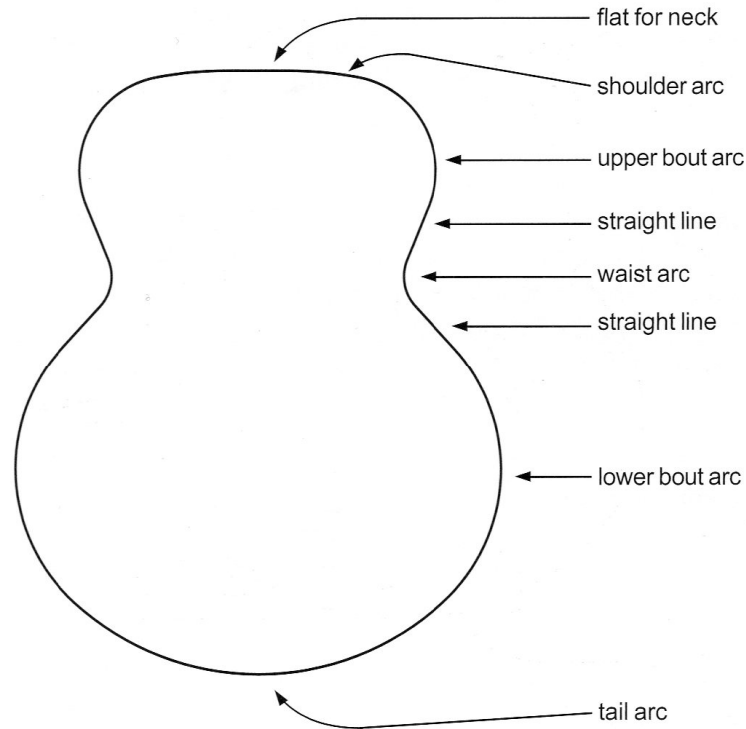


Figure 20. Standard guitar body model design features (Mottola, 2009)

The processes and parameters that are developed by Mottola are highly valuable for the purpose of this project. Unfortunately, given the constraints imposed by the wood blank used for this project, several of the feature parameters that are recommended by Mottola could not be used. However, his procedures used to design a guitar body outline are highly important and were used for this project.

3.1 Constructing a single cutaway model. During the body outline design for the single cutaway model, ratios of the guitar shape are checked and compared constantly to represent the shape of Fender's Telecaster style guitar (TSG). However, the most important criteria are that the parameters of this model have to satisfy the limit imposed by the wood blank. Unfortunately, as previously mentioned, given the constraint of the wood blank, the parameters and the ratio given by Mottola could not be met. Table 2 lists the guitar body outline parameters to fully describe the model. The techniques and procedure of the constructing the outline are discussed afterwards.

Construction lines for the guitar widths must first be drawn to mark the location for the circular arcs as depicted in Figure 21.

Table 2

Parameter values for Single Cutaway model construction

Overall Length	16.75"
Lower-bout width	12.50"
Lower-bout radius	5.00"
Waist width	9.00"
Waist radius	3.20"
Waist offset from tail end	10.70"
Upper-bout width	11.20"
Upper-bout radius	4.00"
Upper-bout offset from tail end	15.00"
Left Hill Cutaway	1.67"
Right Valley Cutaway	1.00"
Right Hill Cutaway	0.55"

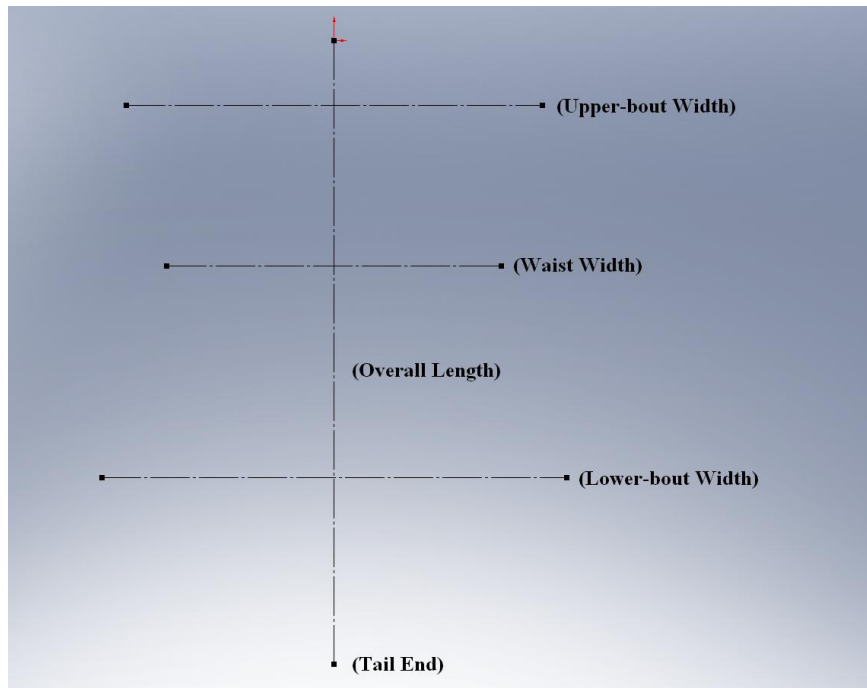


Figure 21. Construction Line

Using the aforementioned parameters, circular arcs were drawn and connected by tangent lines. This particular guitar model shape has a dead flat tail end and neck. To better visualize the

guitar model as a whole, these lines were then mirrored along the centerline as shown in Figure 22.

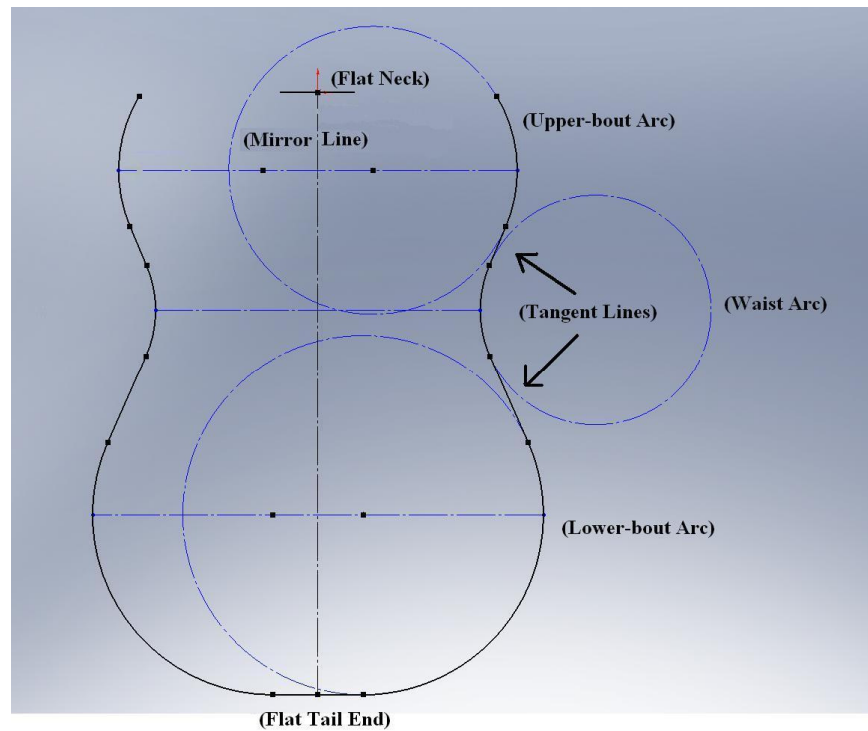


Figure 22. Basic Single Cutaway Form

To finish, the construction of the cutaways is drawn next. Again, the basic parameter shapes for the cutaways are attuned solely to (TSG). The right side cutaway is a little more intricate than the left side cutaway. Depending on the desired depth and width for the single cutaway guitar, the radius of the “right valley” cutaway can be adjusted and it is drawn tangent to the edge of the fingerboard. Meanwhile, the radius of the “left valley” cutaway is left constant due to the basic shape of TSG. Both “right hill” and “left hill” cutaways are placed tangent to the upper-bout arch. A tangent straight line is then drawn between the “valley” and “hill” cutaways. To make the guitar outline look more fluid, it is a good idea to make the cutaway radii as large as possible (Mottola, 2009). The final shape of the single cutaway guitar model is shown in Figure 23.

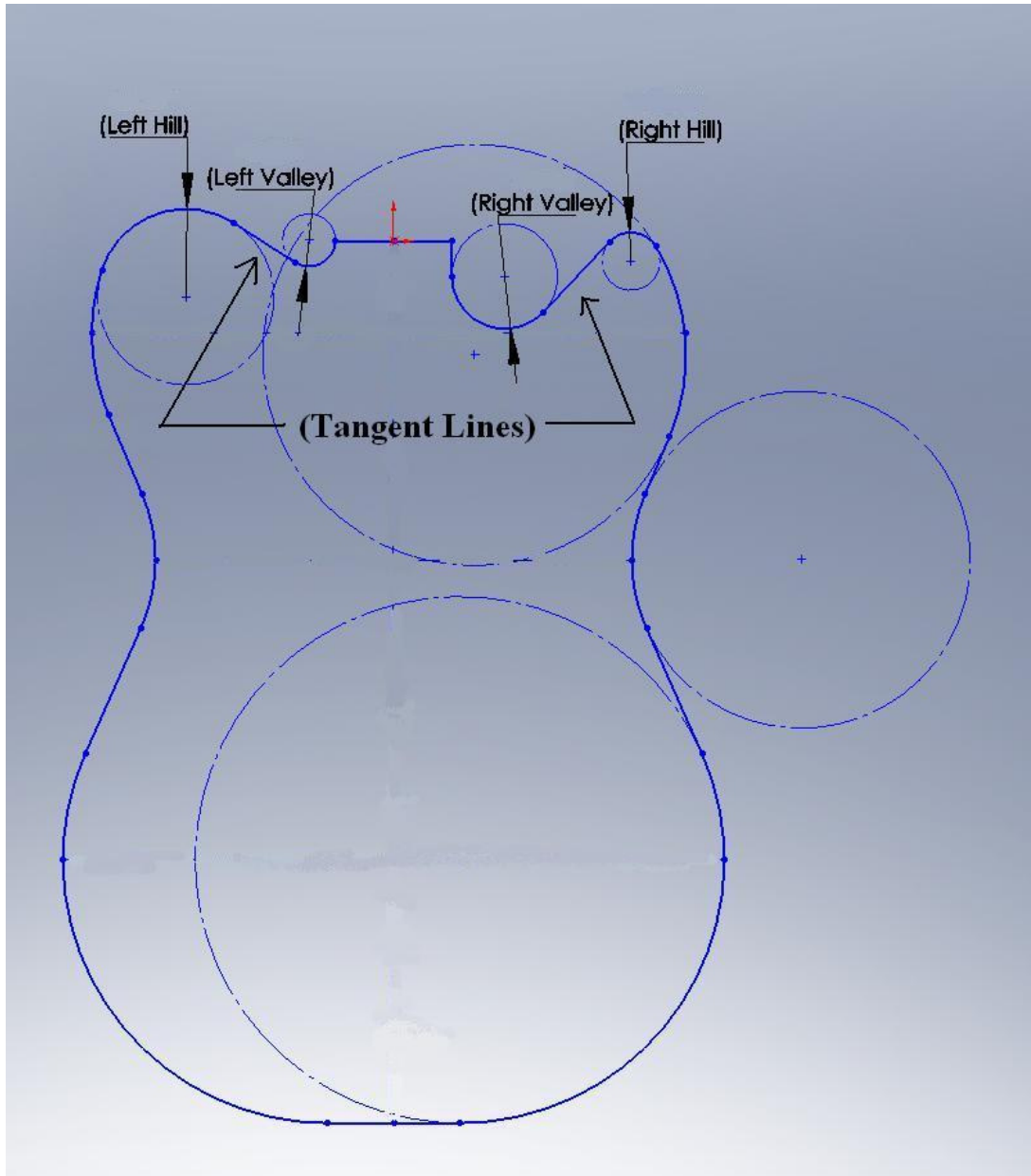


Figure 23. Single Cutaway Final Form

3.2 Constructing a double cutaway model. The shape for double cutaway model is based on Fender's Stratocaster style guitar (SSG). Similar with the single cutaway guitar body parameter, the ratios of the guitar shape are checked and compared constantly so to mimic the shape of a Stratocaster guitar from Fender. The parameters for the model is solely bound by the wood blank and listed in Table 3.

Table 3

Parameter values for Double Cutaway model construction

Overall Length	15.75"
Lower-bout width	12.62"
Lower-bout radius	4.50"
Waist width	8.80"
Waist radius	4.50"
Waist offset from tail end	10.27"
Left Hill Cutaway	0.70"
Left Valley Cutaway	1.00"
Right Valley Cutaway	1.00"
Right Hill Cutaway	0.40"

The technique for construction of the double cutaway model is similar with the single cutaway model but with a small twist. The upper bout arc and width is excluded during the initial construction of the double cutaway guitar because the SSG model has an asymmetrical cutaways shape. Only the lower-bout arc and waist arc, along with tangent lines are drawn to give the fundamental body shape of the Stratocaster model. Similarly, this particular guitar model has a dead flat tail end and neck. Figure 24 shows the initial construction of the double cutaway model with the aforementioned parameters.

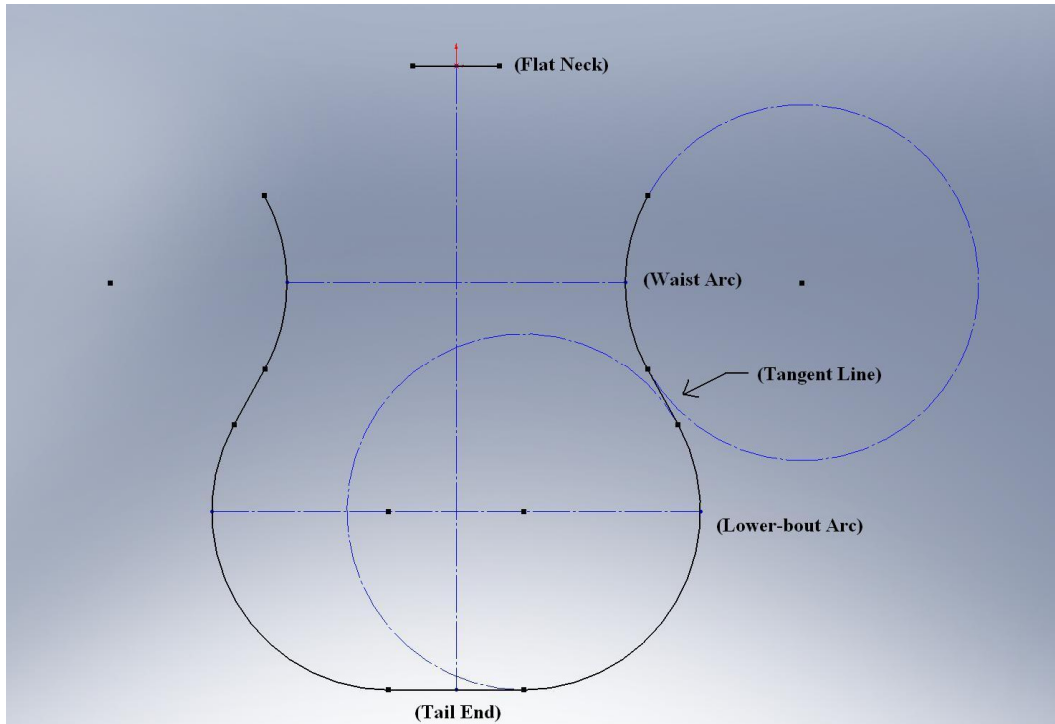


Figure 24. Basic Double Cutaway Form

The double cutaway guitar has a more pronounced cutaway than the single cutaway guitar. To make a more nuanced design, both cutaway “hills” are connected with secondary curves instead of a straight tangent line. The final shape of the double cutaway guitar model is shown in Figure 25 with the aforementioned parameters.

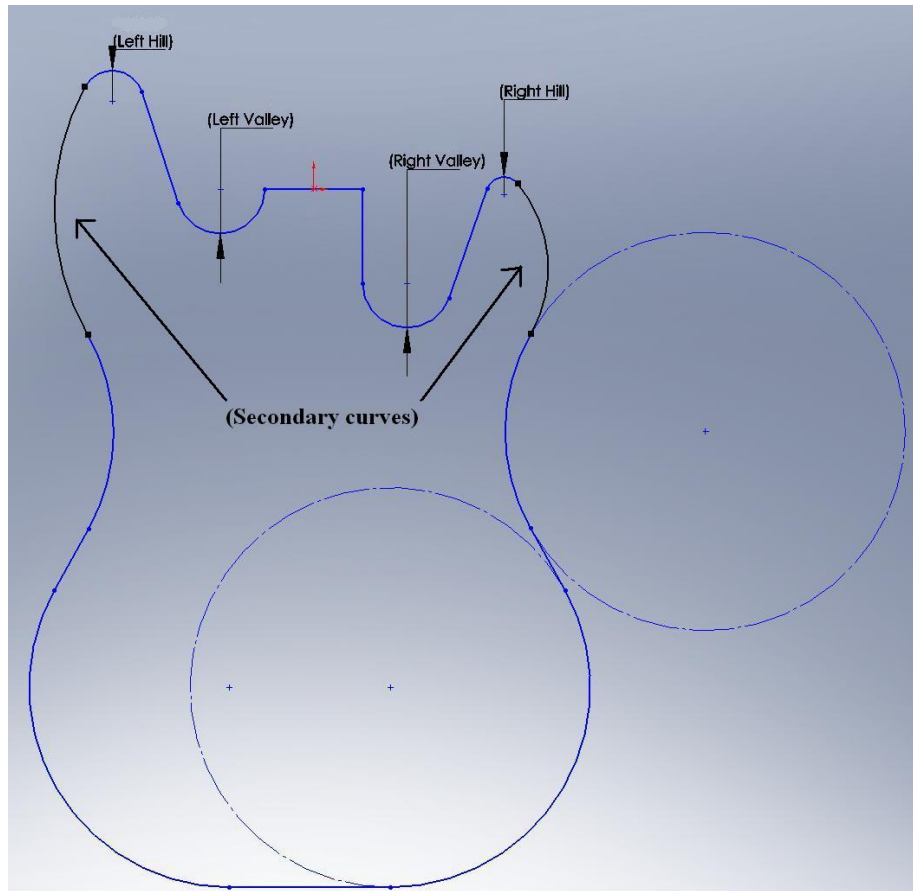


Figure 25. Double Cutaway Final Form

3.3 Determining the level of customization. The level of customization is highly dependent on the constraints that are applied to the model. Some values are mutually exclusive so some parameter values tend to affect each other. For instance, if no line was constrained, changing the waist arc will change the overall length of the guitar as shown in Figure 26.

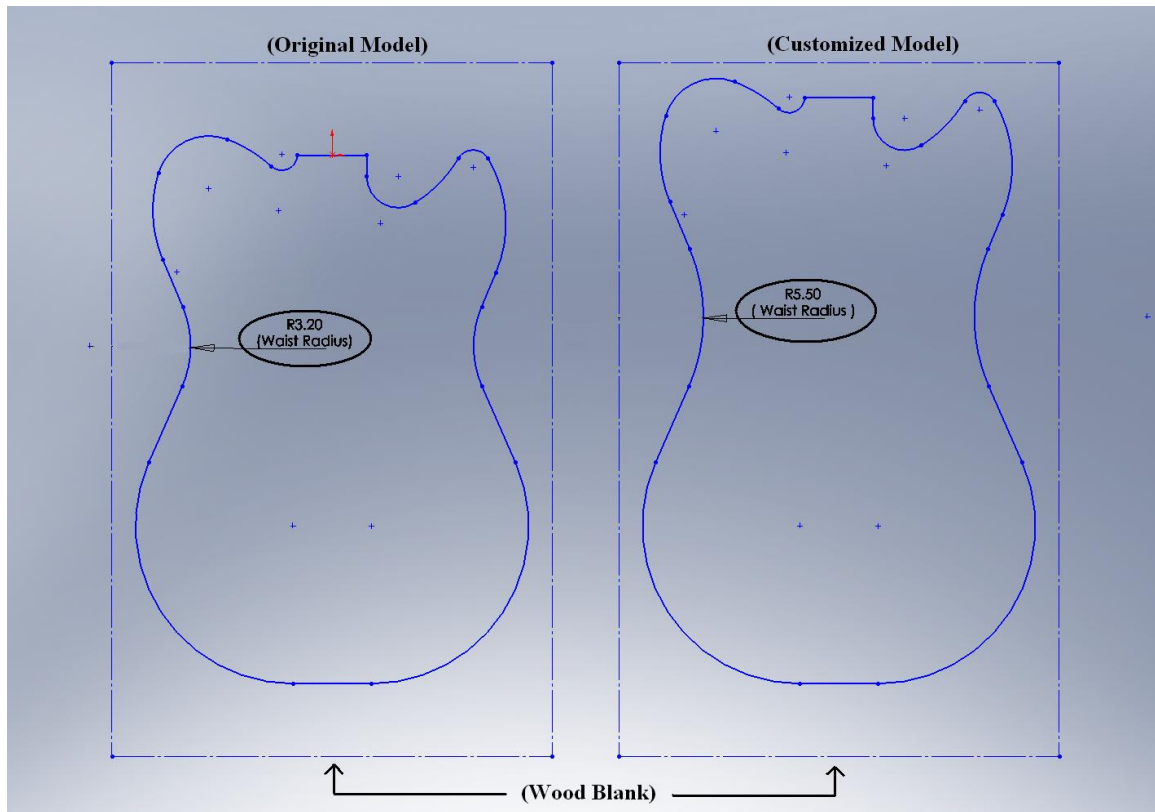


Figure 26. Waist arc customization with no constraint

Another example, if the tail end and flat neck are the only segments that are constrained, changing the waist arc will cause the model to be asymmetrical as shown in Figure 27.

Therefore, keep in mind that since each line of the drawing is tangent to its succeeding and preceding line, changing any parameter values will affect other segments of the model.

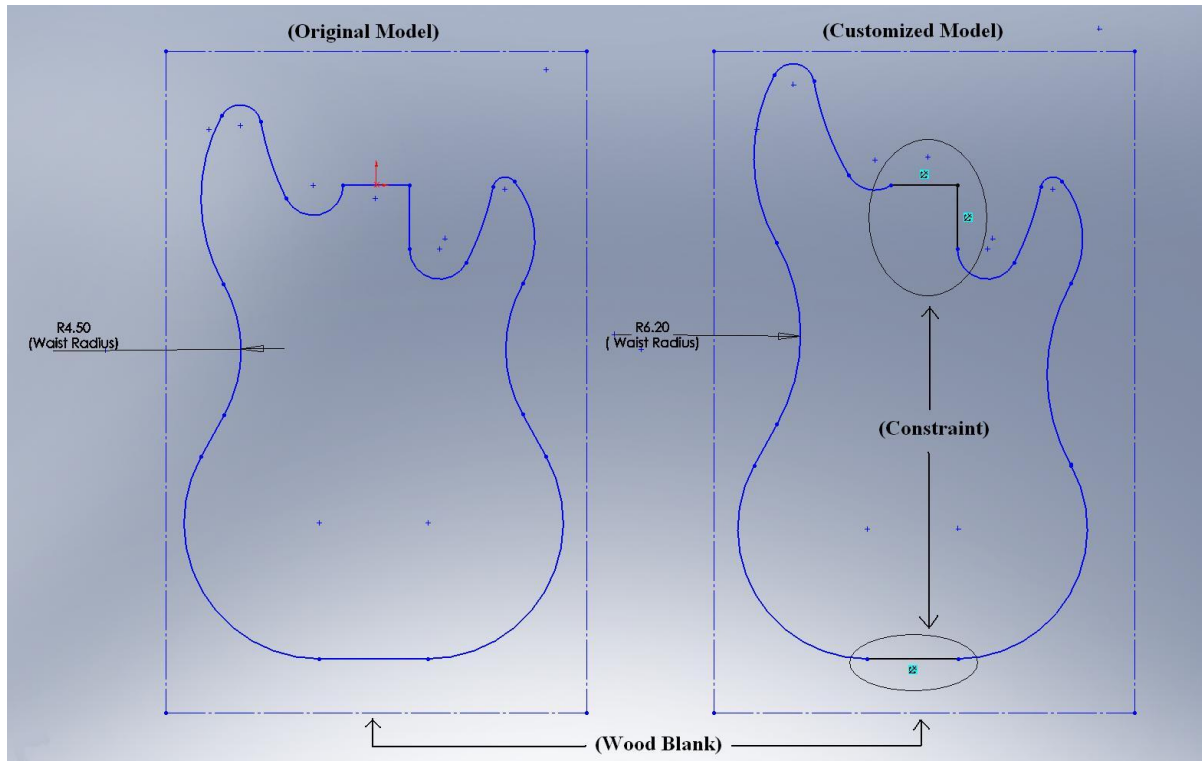


Figure 27. Waist arc customization model with flat neck and end tail constraint

The level of customization needs to be determined before making any decisions in constraining any segments. One of the requirements for this project is to be able to customize the cutaway shape. Any other customizations are welcome and it is counted as an additional advantage.

Both single and double cutaway guitar models have similar problems when determining what to constrain in order to be able to do cutaway customization. Changing the size of the cutaway without constraining any segment on the model will result in a change in the size of the neck flat as shown in Figure 28.

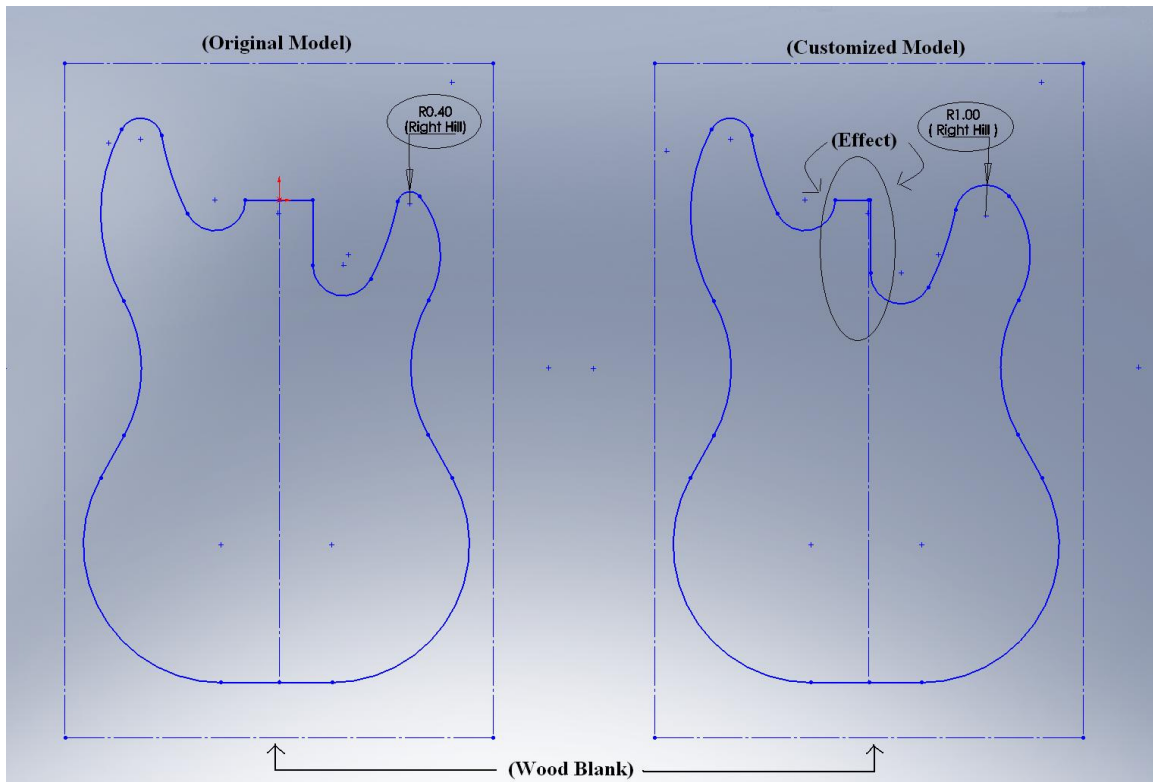


Figure 28. Cutaway Customization with no constraint

Since the size of the neck flat has to fit the actual guitar neck, putting a constraint on the neck flat is required. However, constraining the neck flat is not enough to customize the cutaway shape. Even though the neck flat is constrained, changing the cutaway shape may cause the guitar model to be asymmetrical. To solve this problem, the tail end segment has to be constrained as well. Yet, another problem still appears after constraining both neck flat and tail end when changing the size of the cutaway. The guitar width could shift unevenly when changing the cutaway size. Constraining either the waist arc or the lower-bout arc will cause the shape of the cutaway to be distorted. One final idea is to constrain the secondary curves on the Stratocaster and upper-bout arc on the Telecaster. Therefore, by constraining these three segments, the tail end, neck flat and secondary curves, as shown in Figure 29, almost all the

circular arcs can be customized to give the customers more options in customizing the model shape.

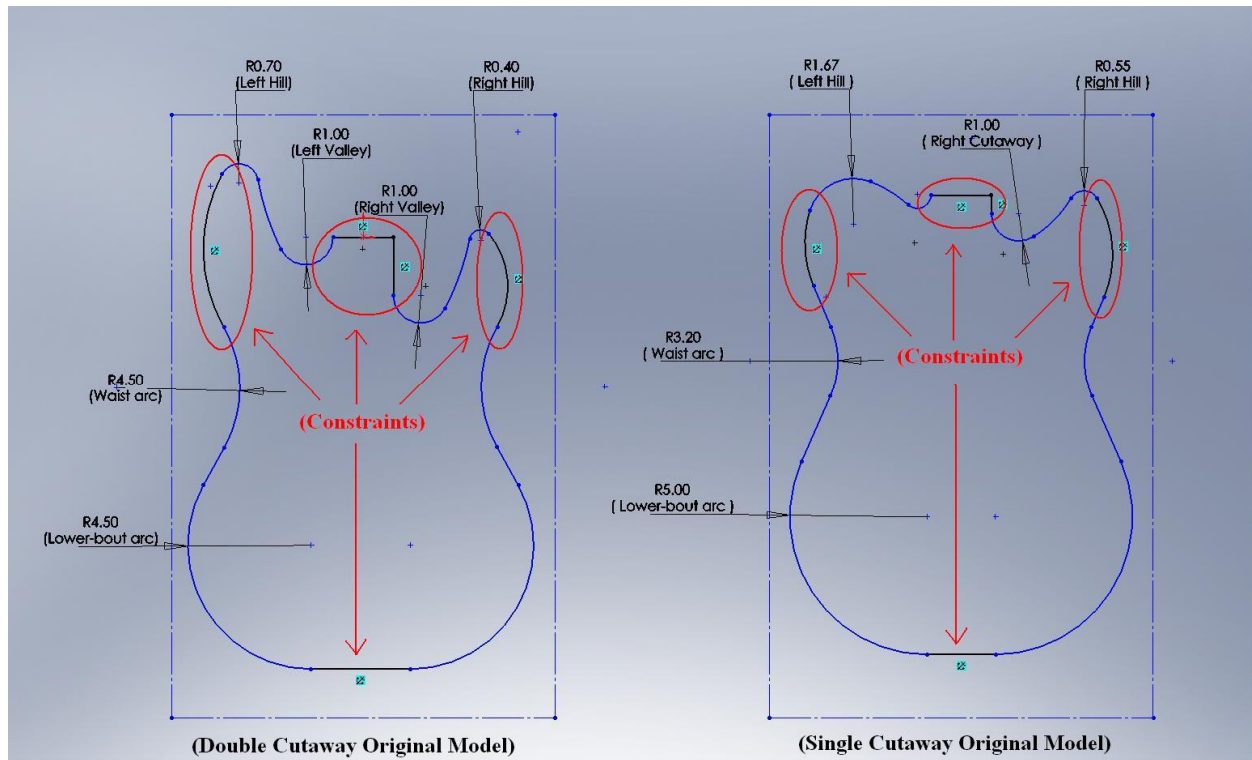


Figure 29. Ideal cutaway customization

With the chosen constraint segment, changing the cutaway size will not affect the width of the guitar. However, altering the size of the lower-bout arc and waist arc will change the width of the guitar.

3.4 Defining the range and customization limit for the customer. The size of the wood blank (22" x 14") plays a crucial role in determining the range and limit for the customization. Another limitation for establishing the range and limit for the customization, is figuring out the effect of changing each parameters size. Altering both lower-bout arc and waist arc sizes will have no effect in the guitar width if both sizes are changed the opposite way. The width of the guitar will be larger when both lower-bout arc and waist arc sizes are changed in the same

direction. Thus, this effect gives the range and limitation for both lower-bout arc and waist arc sizes.

Because of the applied constraints, changing the left side and the right side cutaway will not affect the other side of the cutaway. Furthermore, the change in cutaway will not affect the width of the guitar and therefore the only criterion of the limit and range for the cutaway is the shape of the cutaway. As long as the guitar shape is not distorted, the limit and the range for the cutaway can be established. Table 4 below lists the range and limit for the guitar customization.

Table 4

Limit and Range for guitar customization

Single Cutaway	
Left Hill Cutaway	1.30" - 2.24"
Right Valley Cutaway	0.70" - 1.10"
Right Hill Cutaway	0.10" - 1.00"
Waist Radius	2.80" - 4.00"
Lower-bout Radius	4.00" - 5.00"

Double Cutaway	
Left Hill Cutaway	0.10" - 1.00"
Left Valley Cutaway	0.50" - 1.00"
Right Valley Cutaway	0.70" - 1.10"
Right Hill Cutaway	0.10" - 0.80"
Waist Radius	4.10" - 4.90"
Lower-bout Radius	4.10" - 4.90"

Results

The purpose for this project was to provide customers with the ability to make design changes to the product without the need to possess any type of skills in CAD software by using a user-interface form. More specifically, this project is focused on changing a guitar model with simple parameter values. This project could help to lay the groundwork for the future of the guitar workshop at Purdue University, particularly on the design and manufacturing side of the workshop. It can also provide the participants the ability to make a more customized guitar for themselves.

With this project, customers could change the shape of the guitar, ranging from customizing the cutaway design to customizing the size of the guitar. Of course, certain limitations are in place so the shape of the guitar won't be distorted. The possibilities for this project are endless because changing the cutaway design would not affect the width of the guitar and vice versa. Figure 30 shows guitar model design results from the lower end of the limit provided and the upper end of the limit for both models.

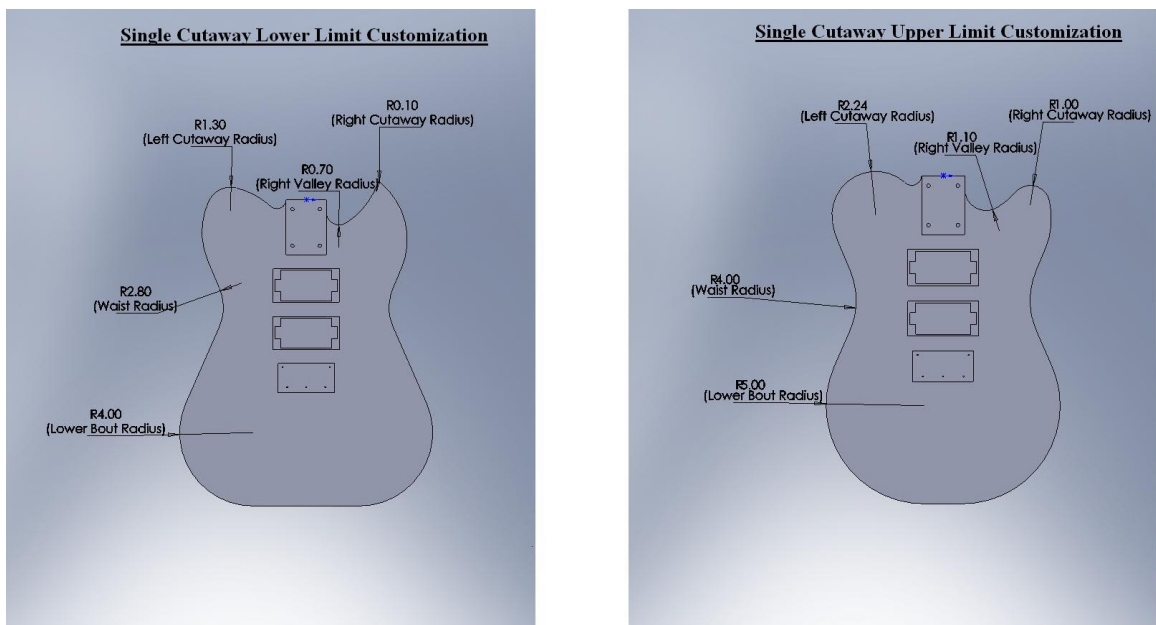


Figure 30. Single Cutaway lower vs. upper limit customization

As you can see in the model, depending on the parameter values, customers can customize the guitar shape, from a sharper cutaway style to a softer cutaway style, from a rounded tail end to a flatter tail end, and many other variations. Furthermore, the customer can choose different varieties of pickup style with a different style of body shape. Therefore, the possibilities for customization are considerable.

Conclusions and Future Improvement

The main objective of the project is to provide customers with the ability to make design changes to a product via a user-interface form without the need to possess of CAD software skills, bridging the gap between design engineering and the end user. Due to the personal investigator's association with the guitar workshop at Purdue University, the framework for this project is focused on the customization of a family of electric guitars. During the design process, the customer gives some parameter values for a model and the program automatically generate the model based on the parameters given. Based on the work done for this project, once the important criteria of the project are understood, the API commands that are needed to complete the project are relatively simple to determine.

The result of this study demonstrates that the level of customization greatly depends on the constraints that are applied to the model. The framework for this project is to focus on customizing the cutaway and the width of the guitar. Because of the constraints that are applied on the model, the cutaway does not change the width of the guitar and vice versa.

The possibilities of future improvement for this project are endless. Aside from changing the cutaway and the guitar size with parameter values, another improvement would be the ability to change the guitar body style, from single cutaway to double cutaway by segmenting the guitar shape into quadrants. With this configuration, the customer can choose which quadrant to modify based on several design choices.

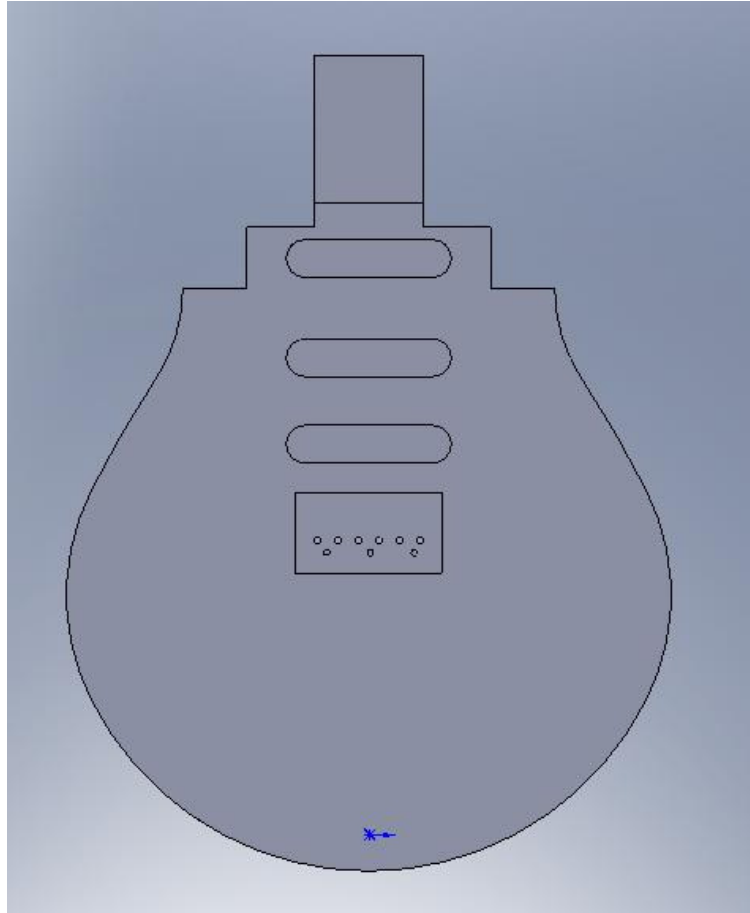


Figure 31. Possible Future Design by segmenting the guitar shape into quadrants

Figure 31 illustrates that the end users are able to choose the cutaway shape using the quadrant method. This improvement will allow the customer to have more options than before. Coupled with the work done for this project, the end users can also customize the cutaway shape after choosing the type of the cutaway style.

This project begins to lay the ground work for the future of the guitar workshop at Purdue University, particularly with the design and manufacturing side of the workshop. This project can be further improved by providing access to the software over the internet. The design interface developed will allow customers in future workshops to customize their guitar body style prior to arriving on campus.

In conclusion, while this project made some significant strides in product customization, there are plenty of opportunities for improvement within the project. Future expansion of this project could be applied to different brands of CAD software since most have similar capabilities and API command structures. Therefore, the possibilities are infinite with this type of project. The concepts developed for this project can be used for any family of products with a broad scope of design opportunities.

Reference

- Åhlström, P., & Westbrook, R. (1999). Implications of mass customization for operations management. *International Journal of Operation & Production Management*, 19 (3), pp. 262-274.
- Autodesk Inventor Object Library. (2009). *How do I access the API?*
- Coates, J. F., & Wolff, M. F. (1995). Customization promises sharp competitive edge. *Research Technology Management*, 38 (6), pp. 6-7.
- Dauner, J., Launder, J., Stimpfig, E., & Reuter, D. (1998). 3D Product presentation online: The virtual design exhibition. *Proc. VRML 1998*, Monterey, CA.
- Gattamelata, D., Pezzuti, E., & Valentini, P. P. (2006.). Using application programming interface to integrate reverse engineering methodologies into solidworks. *XVII congresso di Ingegneria Grafica INGEGRAF*, Barcellona, Spagna
- Gilmore, J. H., & Pine, B. J. (1997). The four faces of mass customization. *Harvard Business Review*, 75(1), pp. 91-101.
- Holweg, M & Pil, F. K. (2001). Successful build-to-order strategies start with the customer. *Sloan Management Review*, 43 (1), pp. 74-83.
- Keenen, B. (1999). VBA lets users add value to software applications. *Instrumentation & Control System*. 72 (1), pp. 55-8.
- Kumar, A. (2008). From mass customization to mass production: a strategic transformation [Electronic version]. *International Journal of Flexible Manufacturing Systems*, 19, 533-547.
- Lampel, J., & Mintzberg, H. (1996). Customizing Customization. *Sloan Management Review*, 38 (1), 21-30.

- Meyer, M. H., & Utterback, J. M. (1992). The product family and the dynamics of core capability. *Sloan Management Review*, 34, pp. 29-47.
- Mottola, R. M. (2009). A method for the design of the guitar body outline. *American Lutherie*, 97, pp. 52-61
- Piller, F. T. (2007). Observations on the present and future of mass customization. *International Journal of Flexible Manufacturing Systems*, 19, 630-636.
- Pine, B. J. (1993). *Mass customization: the new frontier in business competition*. Harvard Business School Press, Boston.
- Prince, S. P., Ryan, R. G., & Mincer, T. (2005). Common API: Using visual basic to communicate between engineering design and analytical software tools. *Proceedings of the 2005 American Society for Engineering Education Annual Conference & Exposition*, pp. 1939-1951
- Sanson, M. C. (2006). Use of an application programming interface (API) to allow non-optical designers to perform specific optical evaluations. *International Optical Design Conference*, 6342.
- Seppanen, M. S. (2000). Developing industrial strength simulation models using visual basic for applications (VBA). *Proceedings of the 2000 Winter Simulation Conference*, 1, pp. 77-82.
- Siddique, Z., & Boddu, K. R. (2003). A mass customization information framework for integration of customer in the configuration-design of a customized product. *Artificial intelligence for engineering design, analysis and manufacturing*, 18, 71-86.
- Wilson, J. (2007). Mass customization and build to order practices: an engineering perspective. *Cincom Systems Report*.

Yen, B. P-C., & Ng, K.Y.M. (2000). Web-based virtual reality catalog in electronic commerce.
Proceedings of the 33rd Hawaii International Conference System Sciences, Maui, Hawaii.

APPENDICES

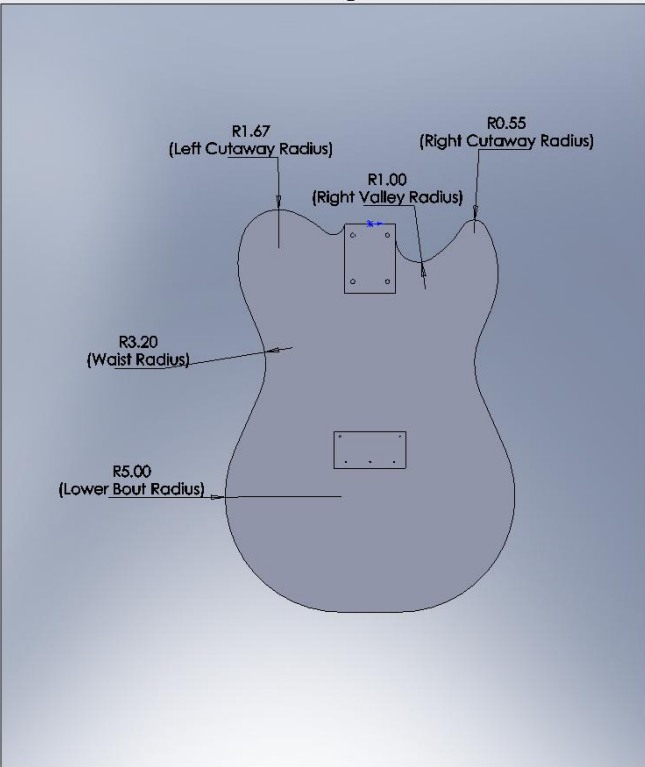
- A) Single Cutaway Customization Form
- B) Double Cutaway Customization Form
- C) Visual Basic Codes

Appendix A – Single Cutaway Customization Form

Telecaster Form

Single Cutaway Customization

Current Original Sizes



R1.67
(Left Cutaway Radius)

R0.55
(Right Cutaway Radius)

R1.00
(Right Valley Radius)

R3.20
(Waist Radius)

R5.00
(Lower Bout Radius)

Filename:

Left Cutaway Radius: ?
(Limit Size: 1.30" - 2.24")

Right Valley Radius: ?
(Limit Size: 0.70" - 1.10")

Right Cutaway Radius: ?
(Limit Size: 0.10" - 1.00")

Waist Radius: ?
(Limit Size: 2.80" - 4.00")

Lower Bout Radius: ?
(Limit Size: 4.00" - 5.00")

Please choose pickups configuration:

3 Singles Pickups
 2 Humbucker Pickups
 2 Singles and 1 Humbucker Pickups

Automate Design

View Customized Single Cutaway in 3D Model

Compare all the Single Cutaway in 3D

Please Select the file that you want to saved:

Your Part List(s) ::

WARNING: Once file is SAVED, guitar customization is CONCLUDED

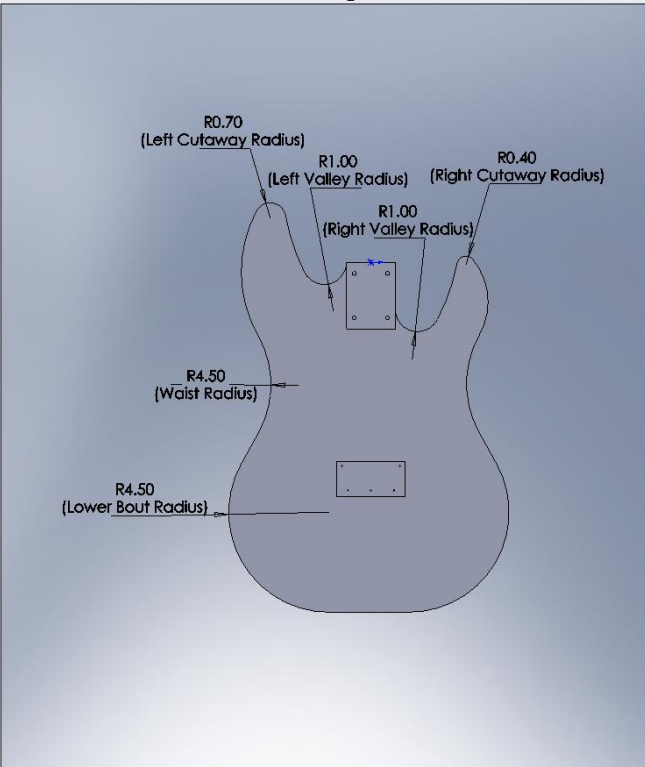
Customized Single Cutaway

Previous Model:

Appendix B – Double Cutaway Customization Form

Double Cutaway Customization

Current Original Sizes



Filename :

Left Cutaway Radius : ?
(Limit Size: 0.10" - 1.00")

Left Valley Radius : ?
(Limit Size: 0.50" - 1.00")

Right Valley Radius : ?
(Limit Size: 0.70" - 1.10")

Right Cutaway Radius : ?
(Limit Size: 0.10" - 0.80")

Waist Radius : ?
(Limit Size: 4.10" - 4.90")

Lower Bout Radius : ?
(Limit Size: 4.10" - 4.90")

Please choose pickups configuration :

3 Singles Pickups

2 Humbucker Pickups

2 Singles and 1 Humbucker Pickups

Please Select the File that you want to saved :

Your Part List(s) :

WARNING : Once file is SAVED, Guitar Customization is CONCLUDED

Customized Double Cutaway

Previous Model :

Appendix C – Visual Basic Codes

```

Public Class TelecasterForm

    Public solidworksFolder As String = "C:\Documents and Settings\Eddy Efendy\Desktop\Solidwork Files"

    Public mainFolder As String = "C:\Documents and Settings\Eddy Efendy\Desktop\Main Files"

    Public picsFolder As String = "C:\Documents and Settings\Eddy Efendy\Desktop\Picture Files"

    Public Sub FillTheListSld()

        lstTeleBox.Items.Clear()
        lstTeleBox.Enabled = False
        Me.Cursor = Cursors.WaitCursor
        Me.Refresh()

        For Each sldptrFile As String In My.Computer.FileSystem.GetFiles(solidworksFolder,
FileIO.SearchOption.SearchTopLevelOnly, "*.sldprt")
            Dim fileName As String = Replace(My.Computer.FileSystem.GetName(sldptrFile),
My.Computer.FileSystem.GetFileInfo(sldptrFile).Extension, "")
            lstTeleBox.Items.Add(fileName)
        Next

        Me.Cursor = Cursors.Default
        lstTeleBox.Enabled = True

    End Sub

    Public Sub FillTheListPic()

        lstTelePicBox.Items.Clear()

```

```

    lstTelePicBox.Enabled = False
    Me.Cursor = Cursors.WaitCursor
    Me.Refresh()

    For Each picFile As String In My.Computer.FileSystem.GetFiles(picsFolder,
FileIO.SearchOption.SearchTopLevelOnly, "*.JPG")
        Dim picfileNames As String = Replace(My.Computer.FileSystem.GetName(picFile),
My.Computer.FileSystem.GetFileInfo(picFile).Extension, "")
        lstTelePicBox.Items.Add(picfileNames)
    Next

    Me.Cursor = Cursors.Default
    lstTelePicBox.Enabled = True

End Sub

' ----- Limiting the KeyPress -----

Private Sub txtLeftCutawayRadius_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtLeftCutawayRadius.KeyPress
    If Not Char.IsDigit(e.KeyChar) And Not Char.IsControl(e.KeyChar) And Not e.KeyChar = "." Then
        e.Handled = True
    End If
End Sub

Private Sub txtLowerBoutRadius_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtLowerBoutRadius.KeyPress
    If Not Char.IsDigit(e.KeyChar) And Not Char.IsControl(e.KeyChar) And Not e.KeyChar = "." Then
        e.Handled = True
    End If
End Sub

Private Sub txtRightCutawayRadius_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtRightCutawayRadius.KeyPress
    If Not Char.IsDigit(e.KeyChar) And Not Char.IsControl(e.KeyChar) And Not e.KeyChar = "." Then

```

```

        e.Handled = True
    End If
End Sub

Private Sub txtRightValleyRadius_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtRightValleyRadius.KeyPress
    If Not Char.IsDigit(e.KeyChar) And Not Char.IsControl(e.KeyChar) And Not e.KeyChar = "." Then
        e.Handled = True
    End If
End Sub

Private Sub txtWaistRadius_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtWaistRadius.KeyPress
    If Not Char.IsDigit(e.KeyChar) And Not Char.IsControl(e.KeyChar) And Not e.KeyChar = "." Then
        e.Handled = True
    End If
End Sub

Private Sub txtFilename_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtFilename.KeyPress
    If e.KeyChar = "." Then
        e.Handled = True
    End If
End Sub

'-----Automate Button Control -----

Private Sub btnAutomateTele_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnAutomateTele.Click

    Dim filename As String
    Dim leftcutaway As String
    Dim rightvalley As String
    Dim rightcutaway As String
    Dim waist As String
    Dim lowerbout As String

```



```

Dim searchText As String = Trim(txtFilename.Text) & ".*"
Dim fileExists As Boolean = False

For Each a As String In My.Computer.FileSystem.GetFiles(solidworksFolder,
FileIO.SearchOption.SearchTopLevelOnly, searchText)
    fileExists = True
Exit For
Next

For Each b As String In My.Computer.FileSystem.GetFiles(mainFolder,
FileIO.SearchOption.SearchTopLevelOnly, searchText)
    fileExists = True
Exit For
Next

If fileExists Then
    MessageBox.Show("The file " & Trim(txtFilename.Text) & " already exists.", _
        "File Error", MessageBoxButtons.OK, MessageBoxIcon.Warning)
    txtFilename.Clear()
    txtFilename.Focus()
Else

    filename = txtFilename.Text
    leftcutaway = txtLeftCutawayRadius.Text
    rightvalley = txtRightValleyRadius.Text
    rightcutaway = txtRightCutawayRadius.Text
    waist = txtWaistRadius.Text
    lowerbout = txtLowerBoutRadius.Text

'----- ERROR CHECKING -----

    If filename = "" Then
        MessageBox.Show("Please enter a filename that you would like to SaveAs!", "ERROR!",
        MessageBoxButtons.OK, MessageBoxIcon.Error)
        Exit Sub
    End If

```

```
    If leftcutaway = "" Then
        MessageBox.Show("Please enter the Left Cutaway Radius!", "ERROR!", MessageBoxButtons.OK,
        MessageBoxIcon.Error)
        Exit Sub
    End If

    If leftcutaway < 1.3 Or leftcutaway > 2.24 Then
        MessageBox.Show("The Left Cutaway Radius must be more than 1.3 and less than 2.24!",
        "ERROR!", MessageBoxButtons.OK, MessageBoxIcon.Error)
        txtLeftCutawayRadius.Focus()
        Exit Sub
    End If

    If rightvalley = "" Then
        MessageBox.Show("Please enter the Right Valley Radius!", "ERROR!", MessageBoxButtons.OK,
        MessageBoxIcon.Error)
        Exit Sub
    End If

    If rightvalley < 0.7 Or rightvalley > 1.1 Then
        MessageBox.Show("The Right Valley Radius must be more than 0.7 and less than 1.1!",
        "ERROR!", MessageBoxButtons.OK, MessageBoxIcon.Error)
        txtRightValleyRadius.Focus()
        Exit Sub
    End If

    If rightcutaway = "" Then
        MessageBox.Show("Please enter the Right Cutaway Radius!", "ERROR!", MessageBoxButtons.OK,
        MessageBoxIcon.Error)
        Exit Sub
    End If

    If rightcutaway < 0.1 Or rightcutaway > 1 Then
        MessageBox.Show("The Right Cutaway Radius must be more than 0.1 and less than 1.0!",
        "ERROR!", MessageBoxButtons.OK, MessageBoxIcon.Error)
        txtRightCutawayRadius.Focus()
        Exit Sub
    End If
```

```

    If waist = "" Then
        MsgBox.Show("Please enter the Waist Radius!", "ERROR!", MessageBoxButtons.OK,
        MessageBoxIcon.Error)
        Exit Sub
    End If

    If waist < 2.8 Or waist > 4 Then
        MsgBox.Show("The Waist Radius must be more than 2.8 and less than 4.0!", "ERROR!",
        MessageBoxButtons.OK, MessageBoxIcon.Error)
        txtWaistRadius.Focus()
        Exit Sub
    End If

    If lowerbout = "" Then
        MsgBox.Show("Please enter the Lowerbout Radius!", "ERROR!", MessageBoxButtons.OK,
        MessageBoxIcon.Error)
        Exit Sub
    End If

    If lowerbout < 4 Or lowerbout > 5 Then
        MsgBox.Show("The Lowerbout Radius must be more than 4.0 and less than 5.0!", "ERROR!",
        MessageBoxButtons.OK, MessageBoxIcon.Error)
        txtLowerBoutRadius.Focus()
        Exit Sub
    End If

```

----- EXCEL CODE -----

```

Dim objExcel As New Excel.Application
Dim xlsWB As Excel.Workbook

xlsWB = objExcel.Workbooks.Open("C:\Directed Project (Solidwork and Visual
Basic)\Solidwork\Design Table\Telecaster with Design Table\Telecaster Design Table.xls")

objExcel.Range("A4").Select()

```

```
objExcel.ActiveCell.FormulaR1C1 = filename

objExcel.Range("B4").Select()
objExcel.ActiveCell.FormulaR1C1 = leftcutaway

objExcel.Range("C4").Select()
objExcel.ActiveCell.FormulaR1C1 = rightcutaway

objExcel.Range("D4").Select()
objExcel.ActiveCell.FormulaR1C1 = rightvalley

objExcel.Range("E4").Select()
objExcel.ActiveCell.FormulaR1C1 = waist

objExcel.Range("F4").Select()
objExcel.ActiveCell.FormulaR1C1 = lowerbout

If rb2Hum.Checked = True Then

    objExcel.Range("G4").Select()
    objExcel.ActiveCell.FormulaR1C1 = "U"

    objExcel.Range("H4").Select()
    objExcel.ActiveCell.FormulaR1C1 = "S"

    objExcel.Range("I4").Select()
    objExcel.ActiveCell.FormulaR1C1 = "S"

ElseIf rb2Singlesand1Hum.Checked = True Then

    objExcel.Range("G4").Select()
    objExcel.ActiveCell.FormulaR1C1 = "S"

    objExcel.Range("H4").Select()
    objExcel.ActiveCell.FormulaR1C1 = "S"

    objExcel.Range("I4").Select()
    objExcel.ActiveCell.FormulaR1C1 = "U"
```

```
ElseIf rb3Singles.Checked = True Then

    objExcel.Range("G4").Select()
    objExcel.ActiveCell.FormulaR1C1 = "S"

    objExcel.Range("H4").Select()
    objExcel.ActiveCell.FormulaR1C1 = "U"

    objExcel.Range("I4").Select()
    objExcel.ActiveCell.FormulaR1C1 = "S"

Else

    MsgBox.Show("Please choose which pickups configuration!", "ERROR!",
    MsgBoxButtons.OK, MsgBoxIcon.Error)
    Exit Sub

End If

xlsWB.Save()

xlsWB.Close()

objExcel = Nothing

xlsWB = Nothing
```

SOLIDWORK CODE

```
Dim swApp As SldWorks.SldWorks
Dim Part As SldWorks.ModelDoc2
Dim boolstatus As Boolean
```

```

Dim longstatus As Long, longwarnings As Long

swApp = New SldWorks.SldWorks()

swApp.Visible = True

Part = swApp.OpenDoc6("C:\Directed Project (Solidwork and Visual Basic)\Solidwork\Design
Table\Telecaster with Design Table\Telecaster Design Table.SLDPRT", 1, 1, "", longstatus, longwarnings)

Part.Visible = True

swApp.Visible = True

swApp.ActiveDoc.ActiveView.FrameState = 1

boolstatus = Part.Extension.SelectByID2(filename, "CONFIGURATIONS", 0, 0, 0, False, 0, Nothing,
0)

Part.ShowConfiguration(filename)

Part.ShowNamedView2("*Top", 5)

Part.ViewZoomtofit2()

Part.DeleteDesignTable()

Part.SaveAsSilent("C:\Documents and Settings\Eddy Efendy\Desktop\Solidwork Files\" & filename &
".SLDPRT", 1)

Part.ShowNamedView2("*Top", 5)

Part.ViewZoomtofit2()

Part.SaveBMP("C:\Documents and Settings\Eddy Efendy\Desktop\Picture Files\" & filename &
".JPG", 740, 872)

Part.SaveAsSilent("C:\Documents and Settings\Eddy Efendy\Desktop\eDrawings Files\" & filename &
".EPRT", 1)

```

```

        swApp.CloseDoc("Telecaster Design Table")

        swApp.ExitApp()

        picCustomizedTele.Image = Image.FromFile("C:\Documents and Settings\Eddy Efendy\Desktop\Picture
Files\" & filename & ".JPG")

        swApp = Nothing
        Part = Nothing

        btnTeleSaved.Enabled = False
        lstTeleBox.SelectionMode = SelectionMode.One

        FillTheListSld()

        btnRemindTele.Enabled = False
        lstTelePictureBox.SelectionMode = SelectionMode.One

        FillTheListPic()

    End If

End Sub

'----- 3D VIEW BUTTON -----

Private Sub btn3DViewTelecaster_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btn3DViewTelecaster.Click

    Dim userName As String

    userName = txtFilename.Text

    Dim StartEdrawing As New System.Diagnostics.Process()
    Dim StartEdrawingInfo As New System.Diagnostics.ProcessStartInfo()

```

```

StartEdrawingInfo.FileName = "C:\Program Files\Common Files\eDrawings2007\EModelViewer.exe"
StartEdrawingInfo.Arguments = "C:\Documents and Settings\Eddy Efendy\Desktop\eDrawings Files\" &
userName & ".eprt"

```

```

StartEdrawingInfo.CreateNoWindow = False
StartEdrawingInfo.UseShellExecute = False

```

```

StartEdrawing.StartInfo = StartEdrawingInfo

```

```

StartEdrawing.Start()

```

```

End Sub

```

----- RESET BUTTON -----

```

Private Sub btnTeleReset_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnTeleReset.Click

```

```

txtFilename.Text = ""
txtLeftCutawayRadius.Text = ""
txtRightValleyRadius.Text = ""
txtRightCutawayRadius.Text = ""
txtWaistRadius.Text = ""
txtLowerBoutRadius.Text = ""

```

```

picCustomizedTele.Image = Nothing

```

```

rb2Hum.Checked = False
rb2Singlesand1Hum.Checked = False
rb3Singles.Checked = False

```

```

btnRemindTele.Enabled = False
lstTelePictureBox.SelectionMode = SelectionMode.One

```

```

FillTheListPic()

```



```
    btnTeleSaved.Enabled = False
    lstTeleBox.SelectionMode = SelectionMode.One

    FillTheListSld()

End Sub

Private Sub lstTelePicBox_SelectedIndexChanged(ByVal sender As Object, ByVal e As System.EventArgs)
Handles lstTelePicBox.SelectedIndexChanged

    If lstTelePicBox.SelectedItems.Count > 0 Then
        btnRemindTele.Enabled = True
    Else
        btnRemindTele.Enabled = False
    End If

End Sub

Private Sub lstTeleBox_SelectedIndexChanged(ByVal sender As Object, ByVal e As System.EventArgs)
Handles lstTeleBox.SelectedIndexChanged

    If lstTeleBox.SelectedItems.Count > 0 Then
        btnTeleSaved.Enabled = True
    Else
        btnTeleSaved.Enabled = False
    End If

End Sub

Private Sub btnRemindTele_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnRemindTele.Click

    Dim picfileName As String = lstTelePicBox.SelectedItem

    Dim fullpicFromPath As String = picsFolder & "\" & picfileName & ".JPG"

    picCustomizedTele.Image = Image.FromFile(fullpicFromPath)
```

```
End Sub
```

```
Private Sub btnTele3DCompare_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles  
btnTele3DCompare.Click
```

```
Me.Visible = False  
TeleCompareForm.ShowDialog()
```

```
End Sub
```

```
Private Sub btnTeleSaved_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles  
btnTeleSaved.Click
```

```
Dim fileName As String = lstTeleBox.SelectedItem  
Dim fullFromPath As String = solidworksFolder & "\" & fileName & ".SLDPRT"  
Dim fullToPath As String = "C:\Documents and Settings\Eddy Efendy\Desktop\Main Files\" & fileName &  
".SLDPRT"
```

```
My.Computer.FileSystem.MoveFile(fullFromPath, fullToPath, True)
```

```
Me.Visible = False
```

```
MessageBox.Show("Thank you for Participating in Guitar Customization", "Thank you",  
MessageBoxButtons.OK, MessageBoxIcon.None)
```

```
Me.Close()
```

```
End Sub
```

```
End Class
```