2008-11-25

ECAD to MCAD Interoperability for Automated Enclosure Design

Adam C. Wilcox
Brigham Young University - Provo

ECAD TO MCAD INTEROPERABILITY FOR AUTOMATED ENCLOSURE DESIGN

by

A. Cade Wilcox

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

December 2008

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

A. Cade Wilcox

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date C. Greg Jensen, Chair

Date Kenneth W. Chase

Date Christopher A. Mattson

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of A. Cade Wilcox in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	C. Greg Jensen Chair, Graduate Committee

Accepted for the Department

Larry L. Howell
Graduate Coordinator

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of Engineering
and Technology

ABSTRACT

ECAD TO MCAD INTEROPERABILITY FOR AUTOMATED ENCLOSURE DESIGN

A. Cade Wilcox

Department of Mechanical Engineering

Master of Science

Enclosure design is the process of creating a package that will support and protect enclosed circuitry. Electronic enclosures are used in almost all industries, such as aerospace, automotive, naval, computer, toy, etc. Many designers use computer aided design (CAD) packages to aid them in creating these enclosures. Enclosure creation involves a working knowledge of the physics behind electrical and mechanical systems. Each of these engineering disciplines has separate CAD packages with their own set of rules, programming language, and interfaces. This creates a barrier for communication flow between electrical CAD (ECAD) and mechanical CAD (MCAD). The purpose of this thesis is to overcome the communication barrier by effectively transferring the knowledge contained in the ECAD package to the MCAD package, and use this information to aid in the electronic enclosure design process.

ACKNOWLEDGMENTS

I would like to thank Dr. Greg Jensen for all the time and effort he has put in to helping this thesis succeed. I also want to thank him for all the advice he has given me. I also want to thank Dr. Kenneth Chase and Dr. Christopher Mattson for their input. I especially want to thank my dear wife Carolyn for the support she has given and the sacrifices she has made to help me succeed. Thanks are also given to Belcan Corporation for the support given in this research topic.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

In an age where technology is becoming more and more complex, where customers demand higher quality and innovation is at its peak, there is a need for faster and more efficient design processes. Many companies are trying to find ways to automate their design practices and incorporate knowledge-based engineering into their product design process. “It is essential for the electronic component design and manufacturing industry to stay ahead of the competition and to provide the customer the best possible product in the shortest amount of time and at the lowest possible cost.” [Garfinkel, 1998]

When creating enclosure designs to house printed circuit boards (PCB), two engineering disciplines, electrical and mechanical, come together. This combination of disciplines is often called mechatronics. Engineers need to work closely together to create a safe and usable product. Decisions made by either engineer affect the design for the other. Therefore, proper communication between the two disciplines is the key to creating a quality product. For example, electrical components (capacitors, resistors, traces, etc.) must be far enough away from mechanical components (screws, mounting plates, brackets, the enclosure wall, etc.) so that current will not arc from the electrical component to the mechanical component. Information exchange between electrical and mechanical engineers is crucial when creating a mechatronic device.

1.1 Problem Statement

In today's industry many companies use high end CAD/CAE packages to solve many of the company's engineering problems. The CAD/CAE packages are very efficient for their respective disciple; "however, the weakest point in software tool implementation is still interfacing." [Garfinkel, 1998] There is currently no universal solution on how to communicate the information from one CAD/CAE package to another.

File transferring methods such as IGES, STEP, DXF, IDF, etc. have been partially successful; however, much of the useful attribute information in one CAD/CAE package may be lost in translation to the other package. This is due to the generalization of trying to match every CAD/CAE package to each other. Many Mechanical CAD (MCAD) packages have also tried to create "in house" Electrical CAD (ECAD) system, but these systems are not comparable to "stand alone" ECAD packages. Similarly, various ECAD packages have tried to incorporate MCAD functionality into its infrastructure, but have not been very successful.

1.1.1 Electronic CAD

"ECAD is fundamentally a two-dimensional layout tool." [Odell, 2002] It allows electrical engineers to place resistors, capacitors, integrated circuits, and other surface mount components on a PCB and then route wires and traces to make all the necessary connections and circuitry. ECAD systems, as provided by the vendors, contain default/built-in rules that guide the placement of the electrical components and wires. End users often customize the ECAD package by adding their own rules and methods.

This enables the user to better ensure higher quality and safer circuitry strictly within ECAD.

1.1.2 Mechanical CAD

MCAD, generally referred to by mechanical engineers as CAD, is a tool that creates parametric 3D models that are used for mechanical manufacturing and analysis. In this thesis I will refer to CAD as MCAD in order to differentiate between what is traditionally referred to as CAD and its ECAD counterpart.

1.2 Thesis Objectives

The objective of this research is to find solutions for the two main issues involved with mechatronic design. The first issue is the loss of useful information when communicating data from an ECAD package to a MCAD package. As was stated above, many file transferring systems try to incorporate all CAD/CAE software into its implementation and loses much of the data specific to company processes.

The second issue to solve is in creating a safe and quality electronic enclosure design in a more efficient manner. As will be discussed in Chapter 2, the current method of designing electronic enclosures can take months to create. At times, the current design process can take too long, and the product must be released before a completely safe and quality design is achieved. If the enclosure device is not properly designed, the circuitry inside could short, or become overheated, which could lead to injury or death. For this reason a new method in enclosure design has been researched and developed.

To solve these two issues, sufficient physical rules (i.e. voltage and thermal output values) from the ECAD package need to be translated over to the MCAD package, and not lost in communication. This additional information can then be used to drive the enclosure design and automatically generate a 3D model. By allowing the physical rules to drive the design process, more time can be allotted to the engineer to enhance the product and release a safe and quality mechatronic device within the product cycles deadline.

The success of this thesis will be based on three criteria. First, the amount of information communicated between the two CAD packages. Second, the reduction of time it takes to create a preliminary case design. Third, the quality of the enclosure design to ensure safety.

1.2.1 Enclosure Design Objectives

The design of the enclosure will focus on creating an inner void that enhances the amount of heat transferred from the PCB to the outside environment. This focus results in decreasing the distance between the enclosure and each component on the PCB. Decreasing the distance between PCB and enclosure not only increases the amount of heat transferred, but it also increases the risk of an electrical short. It should be noted that a great amount of research and development has been performed to minimize the risk of electrical shorting and will be discussed in detail in Chapter 4.

The majority of time invested in developing the enclosure design methodology was dedicated to the thermal and electrical aspects of enclosure design. Therefore the manufacturability and mass property aspects of enclosure design are not intended to be optimal. Chapter 6 discusses the further work that is needed in these areas to more fully create a useable and robust enclosure design.

1.3 Delimitation of the Problem

The methods described in this thesis are intended to be applicable for transferring data from any CAD/CAE package to any MCAD package. The implementations of the developed methods, as discussed in Chapter 4, will use one ECAD package and one MCAD package. The selected ECAD package is MentorGraphics Board Station and the selected MCAD package is Siemens NX5. It is assumed that the results of this methodology can be applied to varying CAD/CAE packages. Understandably, the file formats, data transfer names, data types, etc. might be different, but by following the methodology, an enclosure design can be achieved in any MCAD package.

The purpose of this research is not to create a complete final product, but to show that electrical and thermal rules from an ECAD system are capable of driving the mechanical design of an electronic housing in an MCAD system. Many times a fully designed enclosure has complex geometry and any number of features; however this research is not to create a fully designed product, but to prove a concept. Therefore, the models used to prove this idea will contain simple circuit boards, and enclosures.

An ideal solution to transfer data would be to create a tool that communicates bi-directionally. However, since the majority of the research will be performed inside the MCAD package the scope of the project is limited to a focus on the mechanical engineering aspect of the problem. A limited understanding of electrical circuitry within the ECAD package is necessary to know what information can be extracted.

1.4 Document Organization

The purpose of this chapter is to explain the problem statement and the objectives of the thesis. Chapter 2 contains a literature review of articles and white papers pertinent to this research. It also explores the design process of a typical electronics company as was witnessed by the author. Chapter 3 describes the methods developed and used for automated case generation. Chapter 4 discusses the implementation of the methodology defined in Chapter 3 specific to the research. Chapter 5 reviews the results of the thesis and determines whether the thesis was successful, according to the objectives defined in Chapter 1. Chapter 6 contains conclusions made about the results of the thesis and recommendations for future work.

2 Literature Review

To better understand the issues facing many electronic companies, literature was compiled and reviewed extensively. The literature that was acquired consists of journal articles, white papers, and previous theses. The first section of Chapter 2 defines the history of both ECAD and MCAD to help describe the communication barrier that exists between them. The next section explains, in detail, a current and typical electronic enclosure design cycle. The last section defines what is currently being done to overcome the problems described in the first two sections.

2.1 History of ECAD/MCAD

When CAD packages were first introduced there was not a concept of ECAD and MCAD, only CAD. Many saw CAD as an "answer to all problems". A National Science Foundation Report concluded that CAD "may represent the greatest increase in productivity since electricity." [Staton, 1985] Management within electronic companies believed that a CAD package could fully automate a PCB design. However, after using the software many companies found "the software [wasn't] capable of fully automating the board design process." [Salzman, 1989]

After CAD packages went through the initial "growing pains" many advancements were made to improve the automation process, and designers believed that

CAD had provided enough automation to perform many of the tedious tasks. [Salzman, 1989] As ECAD and MCAD developed and matured the functionality of the two diverged greatly. Mechanical engineers needed a CAD tool to create robust and parametric 3D models. Electrical engineers on the other hand were primarily concerned with 2D board layout designs. Over time the gap between these two domains has become increasingly wider.

Companies today have an increased need for quick and efficient product development methods. Feldman and Franke saw this gap as a major dilemma facing many companies. “To layout a dense mounted circuit board with a large variety of shapes of components the outside case has to be considered. However, conventional ECAD systems are not capable of analyzing collisions of spatial or even free-form bodies.” [Feldman and Franke, 1993] “In a recent online survey conducted by Manufacturing Business Technology, 88% of respondents say that such integration between design disciplines—known as mechatronics—is needed in their product development efforts. Yet most struggle with this synchronization.” [MBT 2007] Many other authors have shared this same concern. [PTC 2008] [Gayretli, 2007] [Breedveld, 2004] [Hsiao, 1999] [Zhou, 1997] [Wang 1996] A major ECAD developer (Mentor Graphics) has said that “Resources dictate that product integrations (often from competing suppliers that are not inclined to co-operate with each other) often remain poor.” [Mentor, 2007]

From the history described above it can be seen that the gap has created the need for better communication between the ECAD and MCAD domains. The lack of sufficient communication between an ECAD package and an MCAD package has plagued industry for many years. The communication gap is responsible for lost time, lower profits, lower quality products, and injury to end users. There is a need in today's industry to bridge the ECAD/MCAD gap for more reliable communication between engineering disciplines.

2.2 Current Design Process

There are many problems and frustrations that engineers experience everyday under the current enclosure design process. As was stated above, one of the biggest reasons for these frustrations is technical communication between electrical and mechanical engineers. There currently is not an effective method to communicate engineering rules from one branch of engineering to another. This will be demonstrated by a walkthrough of a current enclosure design process (Figure 2-1). This is the current process of one of the largest electronics companies in the United States.

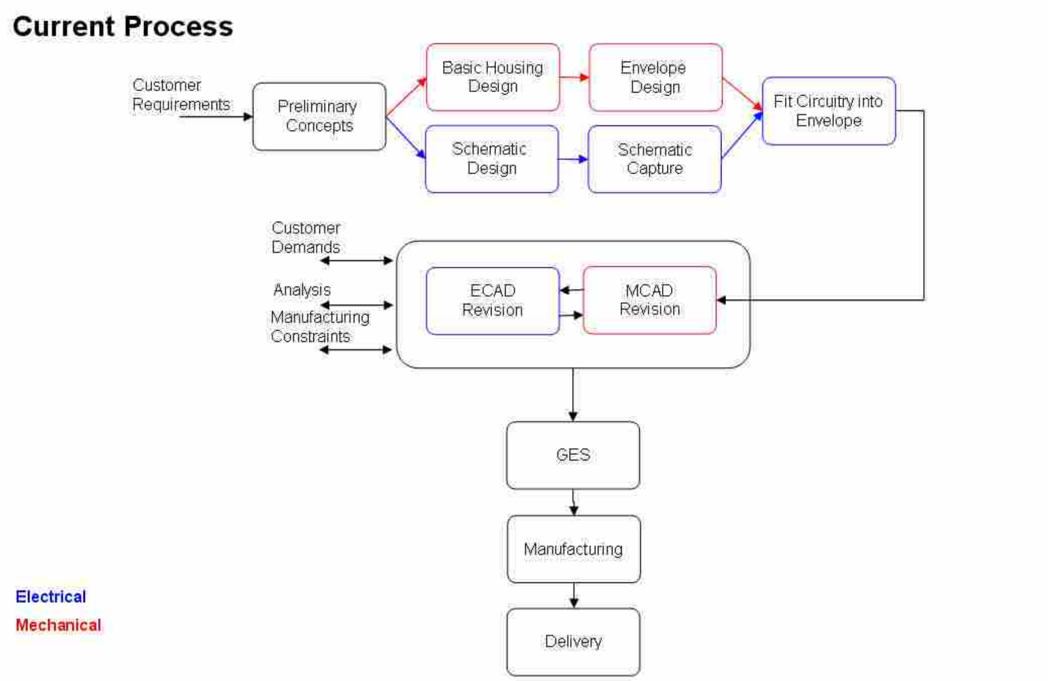


Figure 2-1: Typical Enclosure Design Process

This design process begins with the customer defining the specifications and requirements needed for the enclosure design. A team of engineers (electrical, mechanical, manufacturing, etc.) is then assembled and a plan is formulated to ensure the product is manufactured within time and budget. The electrical and mechanical engineers begin to create their respective designs in parallel to each other. The electrical engineer starts his design by creating a schematic layout of the circuit. He then creates the layout of each component as it will sit on the board assembly. Both of these processes are done with the ECAD package. The mechanical engineer is concurrently designing an enclosure envelope to house the circuit board and its components. The envelope is a rudimentary space definition that specifies where components can and cannot be placed. During this initial design phase the only form of communication between the two is through emails, phone calls, team meetings, etc. When the two are done they use a commercial software program that translates some ECAD information to MCAD information, to combine the two designs.

In most cases the electrical and mechanical designs don't fit together well and rework is needed.   While the designers iterate back and forth on making the board and enclosure fit together, another team is analyzing the parts for durability, heat transfer, voltage shorts, and circuit board vibration. Once the analysis is done requests for changes are made to enhance the design and the electrical and mechanical engineers must again modify their design. This iteration process can take months for a good board/enclosure design to be defined that is safe for the public to use.

At times this process takes so long that the design must be “frozen” (a term used to define when changes can no longer be made to finish the product on schedule). If this occurs, a high quality model is never created, and a faulty product is distributed to the customer. This becomes a major issue and can lead to recalls, injury and even death.

The two main issues that were discussed in Chapter 1 are easily understood by observing the current process. The first main issue is the loss of electrical information when it is translated to mechanical information. For example, electrical engineers not only place components on the PCB layout, but design the layout for traces that electrically connect components. During the translation process trace information is lost. To compensate for this the mechanical engineer has to rely on intuition and guesstimates to check for any problems the traces might have in shorting with the enclosure. If the engineer feels like there could be a problem with trace location they will obtain the data from the electrical engineer and draw the traces manually in MCAD. This could potentially take a lot of time considering there are hundreds and even thousands of traces on a PCB layout.

The second main issue is that the physical rules associated with the electrical analysis are not correctly communicated to the enclosure design. This leads to lower quality and unsafe products. In the ECAD domain the engineer designs in 2D space and is concerned with the layout of the components for electrical integrity of the design. The mechanical engineer, on the other hand, designs in 3D space and is concerned with spatial information and interferences of components. He then must obtain electrical data from the electrical engineer to verify the components are not too close to the enclosure walls to avoid shorting. This process is prone to human error and leads to faulty parts that may lead to injury or death.

The purpose of this section is to show the need for better communication when designing enclosures. The current process relies on outdated forms of data communication and design (i.e. phone calls, email, word of mouth, pencil drafting) which has proven to lead to many errors in the design process. By overcoming these problems a higher quality product can be released to the public in a more efficient manner.

2.3 Current Solutions

These dilemmas have existed for many years and companies have tried to overcome them in various ways. Currently there are two major solutions to help communication between engineering disciplines. These are: Intermediate Data Format file transfer systems (IDF) and Product Lifecycle Management systems (PLM). Both solutions attempt to overcome the communication barrier, but each is unique in its implementation. Each solution has its benefits and drawbacks, but neither is sufficient for the present demand in industry.

2.3.1 Intermediate Data Format

The Intermediate Data Format (IDF) file transfer system is an effective way to transfer information from an ECAD package to an MCAD package. An IDF file is an ASCII file format that contains ECAD information such as: component names, component location, component outline geometry definition, and board outlines. This data can then be translated over to the MCAD system to create 2 ½D representations of components and where they are placed on the board. 2 ½D representation means that the component is not fully defined in its third dimension, only the outline of the component and its overall height are known.

The IDF file format has become the standard for most companies to communicate between ECAD and MCAD software packages. Wonkee Ahn stated in his paper that "IDF translators exist for many, if not most [PCB] layout and mechanical design systems." [Ahn, 2007] IDF translators are mainly used to generate 2 ½D models in order to validate PCB designs. [Kowalski 2004] This validation is usually a visual check for interferences between PCB components and the enclosure housing them, and is efficient for some enclosure designs.

The IDF method is often used because it can be implemented by many CAD/CAE packages, this however is also its weakness. IDF translators have been so generalized to accommodate all CAD/CAE systems that information is lost. Garfinkel concludes in his paper that, "it is often found that some data within the file is output by MCAD, but is not recognized by ECAD and vice versa. All data lost or not carried in the translation need to reach ECAD (MCAD) by other means, usually on paper or orally, with the associated chance of error and delay." [Garfinkel, 1998] While the IDF file transfer contains some useful information that a MCAD designer needs, it does not include such vital information as: trace definition, voltage values, tolerances, thermal output, component pin placement, and 3D part information. This information resides in the ECAD package and, if fully transferred, could greatly increase productivity and reduce design time.

While the IDF method is successful in transferring some useful information, it currently does not allow sufficient information for enclosure design. Research has shown that IDF systems are susceptible to human error and a loss of data in translation.

2.3.2 Product Lifecycle Management

Another way companies are trying to overcome the communication barrier is through the use of Product Lifecycle Management (PLM) systems such as: Teamcenter, Windchill, and Enovia. These systems manage communications and information over the entire lifecycle of a product. A PLM system is a large database that is accessible to all engineering domains. It is a way for a company to keep product revisions organized and up-to-date. When changes or revisions are made the PLM software will send a notice to all who are affected by the change, which is crucial for any product design. In a white paper by PTC (the developer of Windchill) it states: "Tackling the integration of mechanical and electrical designs requires [that] everyone –regardless of their engineering specialty—must be informed immediately when designs are changed in a way that affects them. Notification should be done automatically."[PTC, 2008]

PLM systems not only help with revision control and notification, but they also unify the company as a whole. Engineers of all disciplines can work closely together for faster and more efficient design times. Ken Amman, director of research with analyst firm CIMdata says, "PLM vendors have developed workflows that address inclusion of software and electronics information, as well as integration with those tools that manage the other domains, whether it's electronic design automation tools or software development environments. We're beginning to see workflows expanding to incorporate those other aspects, and tie them into the overall workflow of the product life-cycle." [MBT, 2007] With a PLM system in place, many of the communication errors are eliminated and all engineering domains can have access to the most up-to-date design.

The drawback of PLM systems is that they do not directly link ECAD packages to MCAD packages. The information that is distributed to the mechanical engineer by the PLM system must be implemented into the MCAD system manually. Another drawback with PLM systems is the amount of time and effort it takes to set up and control them. Currently PLM systems are very hard to implement into company standard works.

PLM systems are therefore useful to product design cycles because of their ability to organize databases, control part revisions, and notify engineering personnel. This is beneficial for a product lifecycle, but information between software packages must still be performed manually. In other words, there is no direct translation of data from one software domain to another.

2.4 Literature Review Conclusions

The literature reviewed for this research has shown that currently there exists no universal solution to overcome the ECAD/MCAD communication gap. The barrier that has plagued the industry for years is still present and low quality enclosures are the result of it. There is a need for developing a new method with the ability to limit the probability of human error and enhance the amount of information being transferred between CAD packages.

3 Methods

This chapter contains a number of generalized methods that were used to overcome the issues discussed in the previous two chapters. They have been derived for this research to automatically create an enclosure around a PCB. The methods that are defined in this chapter are: File Parsing, Simple Polyhedron Geometry Creation, Assembling Parts, Tolerance Stack-Up, General Enclosure Creation, and a procedure for making the overall method extensible. These methods are described in the most general of terms and processes to allow their application of any part or assembly needing an enclosure. The implementations of these methods are described in Chapter 4.

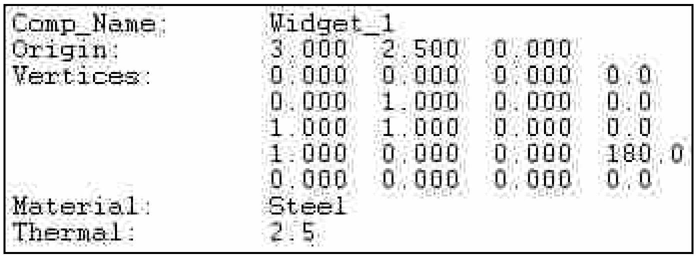
3.1 File Parsing

File parsing has been used for many years as a way to map information from one file formatting style to another. Each software program has a unique way of interpreting and using information, therefore it is important to match the file format to its respective software. As explained in Chapter 2, there currently exists a file transfer system (IDF) that is capable of transferring data from one CAD/CAE software system to MCAD. However, this data is limited to geometric definitions and placements. In order to more fully capture the information residing in CAD/CAE systems the following file parsing method is defined.

There are five basic steps used in file parsing which are:

1. Specify what information is needed for MCAD
2. Understand the file format
3. Establish "flags" that will be used to queue the parser
4. Parse the file line-by-line until you hit a flag
5. Map the data

For clarity in describing the method, an example file format is given in Figure 3-1. The example file is a widget that was defined in a 2D CAE package and now needs to be created in a 3D MCAD package. Once the process is complete, the widget should look like Figure 3-2. To more easily understand the example case, the CAE software we are extracting data from is called the "*from file*", and the MCAD software that will be using the extracted data is called the "*to file*". The desired data for the "*to file*" (MCAD data) will be represented as A, and the extracted data from the "*from file*" (CAE data) is represented as B. The final goal of the example will be to design the widget with enough surface area to act as a heat sink to transfer the heat generated from the widget to its surroundings.



```
Comp_Name:        Widget_1
Origin:           3.000  2.500  0.000
Vertices:         0.000  0.000  0.000  0.0
                  0.000  1.000  0.000  0.0
                  1.000  1.000  0.000  0.0
                  1.000  0.000  0.000  180.0
                  0.000  0.000  0.000  0.0
Material:         Steel
Thermal:          2.5
```

Figure 3-1: Example File

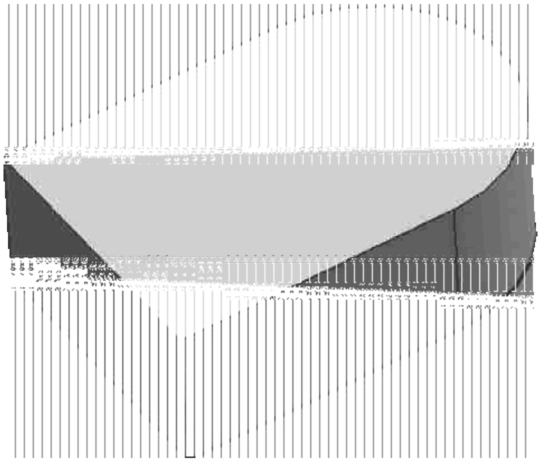


Figure 3-2: 3D Definition of Widget as Heat Sink

3.1.1 Specify Needed Data for MCAD

The first step in parsing a file is to specify which information is desired for the "*to file*" (A). It is crucial to completely understand what information is needed. For example, All MCAD packages need to know data that defines 3D geometry. Names of parts and assemblies are also vital for MCAD systems in order to organize and assemble parts. MCAD systems might also need additional attribute information to better describe a part or assembly.

In order to create the geometry for the widget in an MCAD system the units and geometric definition are needed. Since the goal of the design is to extract heat, it will be important to know the surface area needed to transfer sufficient heat off of the widget. It would also be useful to know the name or any other attributes that define the widget.

3.1.2 Understanding the Format

The next step is to decipher what format is being used to define the data in the "*from file*" (B). In order for the parsing process to work, the MCAD software must know the structure of the information contained within the "*from file*" so that it can be correctly read into MCAD. There are three different scenarios that should be discussed when understanding formats: Direct Mapping, Representative Mapping, and User Interfacing.

Direct Mapping: Direct mapping refers to data that can be mapped directly to the MCAD package without manipulating the data in any way. For example, all CAD/CAE packages use point information to define the vertices of the geometric objects defined in its system. Each point data is described as an X, Y, and Z coordinate location. The point definition will always be the same, whether transferring data from a 2D CAD package (Z value=0) to a 3D CAD package or vice versa, the process is the same. Because the output data format is the same as the input, there is no need to manipulate the data, this is what is meant by direct mapping. Other common direct mapping formats could include names, curve geometries, mathematical equations, or other geometric definitions.

The widget example shows data that can be directly mapped to the MCAD system. The name, origin, and vertex information are all data formats that are understood by the MCAD system. This information can be directly mapped into MCAD to create the base geometry of the widget.

Representative Mapping: Representative mapping refers to data that must be interpreted into a format that can be understood by the MCAD package. For example, if converting information from an FEA package, stress is meaningless to MCAD. However, this same stress data can be translated to be a deformation length, or a topology change in the geometry. Representative mapping can become very useful if correct principles are used, and is fundamental to knowledge based engineering.

For example the widget contains information that describes the widget material and the maximum heat it generates. This information by itself is meaningless to a MCAD package; however using heat transfer equations the height of the widget can be defined and the total surface area of the heat sink can be determined.

User Interfacing: At times the information that is needed for MCAD is not found in a file from another CAD/CAE system. When this occurs it is necessary to gather the needed data from the user. There are various means of prompting the user to input data, but the most common method is through graphical user interfaces (GUI's). A properly designed GUI is easy to use for all users, whether they are experienced or novice, and gives little to no room for erroneous data entry.

The widget example file provides information about the origin, and vertex points and the thermal output of the component; however it says nothing about the units the widget is designed in. In this case a simple GUI can be constructed to prompt the user to specify the unit information.

3.1.3 Establishing Flags

The third step of this method is to use the file formatting as a guide to flag or tell the parser when it has found the needed information (B). This step in the process is only useful for direct or representative mapping. There is no need for flags when prompting the user for information. A flag could be a key word, a specific character, a tabbed space, a number, and so on.

In the case of the widget example, the flags would be the names that are preceded by a colon (Comp_Name, Origin, Vertices, etc.). These keywords, followed by a colon, will tell the parser that the information following the colon is important and should be mapped to the MCAD package.

3.1.4 Parsing Line-by-Line

The fourth step is to read each line of the "*from file*". Again this step is only used for direct and/or representative mapping. A parser is usually designed to only read a file a line at a time. As the characters on each line are interpreted by the parser it constantly checks to see if those characters being read form an established flag. If it encounters a flag, it knows that the following characters are to be mapped to the MCAD package.

Following the widget example, the first line read in would tell the parser that the component name is given. Once it reads in the colon, the name Widget_1 is mapped to the MCAD file. The next line parsed would flag the parser to map the origin data to the MCAD package. This process continues until it reaches the end of the file.

3.1.5 Map the Data

When a flagged format is read in, the data is gathered and can then be mapped to the "*to file*" variable (A). If the data can be directly mapped, then a simple equivalence equation can be used. The widget example shows that A representing the component name, origin, and the geometry definition are directly equal to what it is in B.

$$A = B \tag{3-1}$$

If the data must first be interpreted by the parser for mapping than the B data is run through an interpreting function. The interpreting function must use a known and well defined physical equation that is understood by the MCAD user. For the widget example, A signifies the surface area and *f*(B) represents the heat transfer equation. The material and heat data taken from the "*from file*" are the inputs into this equation.

$$A = f(B) \tag{3-2}$$

If the needed information cannot be extracted from the "*from file*", then it must be entered manually by means of a GUI. The GUI should be well defined and easy to use so the user can enter the desired data. The A data represents the unit information needed for the widget example and B would be provided by the user via a GUI.

$$A = GUI(B) \tag{3-3}$$

By following the file parsing method, data is successfully exchanged from one style of file formatting to another. The widget in the example can now be created inside the MCAD package and the geometry will be created to enable the transfer of sufficient heat away from the widget.

3.2 Simple Geometry Creation

As explained in the previous section, all CAD/CAE packages use points to define geometric objects. The point data that is extracted from the CAD/CAE package is used to create a representation of the CAD/CAE object inside of the MCAD package. This representation can be defined as a polyhedron (solid model) inside of the MCAD system. A definition of a polyhedron is provided to better understand the method of solid model creation.

A polyhedron is often defined by the number of faces it contains (i.e. a tetrahedron has 4 faces; a hexahedron has 6 and so forth). Closed polyhedrons can be defined using Euler's formula, which expresses a relationship between the number of vertices, edges, and faces of a polyhedron:

$$V - E + F = 2 \tag{3-4}$$

For the equation, V is the number of vertices, E is the number of edges, and F is the number of faces. Figure 3-3 shows that each of the polyhedrons shown satisfies Euler's formula. It is also important to note that curved edges and faces can also be used in creating a polyhedron.

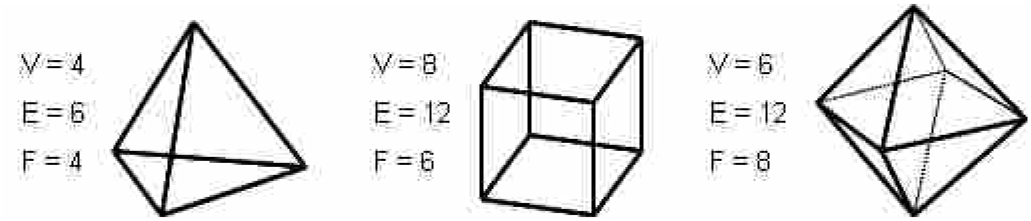


Figure 3-3: Simple Polyhedra with V, E, and F Values

The method behind creating a polyhedron begins by knowing the X, Y, and Z coordinates of each vertex. These values are gathered during the file parsing process. Then edges are defined by creating a curve (straight line, arc, spline, etc.) between two of the vertices. The edge information is also gathered during the file parsing process. If this information does not exist in the CAD/CAE system, then a straight line is assumed to be the edge. It is important to know which two vertices are the start and end points of each edge. The faces are then defined to be the region bounded by a set of coincident edges, and similarly, the solid body of the polyhedron is defined to be the region bounded by a set of contiguous faces.

The file parsing section contained a simple example of a widget being extracted from a 2D CAE system. The vertices of the widget were directly mapped to the MCAD package through file parsing. The vertices can then be used to create the geometry of the widget shown in Figure 3-2 by following the method of creating polyhedra.

Another important topic to cover when talking about solid modeling, is the differences between geometry and topology. Geometry is the metric information or actual dimensions that describe the solid model. In Figure 3-4 a) the dimensions of each curve are identical in both shapes, but the topology is completely different. Topology is the connectivity and associativity of the solid models entities. Figure 3-4 b) shows that the two figures have the same connectivity at the edges, meaning the topology is the same, but the geometry is different. In order to uniquely and consistently define a solid model, both the topology and geometry of the model need to be parsed and carefully duplicated in the new system.

By following the described method of simple geometry creation, information from a CAD/CAE package can be used to construct solid geometry inside of the MCAD package. The process of using this parsed information to create solid models is the key to automated enclosure design. The MCAD designer no longer needs to interpret data and construct models on his/her own, the computer can perform these operations automatically. The resulting solid model created in the MCAD package will be a representation of that same object defined in the CAD/CAE package.

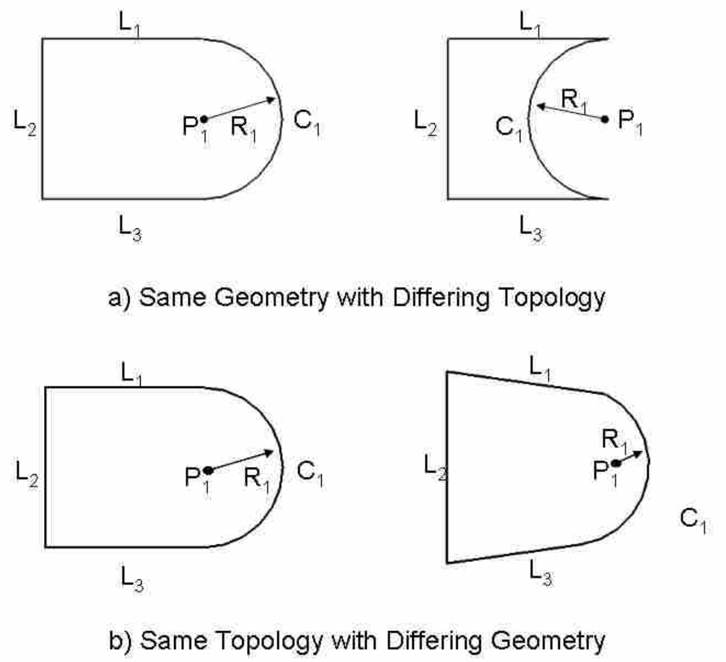


Figure 3-4: Geometry vs. Topology

3.3 Assemblies

In addition to an accurate definition of the CAD/CAE object in MCAD, it is important to know how that object relates to other objects also defined in the CAD/CAE system. Most objects that reside in CAD/CAE systems are part of the overall product assembly. An assembly can be described as a compilation of individual components and their relationships to each other. For this research, an assembly will represent a collection of geometric models (polyhedrons) of individual parts, and their spatial/attachment relationships to each other.

An assembly model can be thought of as a hierarchy of individual parts. Each level of the hierarchy can either be a single geometric model, or a subassembly of geometric models. The hierarchical relationship between parts of an assembly can easily be seen with an assembly tree (Figure 3-5). In this figure, the overall assembly is where all components and subassemblies are placed and related to one another. An S represents a subassembly, and a C represents a single component. The superscript number represents the hierarchal level of the assembly and the subscript represents its number in that level.

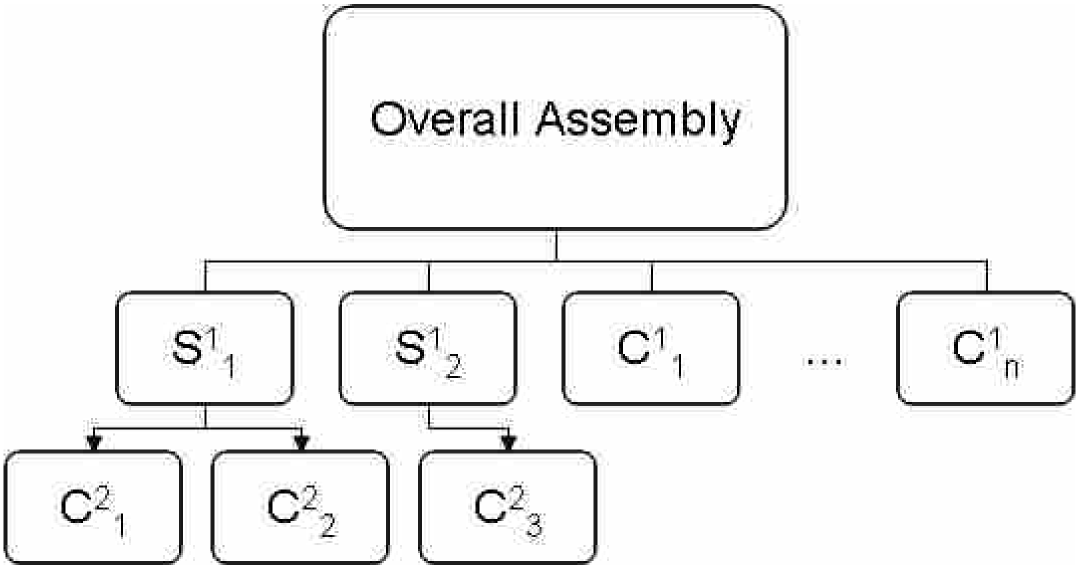


Figure 3-5: Assembly Hierarchy

3.3.1 Establishing Component Relationships within an Assembly

In order to place a model into an assembly, its relationship to other components in the assembly must be defined. To accomplish this, there are six variables that must be defined. These variables are the three rotations (α, β, γ) and the three coordinates (x, y, z) of the origin. By knowing these six variables the model and its relationship with the overall assembly is properly defined. In most instances, the necessary variables are defined in the CAD/CAE and can be gathered during the file parsing process. For example, the widget defined in the file parsing section has an origin location where the widget should be placed in the overall assembly.

There are times when the exact variables are not defined or known. In this case, most MCAD systems provide a way to "mate" components to each other. Mating components refers to a method where constraints or conditions are formed between features of two components in an assembly. Such features can be: datum planes, faces, axes, edges, and vertices. When a mating condition is formed one or more of the six variables become defined. When no mating conditions exist, then the MCAD user must define them through interactive operations. It is good modeling practice to continue creating mating conditions until all six variables are completely defined.

Assembly order ambiguity can cause serious problems on an automated assembly line. Similarly, when designing any product, it is important that all components be placed and related correctly. For this reason it is important to extract the data from the CAD/CAE package, if it exists. This will reduce the probability of human error. If the information does not exist in the CAD/CAE package, the user must define the relationships of the components by way of mating conditions.

3.4 Tolerance Stack-Up

Whenever products are created and/or assembled, it is expected that their parts and their location in the assembly do not match the specifications exactly. Therefore, tolerances are provided to limit the inaccuracy. When creating assemblies, the dimensional variation of all the product's parts add up and accumulate throughout the assembly. This summation of variances is called tolerance stack-up and the prediction of the resulting accumulation is called tolerance analysis. Variation can cause problems with the final product, resulting in rework, scrapped parts, or low performance. There are many different tolerances that could affect an enclosure's design. The list below states a number of different tolerances that could cause variation when creating an enclosure design.

1. Component placement (X, Y, Z) with respect to assembly
2. Component rotation (α, β, γ) with respect to assembly
3. Assembly placement with respect to the enclosure
4. Assembly rotation with respect to the enclosure
5. Component thermal expansion
6. Component electrical expansion (piezoelectric effect)
7. Flex of a product
8. Dimensional Variation of components
9. Dimensional Variation of enclosure

There are a number of methods used to calculate the stack-up of tolerances. Three of these methods are discussed in this section. The three methods discussed are: Worst Case, Root-Sum-Squares, and Uniform Distribution. The information given below is based on a chapter by Dr. Ken Chase (Brigham Young University). [Chase, 2004]

Before a tolerance stack-up analysis can be performed, there are a few standards that should be taken into account. The bulleted list below defines each of the standards needed for a tolerance stack-up analysis.

- Tolerance Domination – A preliminary check of the tolerance values should be made to assure that no particular tolerance value is dominant over the others.

- Central Limit Theorem – For the statistical methods defined in this section, it is important that there be enough tolerance values included in the equation to approach a normal distribution. In most cases five or more values are needed in a tolerance stack.

- Quality Level – For statistical methods, it is important to define the acceptable level of tolerance variance for components and assemblies. The level is also known as the sigma quality level. The target value most widely used today is ±3 sigma.

3.4.1 Worst Case (WC)

The WC method is a method to compute the worst combination of variances within the assembly, meaning that each dimension will be set to its extreme tolerance value. It is not a statistical approach to the tolerance stack-up issue. To perform a WC calculation the extreme tolerance values are simply summed together. Even when dealing with a difference in the sign of nominal dimensions being summed, as in a clearance, the positive tolerances are still summed together to achieve the worst case.

$$\sigma_{total} = \Sigma(T_i) \qquad \textbf{(3-5)}$$

In the equation above the σ_{total} represents the overall tolerance, or worst case tolerance for the assembly. The T_i represents each individual dimension tolerance accounted for in the tolerance analysis.

3.4.2 Root-Sum-of-Squares (RSS)

The RSS method is a more realistic approach for analyzing the tolerance stack-up than the WC method. It is not probable that all tolerances will occur at their extreme values at the same time in an assembly. The RSS method uses statistics to predict the range of critical clearances of features, based on statistical probabilities of the dimensional variances. An estimate of the stack-up by the RSS method takes the square root of the sum of each variation squared.

$$\sigma_{total} = \sqrt{\left(\frac{\Sigma(T_i)}{3}\right)^2} \qquad \textbf{(3-6)}$$

The symbols for this equation are the same as in the WC method. The division by three is based on the assumption of a normal distribution that each tolerance represents 3 standard deviations from the nominal.

3.4.3 Uniform Distribution (UD)

The UD method says that every value residing between the maximum and minimum tolerances limits has an equal probability of occurring. This method is more conservative than the RSS method, but not as extreme as the WC method. It is typically used when the actual tolerances are not yet known or verified. The calculation of the UD method is very similar to that of the RSS method. However, because the distribution is uniform and finite, the specified tolerance limits are equivalent to ±√3 standard deviations, hence the following modified formula is used to estimate the variation stack-up.

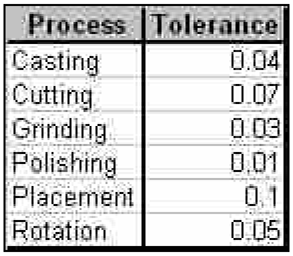
$$\sigma_{total} = \sqrt{\left(\frac{\Sigma(T_i)}{\sqrt{3}}\right)^2} \qquad \textbf{(3-7)}$$

It is interesting to note that the resulting distribution of the sum approaches a normal distribution, as more tolerances are included in the chain. It is assumed in practice, that if 5 or more tolerances are summed, the resultant distribution is approximately normal. This will be true as long as no single tolerance is so large that it dominates the stack.

3.4.4 Comparison of Methods

To compare the three tolerance stack-up methods the widget example is used. The manufacturing process to create the widget involves casting, cutting, grinding, and polishing. The widget is also placed as a component of an assembly and has additional placement and rotation tolerances. The tolerance values were extracted from the CAD model through the use of a GUI and the values of which are shown in Table 3-1. It should be noted that the values placed here are fictional, for the purpose of visualizing the method, and not intended to match real tolerance values.

Table 3-1: Widget Tolerance Values

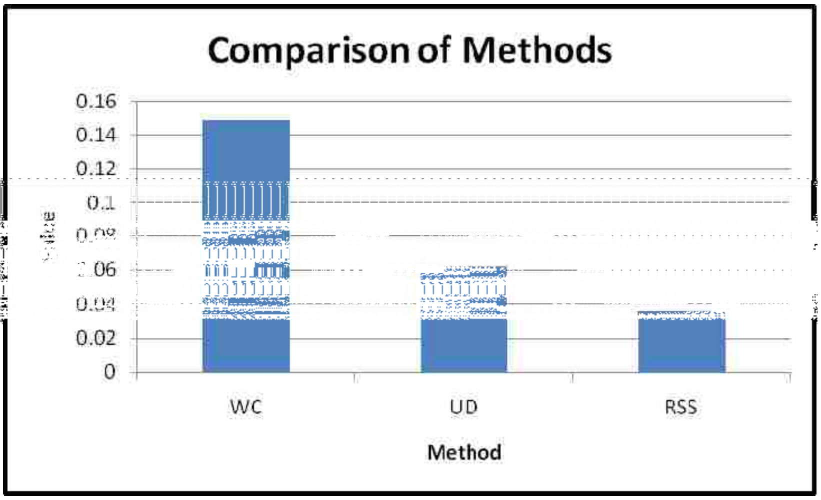


Process	Tolerance
Casting	0.04
Cutting	0.07
Grinding	0.03
Polishing	0.01
Placement	0.1
Rotation	0.05

The table shows the varying tolerance values attributed to their respective manufacturing process of the widget. The placement and rotation of the widget on an assembly also contributes to the overall tolerance. By inserting these values into the equations given above, estimates of overall assembly values were calculated for each stack-up model, and are presented in Table 3-2 and the accompanying bar chart.

Table 3-2: Method Comparison

thod	Value
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]



As can be seen in the table above, the WC method is by far the most conservative estimate of the three methods. The UD method is more conservative than the RSS method, however, not as extreme as the WC method. A combined method, which combines RSS and UD is presented in Appendix B.

Tolerance stack-up analysis is very important to any enclosure design. If tolerances are not taken into account when designing the enclosure, the outcome could be very costly and/or dangerous. For example if a component is placed too closely to the enclosure for an electronic design, arc flashing can occur causing the PCB to fail.

The proposed method permits the effects of tolerance stack-up in an enclosure to be estimated and evaluated. The enclosure dimensions may then be altered as needed to maintain critical clearances in the assembly process. For more information about tolerance analysis see the chapter referenced above. [Chase, 2004]

3.5 Enclosure Creation

The purpose of this thesis, was not only to transfer data from one CAD/CAE system to a MCAD system, but to use that information to create an enclosure. Up to this point in the methods section the product, as defined in the CAD/CAE model, has been created and assembled in the MCAD package. It is now necessary to create an enclosure around the product to protect it. There are various forms of protection that an enclosure needs including: protection from the environment (i.e. humidity, extreme temperatures, etc.), protection from mechanical stress (i.e. impact, vibration, etc.), and/or to protect people from being injured by the product (i.e. sharp edges, voltage shock, etc.).

The method for creating a protective enclosure design for many products basically follows the same steps, which will be described below in detail. These steps are:

1. Obtain the boundary curves of the product
2. Offset the curve to provide clearance, and again to define the wall
3. Create the inner envelope of the enclosure
4. Create the outer wall of the enclosure

In order to visually describe the process, the widget example defined in Section 3.1 will be used as the product being enclosed.

3.5.1 Obtaining the Boundary Curves

The first step of the enclosure creation method is to create a splitting plane. The splitting plane is a plane created inside of the MCAD package. The definition of which is provided by the user. The plane is then used to acquire the 2-D boundary curves of the product. These curves are defined to be the intersection between the plane and the faces of the object. The equation below describes this relationship where C_n is the intersection curve, D_{sp} is the splitting plane, and F_n is the face intersecting the plane. These curves are used as the basis for the 2-D definition of the enclosure's overall shape.

$$C_n = D_{sp} \cap F_n \tag{3-4}$$

Looking at the widget example, a plane is first made to intersect the object at its mid-length (Figure 3-6 (a)). The intersection between the faces of the widget and the plane create curves that represent the outer boundary or shape of the widget (Figure 3-6 (b)).

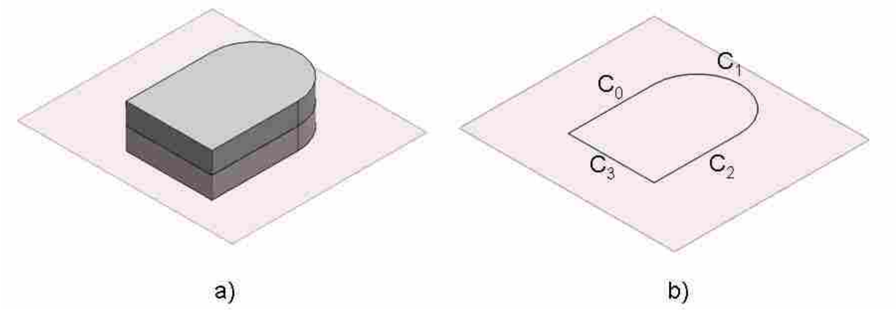


Figure 3-6: Product Boundary Curves

3.5.2 Defining the Enclosure Wall

To create the walls of the enclosure, the boundary curves are offset. An offset curve is defined to be a duplicate curve shifted to the outer side of the enclosed boundary curves that still maintains coincidence with its neighboring offset curves. The distance to shift the duplicate curve is the offset value and is defined by the enclosure designer.

$$O_n = Offset(C_n) \tag{3-4}$$

Two separate offset functions are performed on all boundary curves. The first is a tolerance offset which accounts for the accuracy, or lack thereof, of placing the enclosed product inside of the enclosure and provides the required clearances. The tolerance offset is the inner boundary of the enclosure wall. The second offset is a structural offset. The structural offset defines the outer boundary of the enclosure wall, which gives strength and stability to the enclosure design. The offsets for the widget example is shown in Figure 3-7 below.

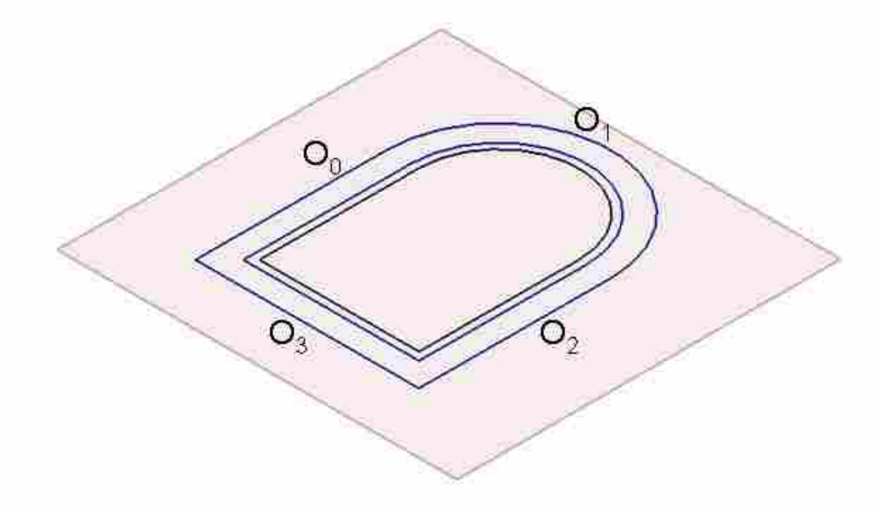


Figure 3-7: Enclosure Offset Curves

Up to this point, the enclosure is only designed on a 2D plane and needs to be defined in 3 dimensions. To create a 3D geometric model, a simple polyhedron is created (see Section 3.2). Following the method of polyhedron creation, the vertices, edges, and faces of the enclosure must be defined. For the enclosure design, two sets of vertices are needed.

3.5.3 Creating the Inner Envelope

The first set of vertices is used to create the inner envelope of the enclosure. The inner envelope is the void inside the enclosure that houses the product. The connecting points of the inner offset clearance curve are the lower vertices for the envelope. Each of these vertices will have a paired higher vertex. The higher vertices are the corner points of the ceiling surface of the inner envelope. The ceiling surface can be created in any number of ways, but must have the same number of vertex points as the lower vertices. This is to ensure Euler's equation remains valid. The next step is to connect the lower and higher vertex pairs to form an edge. Using the edges, the faces can be defined as is described in Section 3.2.

For the widget example the inner envelope is created by gather the lower vertices shown in Figure 3-8(a). The ceiling surface is then defined as the face bounded by offsetting the tolerance curves a distance equal to height of the widget (Figure 3-8 (b)). The vertex pairs are the connected together by a straight line (Figure 3-8(c)), and the faces bounded by the edges are created (Figure 3-8 (d)), thus defining the geometry of the inner envelope.

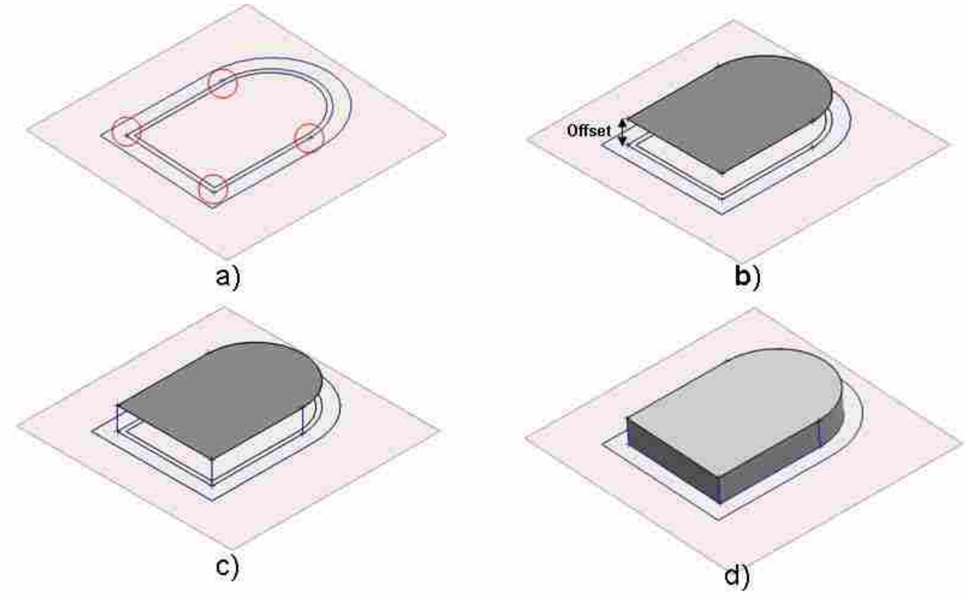


Figure 3-8: Inner Surface Creation

3.5.4 Creating the Outer Wall

The second set of vertices defines the structural or outer walls of the enclosure. Similar to the inner envelope vertices, the connecting points of the structural offset curves are the lower vertices of the set. The lower structural offset vertices also have paired vertices, which are defined to be the corner points of the roof surface of the enclosure. The roof surface can take on any size or shape; however it must contain the same number of vertex points as the lower vertices. The outer wall definition is completed by creating edges between vertices, and faces from edges as has been discussed.

For the widget example, the roof surface is defined as the surface bounded by offsetting the structural curves, as the inner ceiling surface was defined. The offset for the enclosure roof is equal to a distance equal to the maximum height of the product, plus the wall thickness. Just as was described with the inner ceiling surface, each set of vertices, with their paired vertices, are used to create their respective edges. The edges then define the faces of the polyhedron and a solid enclosure body is defined (Figure 3-9) which satisfies Euler's formula.

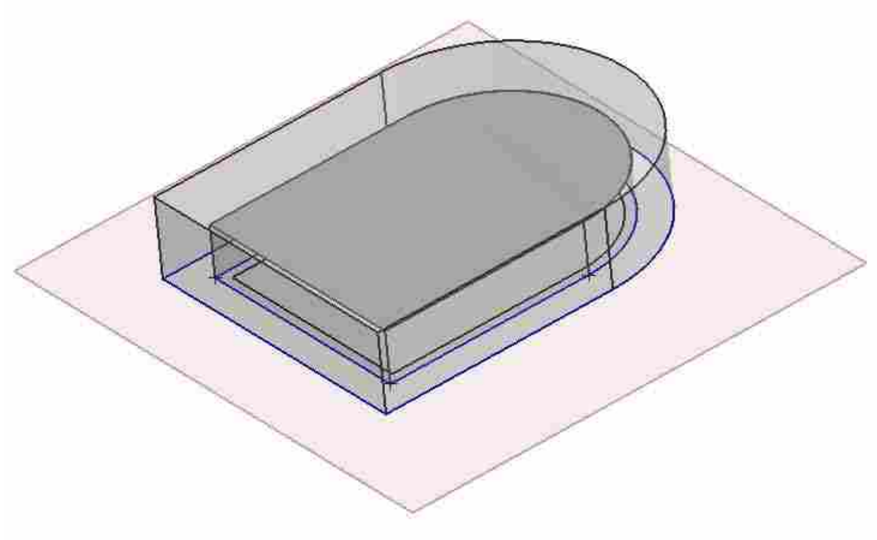


Figure 3-9: Solid Model of Enclosure

As was stated earlier, the widget example was used for simplicity in visualizing the process. However, it is important to note that the edges or faces between the vertices do not have to be linear or flat. Any arc, spline, or curve can be used as an edge, and any b-surface can be used as a face and still satisfy Euler's equation. An example of this will be seen in the implementations of the methods in Chapter 4.

The enclosure serves to protect the product it houses, for this reason the method has been developed to assure creation of a housing that is structurally sound. The method defined can be applied to a wide spectrum of enclosure designs. The main benefit of this method is that it can be fully automated, which increases the efficiency of the design cycle.

3.6 Enclosure Extensibility

The method described in Section 3.4 gives the steps to create the basic enclosure design. A fully designed enclosure could require many other features than just the basic body. For example, heat fins are needed on many PCB enclosures to draw heat away from the PCB itself. Often, access to the product inside the enclosure is required, so holes or mechanical parts, such as doors are added. Each of these added features are extensions to the basic body which was built in Section 3.4.

To create additional geometry, boolean operations are used to achieve the desired results. There are three boolean operations that exist. These are: union ($\cup$), intersection ($\cap$), and difference (-). The union operation is used to combine or add together two polyhedrons. The intersection operation results in a shape equal to the common volume of the two polyhedrons. And the difference operation is used to subtract one object from the other. Figure 3-10 shows the resulting solid model of the boolean operation used.

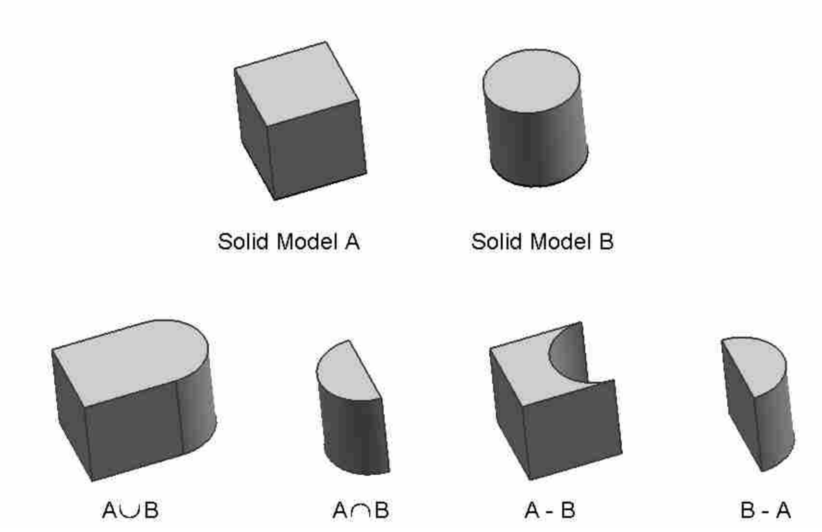


Figure 3-10: Boolean Operations

Once a Boolean operation has been performed, it is important to note that Euler's equation is still valid. The equation itself has changed due to the fact that the number of vertices, edges, and faces has changed, but the equation still holds true.

To illustrate the enclosure method extensibility, the widget model is once again used. The widget has been defined as an object acting as a heat sink. The enclosure designed in the previous section needs to be modified to allow the heat to escape. Therefore, it is necessary to perform a subtraction Boolean operation to create a hole for venting. To accomplish this, a simple cylinder solid model is created to pierce through the enclosure model (Figure 3-11 (a)). The cylinder solid body is then subtracted from the enclosure solid body creating a new feature on the enclosure (Figure 3-11 (b)).

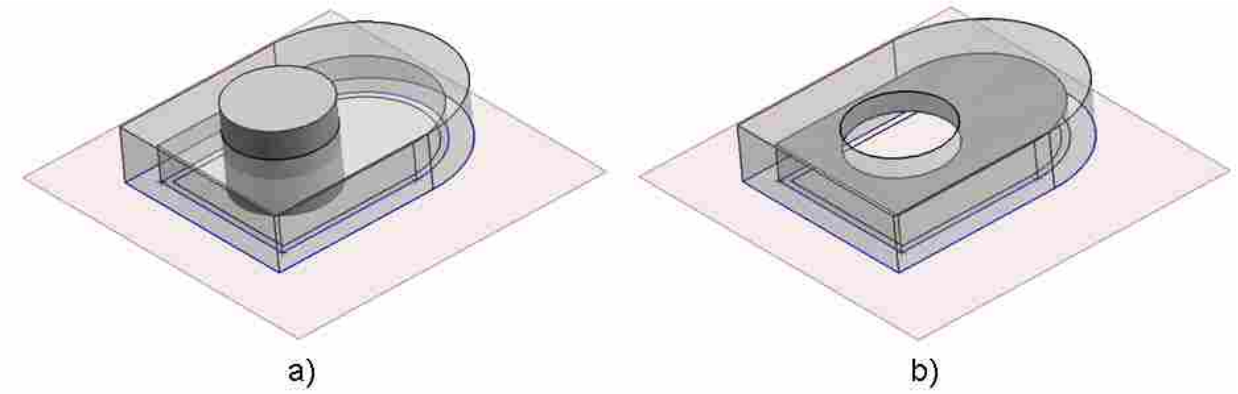


Figure 3-11: Venting Hole Creation

The option to add extensible features to the enclosure design makes the overall method more applicable to all enclosure designs. The method of creating extensible features can also be implemented into an automated process. The Boolean operations defined in this section can add, intersect, or subtract any feature needed to complete the design of the enclosure model.

3.7 Conclusions of Methods

The methods described in this chapter have demonstrated the potential of the thesis work. The file parsing and geometry creation sections demonstrated how to fully capture the information from any CAD/CAE package and use that information to create solid model representations in MCAD. The models are then placed and related together to create an overall assembly of the product. Before designing the actual enclosure, careful attention must be given to the many tolerances and required clearances associated with assembly design.

Once the product has been fully assembled, and product tolerances defined, the method of creating a basic enclosure design has been described in such a manner as to be applicable to any enclosure design. A method was also described to allow additional features that are needed when creating detailed enclosure designs.

The significance of this method is that it can be fully automated within the MCAD package. By automating the methods described in this chapter, the probability for human error is decreased, data is not lost in translation, and the time needed to create enclosure designs is drastically reduced. The automation of the methods is explained in Chapter 4, and the results of which are discussed in Chapter 5.

4 Implementation

This chapter contains the implementations of the methods used to create a simple enclosure around a PCB. The objective of this chapter is to demonstrate that following such steps, as described in Chapter 3, will automatically create a preliminary enclosure design. The implementation illustrates how the process accurately translates electronic and thermal information from an ECAD package and uses that data to automatically create an enclosure to house a PCB in a MCAD package. Although this process has been developed for specific ECAD and MCAD packages, it will be presented as generally as possible in order to apply it to other CAD/CAE packages. The process was developed using C-style coding and the language of the thesis will reflect this, however the content of the process may be applied to any other code developing environments.

4.1 Implementation of Methods for Automated Enclosure Design

The purpose of this thesis is to demonstrate that electrical and thermal rules can govern the design of the enclosure surrounding a PCB. The previous chapter discussed a generalized method to create an enclosure around any product. This chapter explains step-by-step the specific method used create an enclosure that protects a PCB. This thesis is a proof of concept research project and is not intended to create a fully designed commercial product.

The end user interfaces with the developed code by a number of buttons on a toolbar as shown in Figure 4-1. Each button performs a specific operation as explained in detail below. By following this process, an entire enclosure around the PCB board is created.

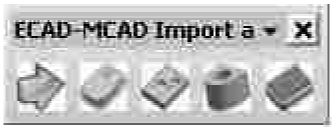


Figure 4-1: Enclosure Generation Toolbar

4.2 Button 1: ECAD Import and PCB Assembling

There are three main objectives involved with button 1. The first objective is the parsing of the ECAD file to gather the information necessary to create a PCB inside of MCAD. The second objective is to create the geometry of each building block of the PCB. The last objective is to assemble the building blocks together, forming a representation of the PCB. The end product of Button 1 is a fully assembled PCB.

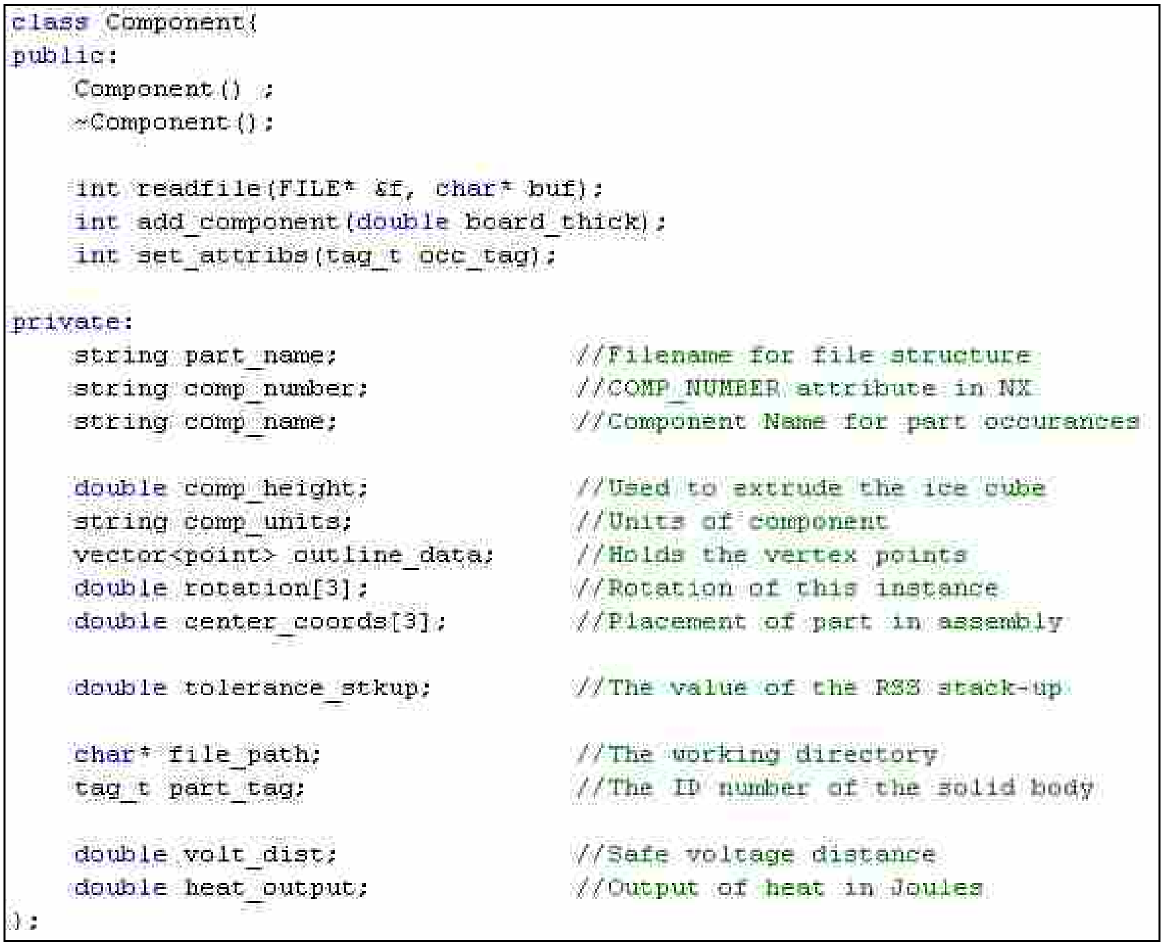
In order to make a complete PCB, three building blocks that are needed, which are: components, traces, and the wiring board. A component could be any number of different parts that exist on a PCB. It could be a resistor, inductor, capacitor, integrated circuit, etc. The components on the PCB are what define the operation of the PCB. A trace is a single strip of conductive material that electrically connects components on the wiring board. The wiring board (WB) is used to hold electronic components and electrically connect them by means of traces. For the rest of this thesis a component will be denoted as a C, a trace as a T, and the wiring board as WB. Therefore, a PCB is a compilation of all three of these building blocks.

$$PCB = \Pi(WB, \Sigma(C), \Sigma(T)) \tag{4-1}$$

4.2.1 Gathering Information

The first objective of Button 1 is to gather the necessary data from both the ECAD package and the end user. The most efficient manner to gather information is by following the method described in Section 3.1 (File Parsing). The file parsing method is very similar for all three types of building blocks and a discussion on the procedure for each would be excessive. Therefore, only the file parsing for the component building block will be described in this section. In order to define a component in the code, a component class was created. The class is shown in Figure 4-2. The component class holds the functions and data necessary to create a component in the MCAD system.

There are three functions defined in the component class. The *readfile* function is called when the code is ready to parse the ECAD file for the component data. The *add_component* function is called when the PCB assembly is being created and is used to place the component into the assembly. The *set_attribs* function is also called when the PCB is being created. This function takes certain variable information and links this data to the solid component model. These three functions will be described in more detail in their respective sections.



```
class Component{
public:
    Component() ;
    ~Component();

    int readfile(FILE* &f, char* buf);
    int add_component(double board_thick);
    int set_attribs(tag_t occ_tag);

private:
    string part_name;               //Filename for file structure
    string comp_number;             //COMP_NUMBER attribute in NX
    string comp_name;               //Component Name for part occurances

    double comp_height;             //Used to extrude the ice cube
    string comp_units;              //Units of component
    vector<point> outline_data;     //Holds the vertex points
    double rotation[3];             //Rotation of this instance
    double center_coords[3];        //Placement of part in assembly

    double tolerance_stkup;         //The value of the RSS stack-up

    char* file_path;                //The working directory
    tag_t part_tag;                 //The ID number of the solid body

    double volt_dist;               //Safe voltage distance
    double heat_output;             //Output of heat in Joules
};
```

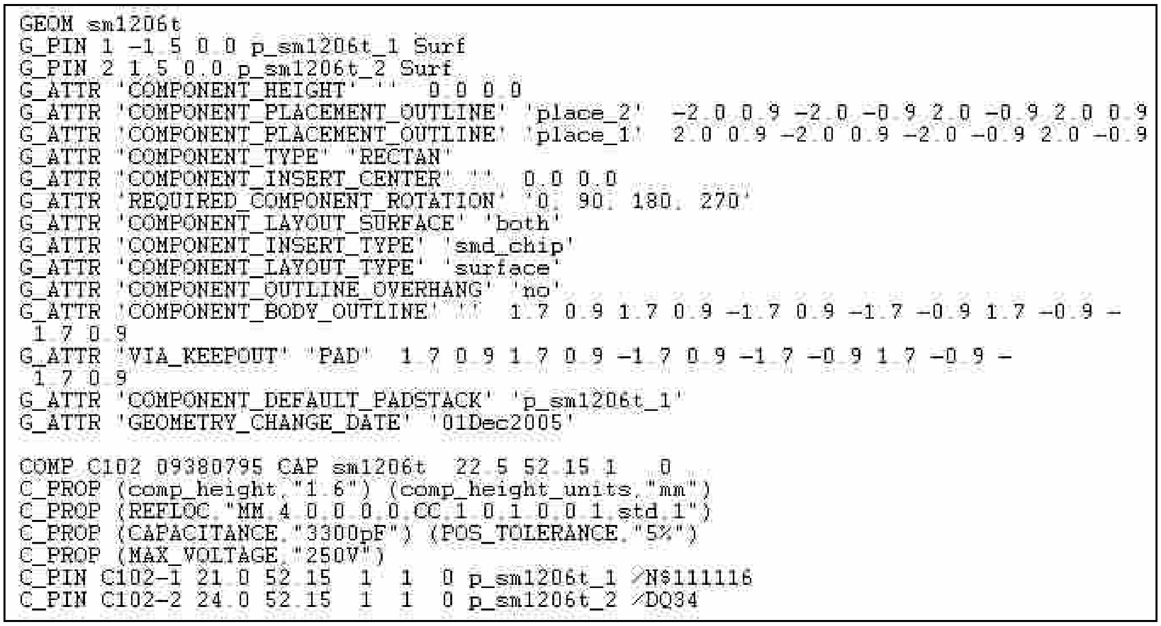
Figure 4-2: Component Class

Specify the Needed Information:

The first step of the defined method is to specify what information is needed. In order to create a component in the MCAD system, component names, a location and rotation on the PCB assembly and the geometry and topology information that define its shape are needed. It is also important to know how much clearance with the enclosure is required so arc flashing does not occur. This information will be useful when creating the enclosure design. Thermal knowledge is also important when designing the heat sink feature of the enclosure. The needed information becomes the variables that are described in the component class.

Understand the Format:

The next step defined in the file parsing method is to understand the file format of the ECAD system. Most ECAD systems output ASCII (American Standard Code for Information Interchange) files. An ASCII file format contains characters that are based on the English alphabet. The ECAD component data is defined in words that can easily be deciphered. An example of the file is shown in Figure 4-3.



```
GEOM sm1206t
G_PIN 1 -1.5 0.0 p_sm1206t_1 Surf
G_PIN 2 1.5 0.0 p_sm1206t_2 Surf
G_ATTR 'COMPONENT_HEIGHT' ''  0.0 0.0
G_ATTR 'COMPONENT_PLACEMENT_OUTLINE' 'place_2'  -2.0 0.9 -2.0 -0.9 2.0 -0.9 2.0 0.9
G_ATTR 'COMPONENT_PLACEMENT_OUTLINE' 'place_1'  2.0 0.9 -2.0 0.9 -2.0 -0.9 2.0 -0.9
G_ATTR 'COMPONENT_TYPE" 'RECTAN"
G_ATTR 'COMPONENT_INSERT_CENTER' ''  0.0 0.0
G_ATTR 'REQUIRED_COMPONENT_ROTATION' '0, 90, 180, 270'
G_ATTR 'COMPONENT_LAYOUT_SURFACE' 'both'
G_ATTR 'COMPONENT_INSERT_TYPE' 'smd_chip'
G_ATTR 'COMPONENT_LAYOUT_TYPE' 'surface'
G_ATTR 'COMPONENT_OUTLINE_OVERHANG' 'no'
G_ATTR 'COMPONENT_BODY_OUTLINE' ''  1.7 0.9 1.7 0.9 -1.7 0.9 -1.7 -0.9 1.7 -0.9 -
 1.7 0.9
G_ATTR 'VIA_KEEPOUT" "PAD"  1.7 0.9 1.7 0.9 -1.7 0.9 -1.7 -0.9 1.7 -0.9 -
 1.7 0.9
G_ATTR 'COMPONENT_DEFAULT_PADSTACK' 'p_sm1206t_1'
G_ATTR 'GEOMETRY_CHANGE_DATE' '01Dec2005'

COMP C102 09380795 CAP sm1206t  22.5 52.15 1   0
C_PROP (comp_height,"1.6") (comp_height_units,"mm")
C_PROP (REFLOC,"MM,4,0.0,0.0,CC,1,0,1,0,0,1,std,1")
C_PROP (CAPACITANCE,"3300pF") (POS_TOLERANCE,"5%")
C_PROP (MAX_VOLTAGE,"250V")
C_PIN C102-1 21.0 52.15  1  1  0 p_sm1206t_1 /N$111116
C_PIN C102-2 24.0 52.15  1  1  0 p_sm1206t_2 /DQ34
```

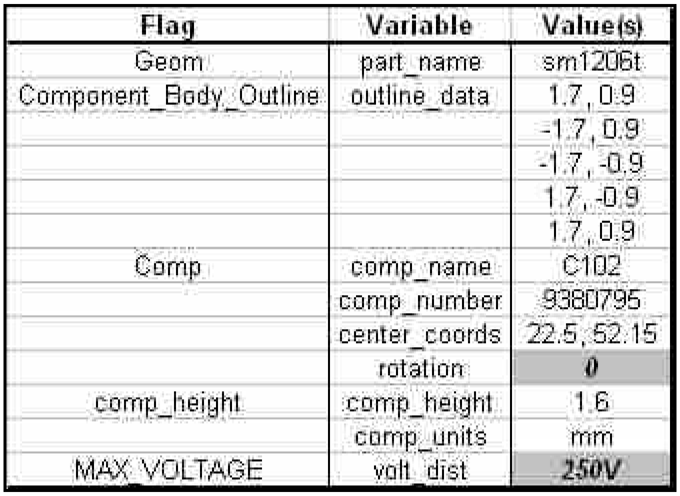
Figure 4-3: Component ASCII File

The file at first can be difficult to read, but over time becomes understandable. In this example, any line that begins with a G contains information about the geometry and topology of the component. A line that begins with a C contains information that describes the individual component attributes. After studying the file, the information that we need can be seen. For example, sm1206t is the part name of the component. The line containing the words 'COMPONENT_BODY_OUTLINE' has a list of numbers. These numbers represent the X and Y values of each vertex of the footprint. Looking further down the file, the component height is given as "1.6" with units of "mm". The maximum voltage and thermal output are also given, along with other data that is needed for the component class.

Establish Flags:

Most of the information that is given in the example will not be used in the MCAD package so it is important to tell the parser where the important information is. Referring back to the component class the first variable needed is the part name. As was described above, the part name is the name that comes directly after the word **GEOM**. **GEOM** will be the first flag established. The comp_name and comp_number variables are found to be first and second names that follow the word **COMP** in the file. Therefore, **COMP** becomes the second flag. The first two numbers on this same line are the X and Y coordinates of the center_coords variable. The number at the end of the line tells if the component is to be placed on the top of the board (0) or on the bottom of the board (1). The **COMP** flag is the flag for all four variables. This process continues until all the needed data is found. The table below shows what variables are mapped by which flags.

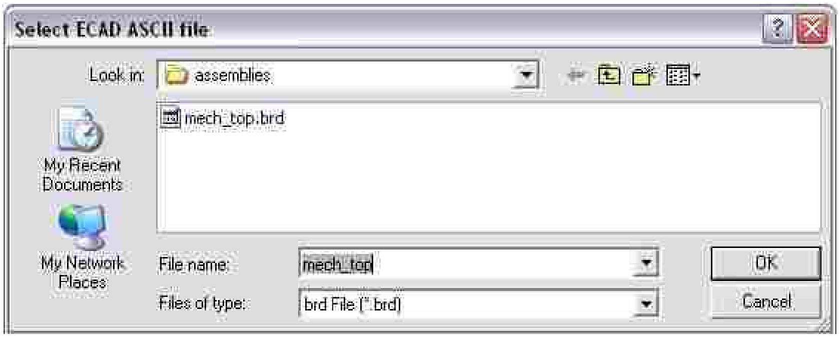
Table 4-1: Variable to Flag Table



Flag	Variable	Value(s)
Geom	part_name	sm1206t
Component_Body_Outline	outline_data	1.7, 0.9
		-1.7, 0.9
		-1.7, -0.9
		1.7, -0.9
		1.7, 0.9
Comp	comp_name	C102
	comp_number	9380795
	center_coords	22.5, 52.15
	rotation	***0***
comp_height	comp_height	1.6
	comp_units	mm
MAX VOLTAGE	volt_dist	***250V***

Parse File Line-by-Line:

After the flags are established and the variables are assigned to their flag, the parsing code can start to read the file. At this point, the *readfile* function is called and the user is prompted through a GUI to select the ASCII file that contains the component data (Figure 4-4). The parser reads each line of the selected file and compares the characters to each flag. Once the characters read in match the flag characters, the parser not only knows that the data to follow is important, but it also knows which variable to assign the data to. The parser continues to read in characters line-by-line until it reaches the end of the file.



Select ECAD ASCII file

Look in: assemblies

mech_top.brd

File name: mech_top

Files of type: brd File (*.brd)

Figure 4-4: ASCII File GUI

Map the Data:

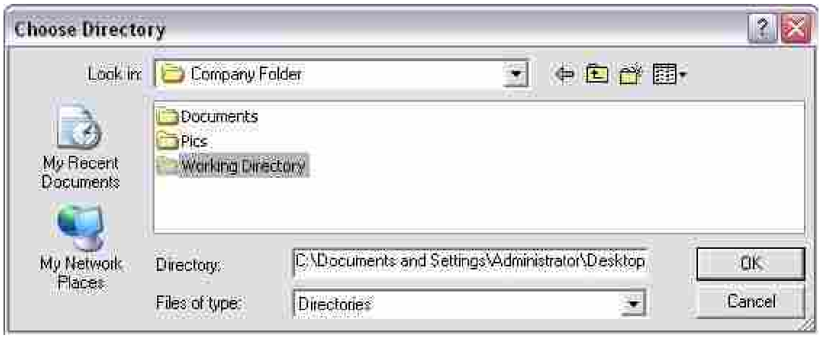
Once the parser hits a flag and reads the following data, it passes the information to the code. The code then takes the information it is given and decides if it can be directly mapped, or if it needs to be representatively mapped. In Table 4-1 the data format for the variable values that can be directly mapped are shown in white cells. This means that the MCAD data type is equal to the ECAD data type. The shaded cells in the table contain data that does not match the MCAD data type. It should also be noted that there are three variables in the component class that were not found in the ECAD ASCII file, such as the file path, and the tolerance stack-up. These variables must be acquired through the use of a GUI.

For direct mapping, the variable is set equal to the data. For example: part_name=sm1206t, comp_name=C102, and comp_height=1.6. The outline_data vector is slightly different. Each of the paired X and Y values are pushed back onto the vector, until there are no more values to be read.

The representative mapping must be handled carefully. In some instances it is a simple interpretation of a value. For example, the rotation value is read in as a 0. This is interpreted to mean it is placed on the top of the wiring board in the assembly. Because it is placed on the top, the rotation vectors for the X, Y, and Z directions are the same as the wiring board's rotation values. If the value had been read in as a zero the component is placed on the bottom of the wiring board and its rotation values are –X, -Y, and –Z of the wiring boards.

In other instances, the mapping is a mathematical equation, as will be the case with the thermal data discussed in Section 4.5. And in other instances, the value is used in a look-up table to acquire the mapped data, as is the case for the voltage value. From the look-up table a value of 250V is mapped to be a voltage distance of 3.5 mm.

The last method of mapping data is through the use of GUIs. The first GUI that was created, prompts the user to choose a working directory for the PCB (Figure 4-5). When the working directory is chosen, it is then mapped to the file_path variable.


Look in: Company Folder

Directory: C:\Documents and Settings\Administrator\Desktop

Files of type: Directories

Figure 4-5: Working Directory GUI

Similar to the previous GUI, the tolerance GUI prompts the user to enter values for each tolerance. The tolerances are calculated together using the RSS method and the value returned is mapped to the tolerance_stkup variable.

At this point in the implementation, all the needed data has been gathered for the component building blocks. The same method is used to gather the needed data for the wiring board and traces and will not be discussed further.

4.2.2 Creating Building Block Geometry

The second objective of Button 1 is to create a solid model of the wiring board, including all components, and all traces. The solid model of each of these building blocks is a polyhedron and satisfies Euler's equation, as was described earlier. The same method for simple polyhedron geometry creation is used to create these building blocks.

The creation of the wiring board and components is exactly the same and will be described together. The first step of simple polyhedron geometry creation is to know each vertex of the solid model. The file parsing process, described above, gathered the footprint vertex values and stored them in a vector called outline_data. The values are used to create enclosed curves that define the 2D topology and geometry of the building block. To create the curves, each vertex is read from the point vector. Each vertex has an X and Y value. The vertex vector is defined such that the first vertex in the vector, and the last vertex in the vector are identical, which causes the curves to form a closed loop (Figure 4-6 (a)). To create the curves, a line is created between two points. This process continues until it reaches the end of the vertex vector.

$$C_n = (X_n, Y_n) \,\&\, (X_{n+1}, Y_{n+1}) \tag{4-2}$$

These curves are all coincident and make up the 2-D shape of the building block. This process is shown in Figure 4-6 (b).

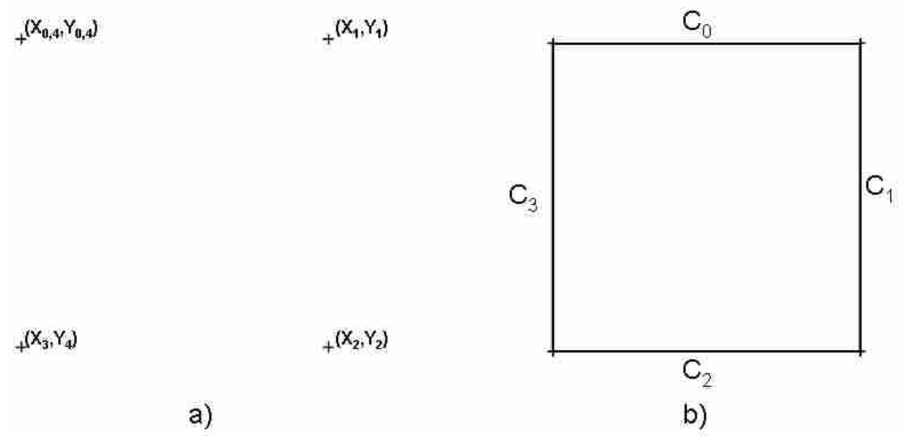


Figure 4-6: 2D Shape of Component

The resulting closed loop only contains the lower vertex points of the polyhedron. In order to create a solid model the upper vertex points need to be defined. All MCAD packages have a tool that creates a 3D solid model from a 2D sketch. For this thesis the operation will be called an extrusion. The extrusion function receives the base sketch geometry and extrusion length as inputs. The base sketch for the extrusion is the enclosed curves just created, and the extrusion length is the height of the building block that was gathered during the file parsing process (comp_height). Once the extrusion is created, a solid model is formed (Figure 4-7).

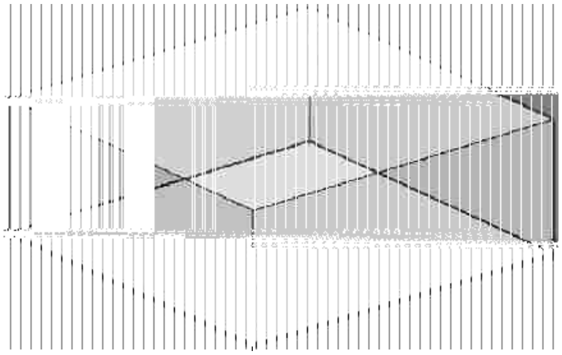


Figure 4-7: 2½-D Representation of Building Block

The solid model is only a 2½-D representation or "ice cube" of the actual building block. Figure 4-8 shows ice cube representations of two components. The detailed 3D part is shown residing inside of the ice cube. It would be ideal to have the ability to extract the 3D detailed solid model geometry from the ECAD package. Unfortunately, only the footprint of the component is given, but is sufficient for the proof of concept nature of this research.

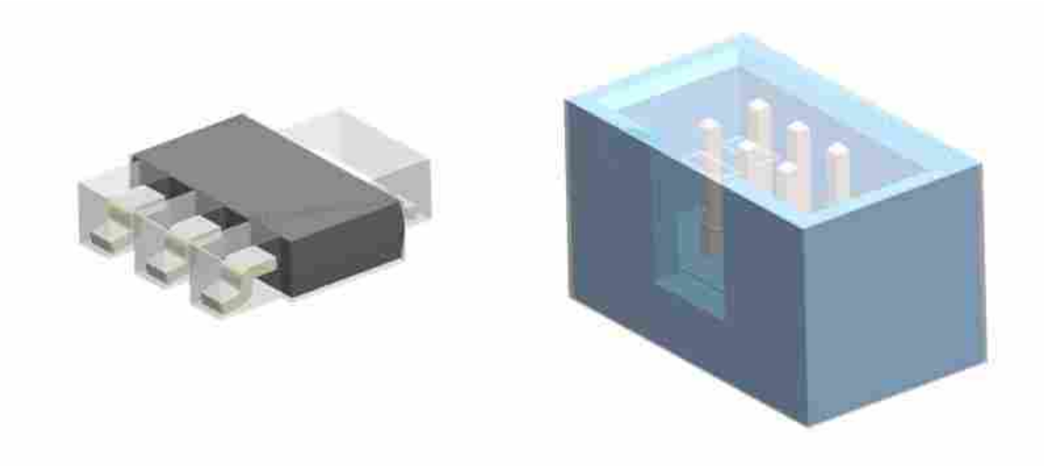


Figure 4-8: Ice Cube Representations

Trace geometry creation is slightly different from the wiring board/component method. This is due to the fact that the ECAD system defines each leg of the trace as a starting and ending point. During the file parsing process, the start and end points of each leg were read in and put into a vector of vertices. Similar to the WB/C, method a line is created between each set of points. Unlike the wiring board and components, the point vector does not create an enclosed curve definition, as seen in Figure 4-9 (a). Therefore, the vertex points of the base shape of the trace must be determined by a different method. To find the vertex points, the width of the trace is used. Figure 4-9 (b) shows that the vertex points are at a distance half the trace width in a direction normal to each curve. After the base vertex points are defined, the base sketch can be extruded to create a solid model (Figure 4-9 (c)).

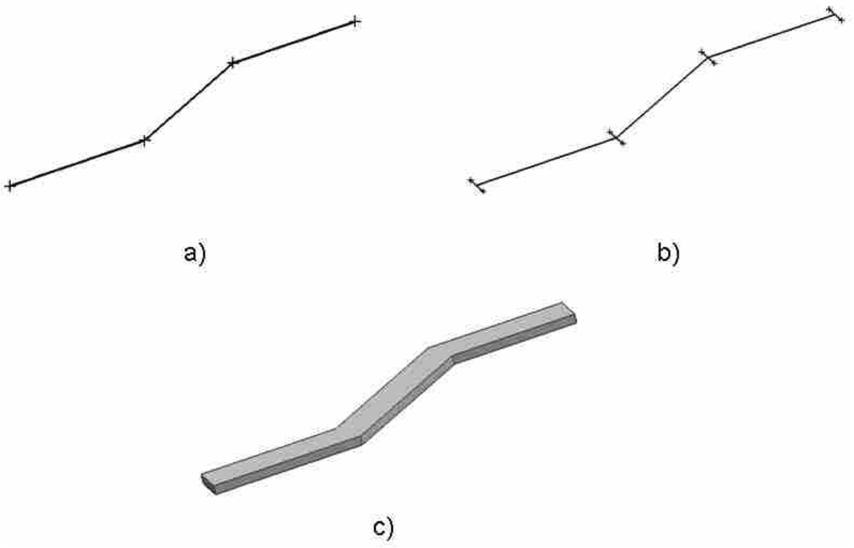


Figure 4-9: Solid Trace Creation

After the geometry/topology of the building blocks has been created, the part files are named and saved to the working directory. The code was developed to create the wiring board and all traces in the same part file. This is done to simplify the assembly. It is not logical or efficient to have each trace be a separate part file. The wiring board/trace part file inherits the name of the part_name variable that was acquired during file parsing.

Each component is saved as separate part file and it too, inherits the name of the part_name variable extracted from the ECAD file. The part file is stored in the working directory, where it can be easily found during PCB assembly.

4.2.3 Assembling the PCB and Assigning Attributes

The overall process for preparing each building block for assembly is shown in Figure 4-10. After that all the building blocks are created in the MCAD package they can be assembled to form a representation of the PCB.

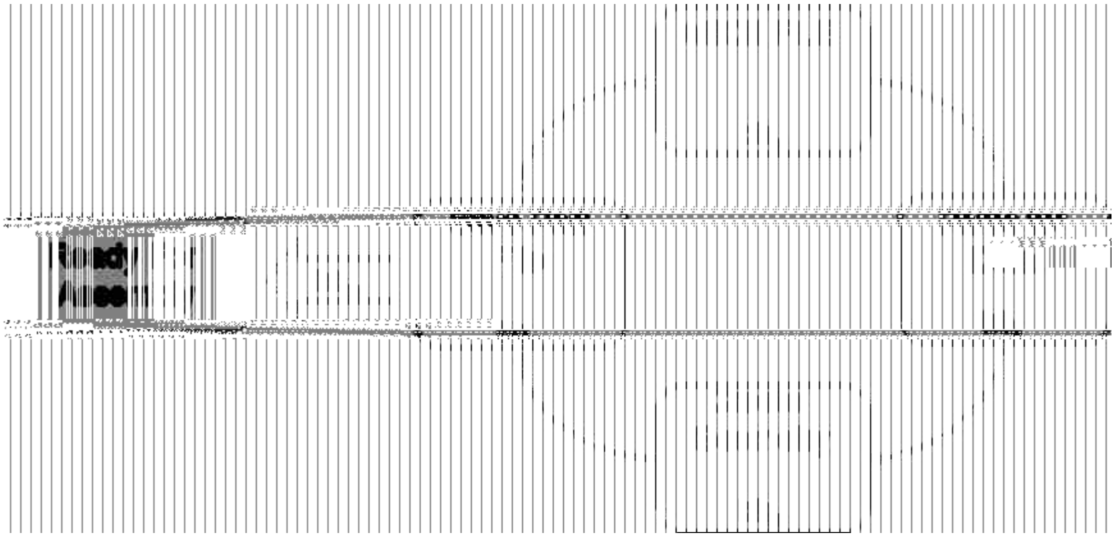


Figure 4-10: Translation Process

At this point in the process the add_component function is called to place each building block into the assembly. The assembly method described in Chapter 3 states that 6 variables are needed to properly place and relate a component into an assembly. These variables are the three rotations (α, β, γ) and the three coordinates (x, y, z) of the origin. The six variables were all defined during the file parsing process as was described above. Each building block is placed in the assembly at a location defined by its center_coords variable and rotated according to its rotation variable.

As each building block is being placed into the assembly the set_attribs function is called. This function assigns a number of attributes to the part file. The assigned information is important for the enclosure automation process. Assigning an attribute to a part file means that the part file not only stores geometry information, but attribute information as well. The attributes that are assigned to each part file are:

1. Component name
2. Part name
3. Component number
4. Voltage distance
5. Thermal output
6. Tolerance Stack-Up

For the fully assembled PCB, and each part file contains vital information that will be important for designing the enclosure. An example of a final PCB assembly is shown in Figure 4-11 below.

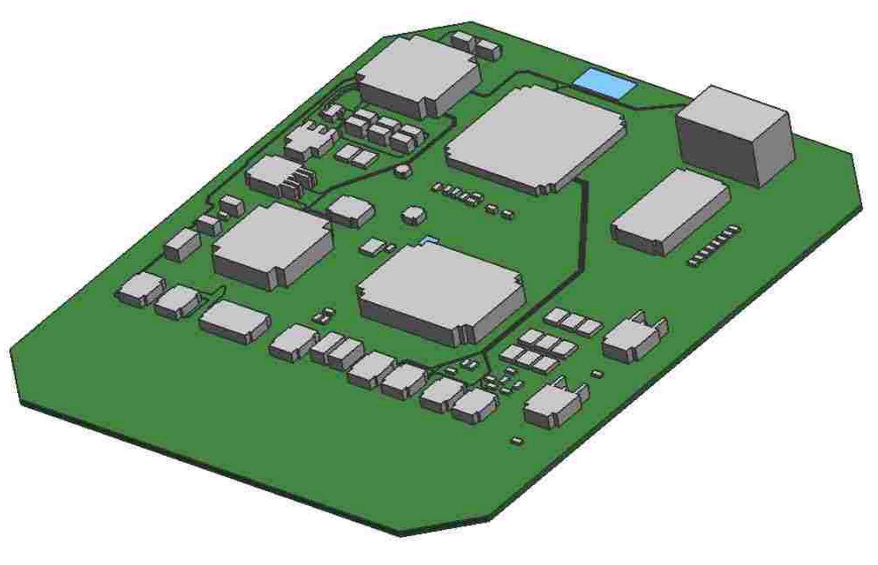


Figure 4-11: Final PCB Assembly

4.3 Button 2: Clearance Zone Creation

In order to create an optimal and safe design, electrical rules are used to shape the inner surface of the enclosure housing. Electrical circuits can be very dangerous if not designed correctly. Each building block on the PCB has various voltage potentials, and if placed too close together, the voltage can jump through the air causing a flash arc. Much of the time spent in designing electro-mechanical devices is spent on analyzing the distance between building blocks, the housing, and other electro-mechanical parts. To diminish the time spent analyzing the distance between voltages; the method developed uses the distance information from the ECAD package to aid in the design of the enclosure and not merely to analyze it.

To ensure that a voltage distance is not violated, a clearance zone is created around each building block. A clearance zone is an area surrounding the solid body of the building block where no other solid body can reside. If any other solid body intersects with the clearance zone, flash arcing is possible.

The difficulty in creating a clearance zone is knowing how large of an area is needed to surround a building block, and knowing how the geometry of the zone should be defined. The solution to each of these problems is discussed below in detail.

4.3.1 Determining the Clearance Zone Size

The size of the clearance zone is defined by how far away the building block needs to be from other objects to reduce the risk that flash arcing will not occur. The distance needed to avoid flash arcing, is equal to the summation of the voltage distance (V_{dist}) and the tolerance stack-up (Tol) of each building block.

$$CZ_{offset} = V_{dist} + Tol \tag{4-3}$$

Here, CZ_{offset} is the safe distance to prevent arc flashing. Both the voltage distance and tolerance values were specified during the file parsing process. The user determines which of the three methods is desired depending upon how conservative the enclosure design must be. These values were stored as attributes in the part file and can be used for the CZ_{offset} equation. The method for solving the tolerance stack-up value was defined in the methods section of the thesis. The method stated that there are three ways to calculate the tolerance stack-up value; Worst Case, Root Sum Squares, or Uniformly Distributed. During the file parsing process, the user selects the desired process with the tolerance GUI.

4.3.2 Determining the Clearance Zone Shape

The shape of the clearance zone is the geometry that is formed by creating a surface that is always the CZ_{offset} distance away from the building block. To better explain the method of creating a clearance zone a 2D sample of the process is explained, and the same methodology is then applied to a 3D object.

Figure 4-12 (a) shows a 2D object which represents a PCB building block. The clearance zone could be created by offsetting each edge of the part, while assuring coincidence at the corners (Figure 4-12 (b)). However, this is not accurate to the definition of a clearance zone. The distance from the corner of the object to the corner of the offset curve is greater than the CZ_{offset} (Figure 4-12 (c)). To correct this, a circle can be placed on each vertex with a radius length equal to the CZ_{offset}. The circle is always tangent to the offset curves. To create a true clearance zone the offset curve, and circle are trimmed at the tangency points as shown in Figure 4-12 (d).

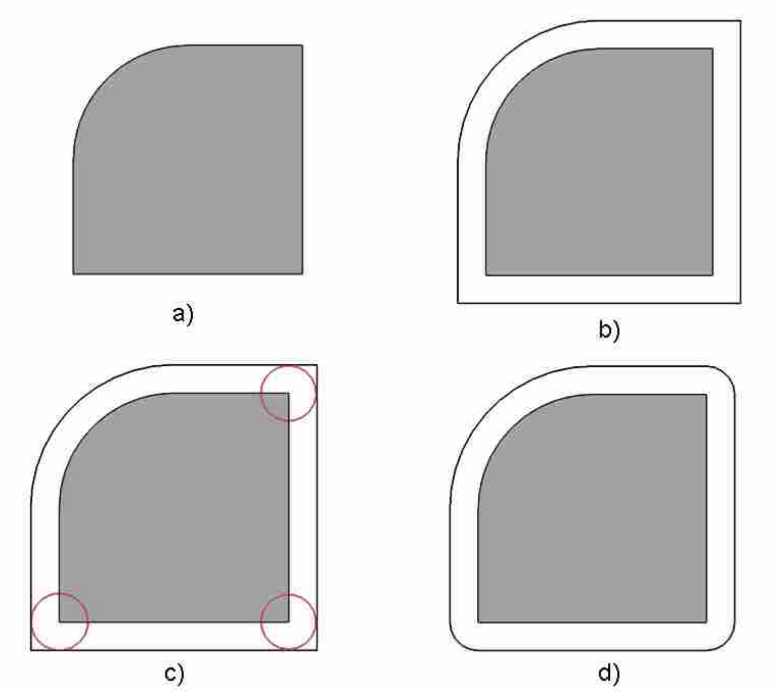


Figure 4-12: 2D Clearance Zone Creation

To apply this process on a 3D object, the circles at the corners become spheres and are placed on every vertex of the object (Figure 4-13 (a)). The clearance zone is created by forming a surface that is offset a distance of CZ_{offset} from the object's faces and is then trimmed by the tangency points of the spheres (Figure 4-13 (b)).

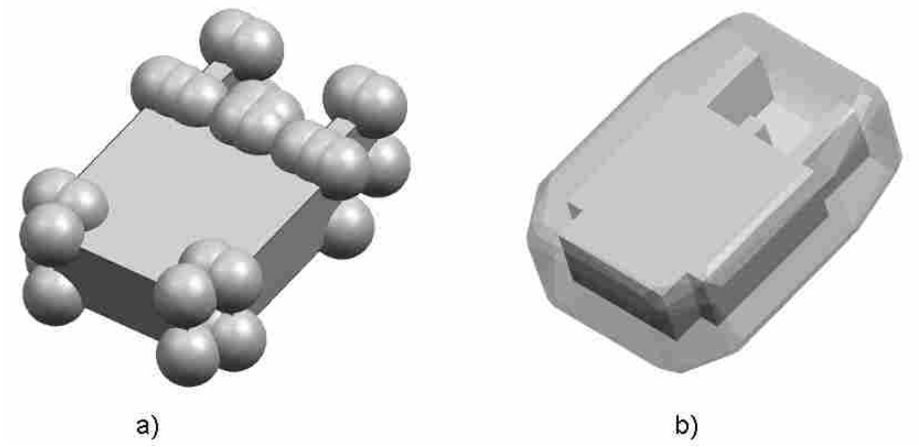


Figure 4-13: 3D Clearance Zone Creation

4.4 Button 3: Generating the Main Body of the Enclosure Design

There are three main processes involved with Button 3. The first objective is to define the 2D geometry of the enclosure wall. The second objective is to create the inner envelope of the enclosure. The inner envelope is the void inside of the enclosure that houses the PCB. The third objective is to create the outer shell of the enclosure. The outer shell is the structural wall that protects the PCB. These three processes create a solid polyhedron and follow the outline discussed in the methods chapter.

4.4.1 2D Wall Geometry

To begin the process of creating an enclosure the overall shape of the solid body must be defined. The first step in the enclosure creation method is to choose a splitting plane to extract the enclosed boundary curves. The splitting plane is the XY plane of the wiring board/trace part file (Figure 4-14 (a)). The boundary curves are extracted from the 2-D footprint of the wiring board and are shown in Figure 4-14 (b). These curves are used as the basis for the 2D shape of the enclosure base body.

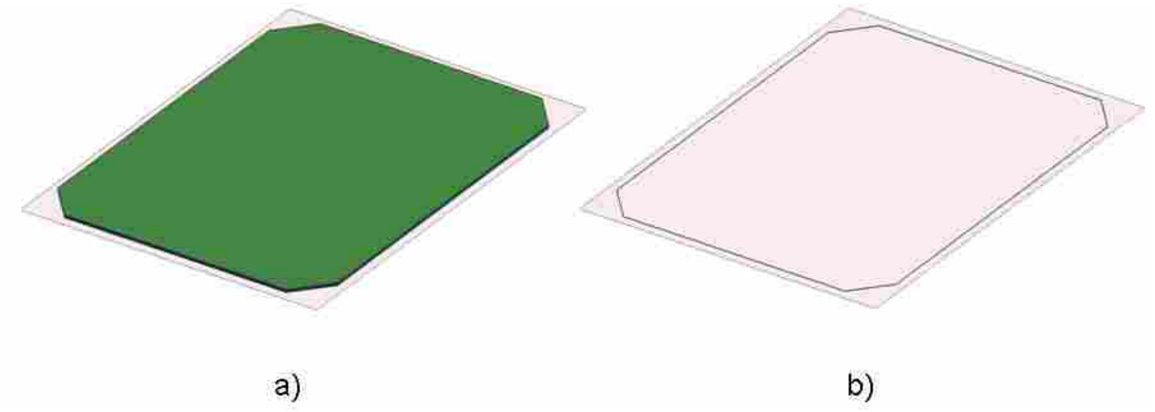


Figure 4-14: Wiring Board Boundary Curves

The next step in the described method is to create offset curves that define the walls of the enclosure. As the method stated above, the first offset performed is a tolerance offset to create the inner wall curves. The tolerance curves are offset a distance equal to the position tolerance of the wiring board.

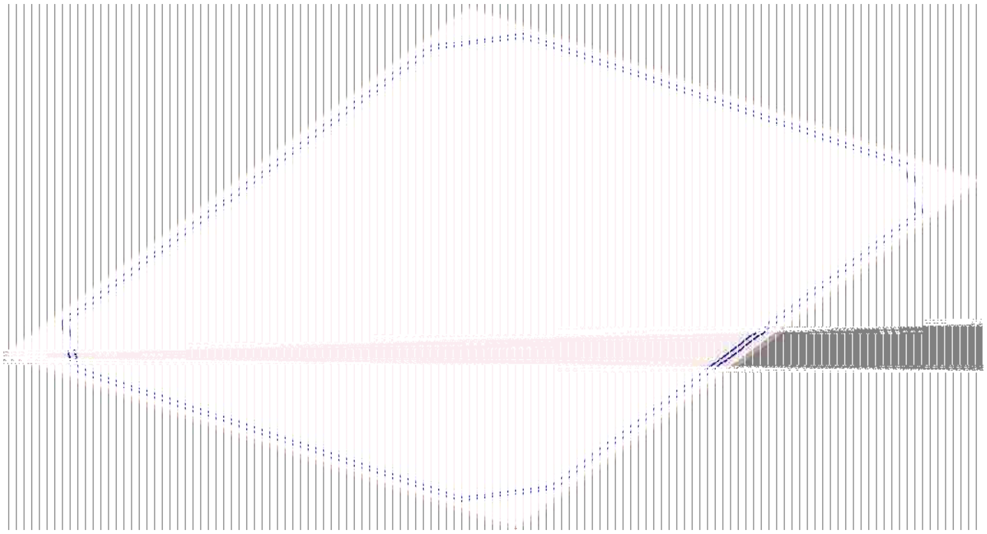
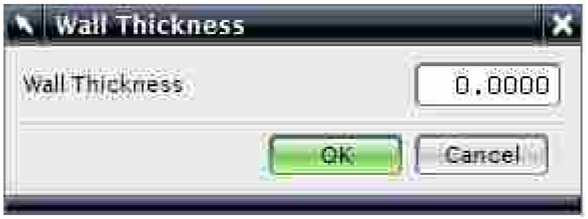


Figure 4-15: Tolerance Offset Curves

The second offset curve is a structural offset and defines the outer wall limit of the enclosure. At this point in the process, a wall thickness is needed and a wall thickness GUI prompts the user to supply a value (Figure 4-16).


Wall Thickness

Wall Thickness 0.0000

Figure 4-16: Wall Thickness GUI

The structural offset curves are offset from the tolerance offset curves by a distance equal to the wall thickness. This creates the outer edges of the enclosure wall and the 2D geometry for the enclosure wall is fully defined (Figure 4-17).

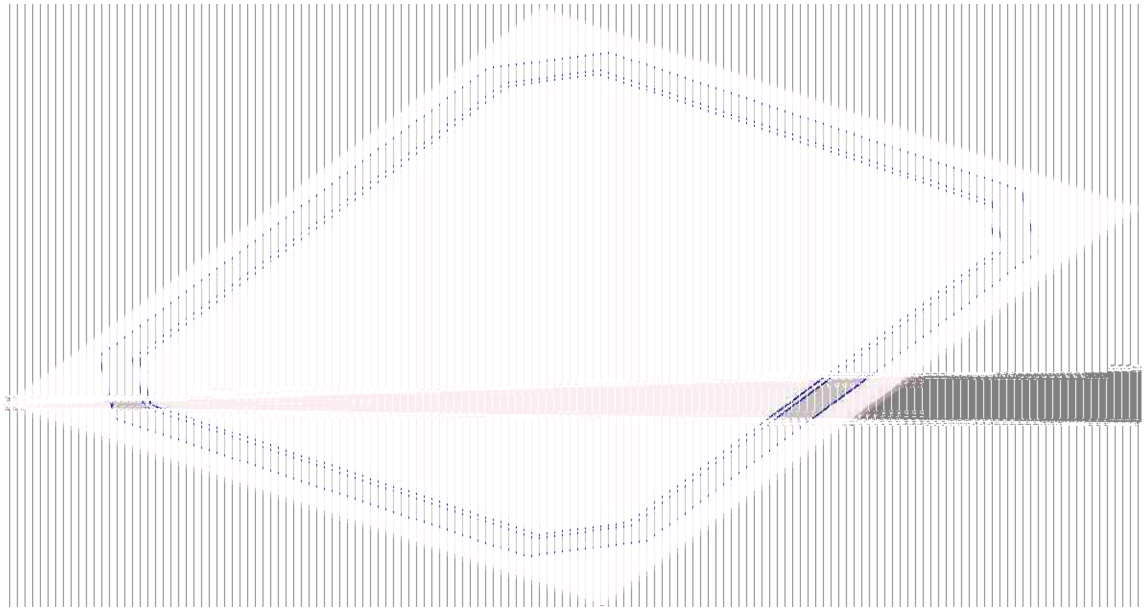
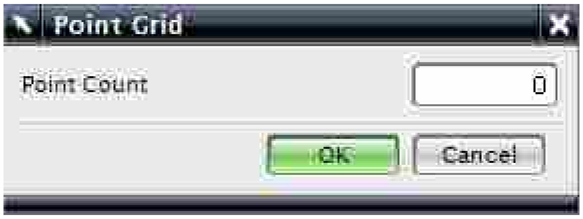


Figure 4-17: 2D Geometry for Enclosure Wall

4.4.2 Inner Envelope Creation

The next objective of button 3 is to create the inner envelope of the enclosure. The inner envelope is defined by a set of higher and lower vertices. The lower vertices are the connecting points of the tolerance offset curves, and the higher vertices are the corner points of the ceiling surface. As was stated in the enclosure creation method, the ceiling surface can take any form. A simplistic ceiling surface would be to simply create a planar surface, as was described in the methods section. This design would be too conservative for the PCB enclosure. The purpose of this research is to not only automate the enclosure design, but to optimize it. As such, an optimal ceiling surface design would mimic the topology of the PCB, and yet not come too close to the building blocks themselves. This is to increase the amount of heat transfer from the PCB and still maintain a safe distance for flash arcing. For this reason the clearance zones were created to control the formation of the ceiling surface.

The ceiling surface created for the PCB is a b-surface that is defined by a grid of points. The number of points is specified by the user through the use of a GUI (Figure 4-18).


Point Grid

Point Count 0

Figure 4-18: Point Count for Grid

The number of points determines how accurate the b-surface will mimic the clearance zone topology. If there are too many points, the b-surface becomes rippled due to an effect called ringing (Figure 4-19). If there are too few points, there is not enough definition to match the PCB topology (Figure 4-20). With the right amount of points the b-surface can be smooth and yet closely resemble the clearance zones beneath (Figure 4-21). The right amount of points can be determined through trial and error since each board has a unique collection of large and small components.

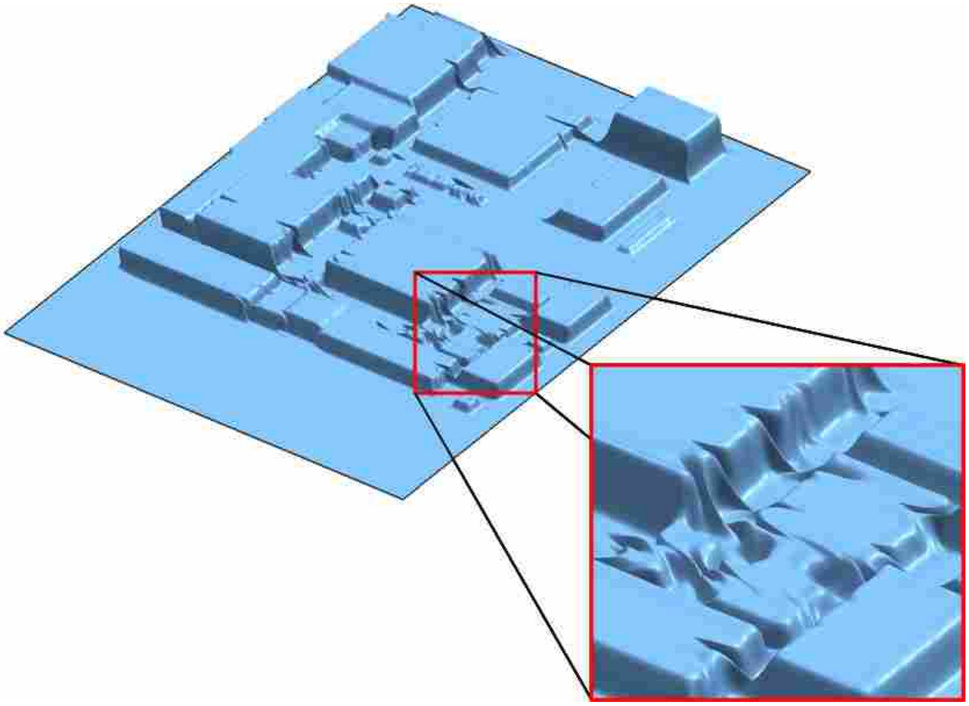


Figure 4-19: Too Many Points for Surface Definition Results in Ringing

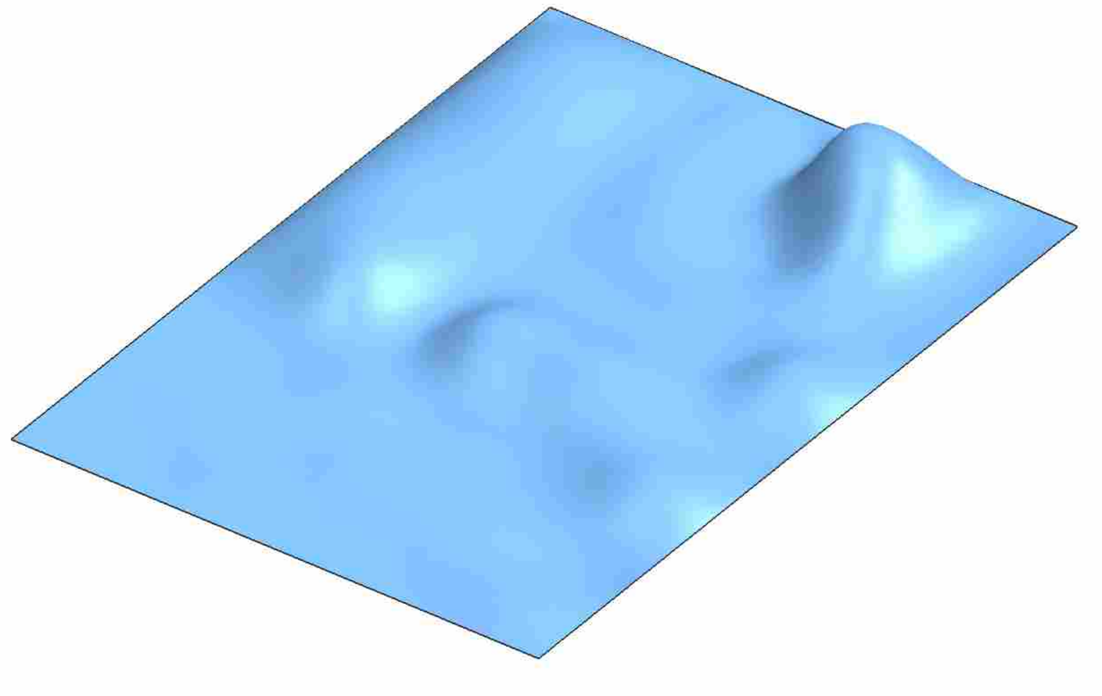


Figure 4-20: Too Few Points for Surface Definition Does Not Match PCB Topology

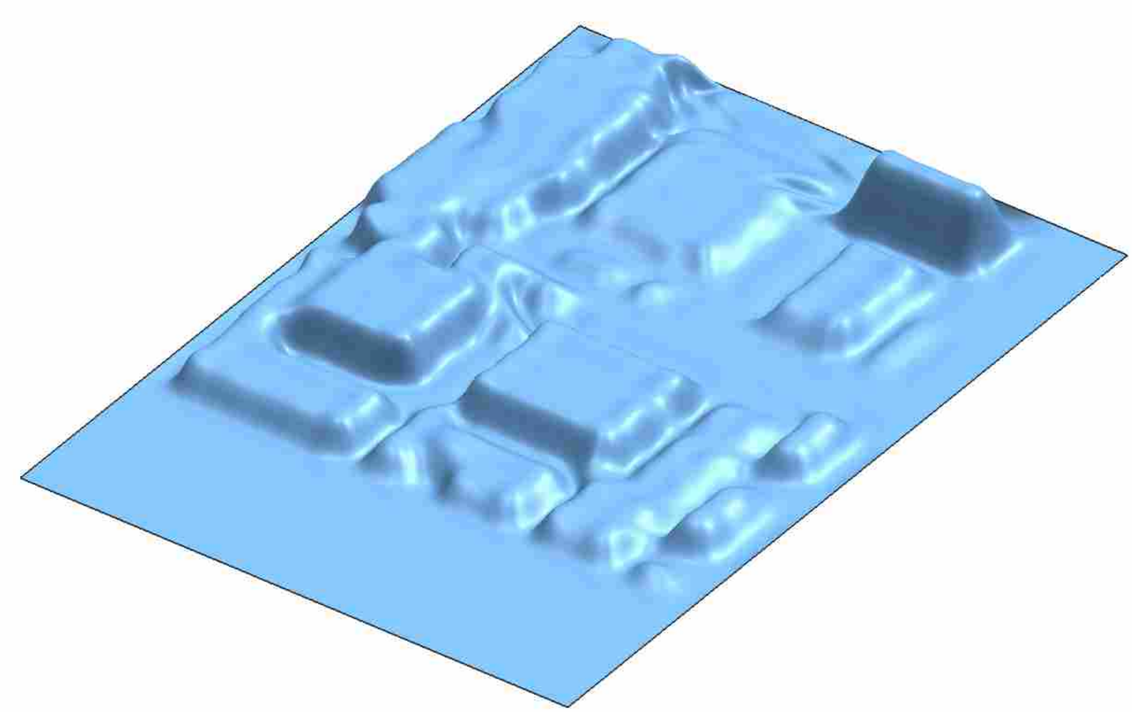


Figure 4-21: Good Point Amount for Better B-surface Representation

The X and Y coordinates of the points are decided by laying the points out in a grid pattern on the XY plane. The boundaries of the grid are the extreme X, and Y values of the wiring board. This means that the maximum X boundary of the grid is determined by the edge that has the largest X value of the board. Similarly, the edge that has the smallest X value determines the minimum X boundary of the grid. The Y boundaries are determined in a likewise fashion. With the grid boundaries specified, the points are placed equidistantly in the X and Y directions (Figure 4-22).

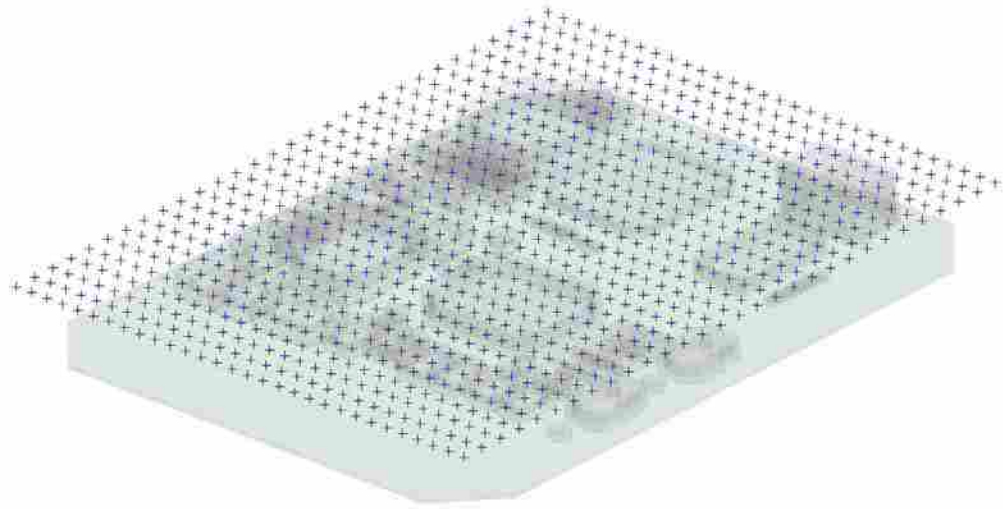


Figure 4-22: Point Grid Definition for X and Y

The Z value for each point is determined by the clearance zones of the building blocks. The Z coordinate takes on the value of the maximum Z coordinate of the clearance zone at that particular X, Y point location (Figure 4-23). Once the points are defined, a b-surface is created to go through all points (Figure 4-24).

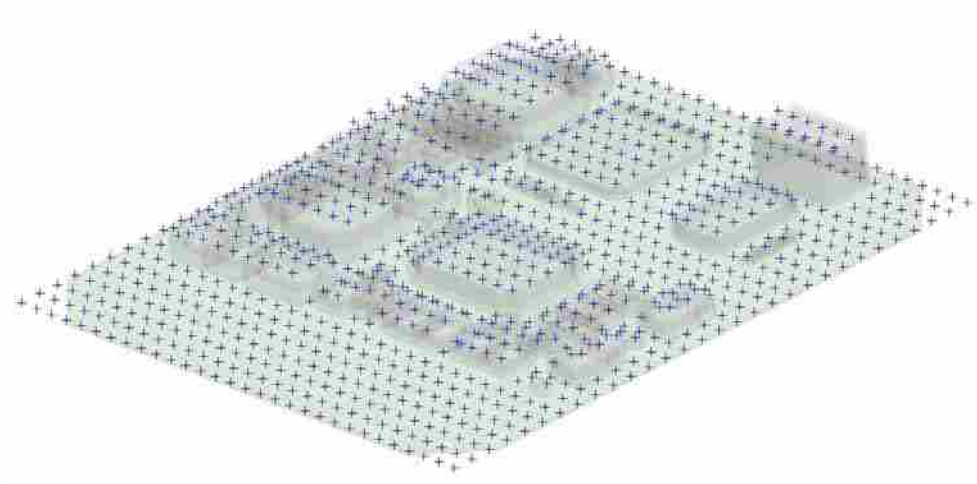


Figure 4-23: Point Grid Definition with Z

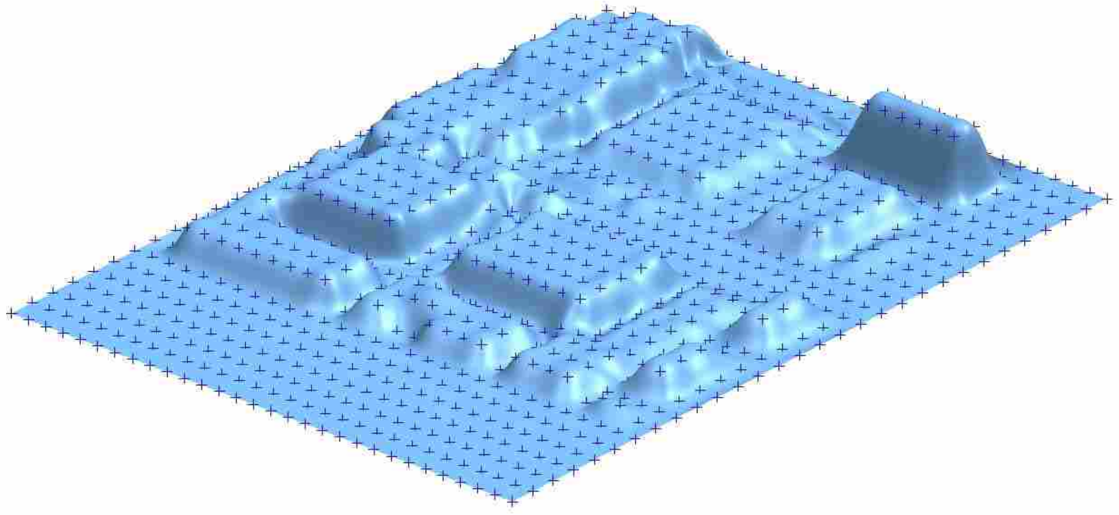


Figure 4-24: Point Grid with B-Surface

In order for the higher vertex points to be paired with the lower ones, the ceiling surface needs to be trimmed to match the vertices from the clearance offset curves. This is done by projecting the tolerance offset curves (defined above) to the ceiling surface in a direction normal to the splitting plane (Figure 4-25). These curves are then used to trim the b-surface (Figure 4-26).

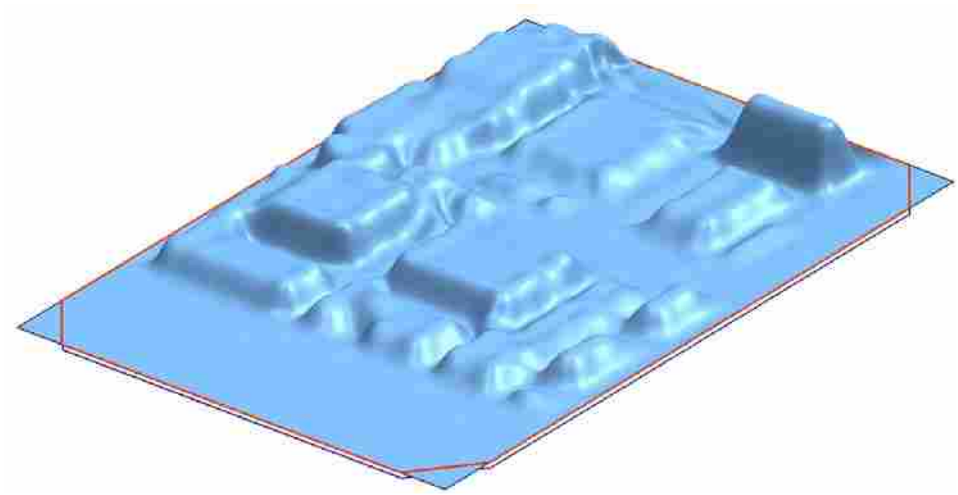


Figure 4-25: B-Surface with Projected Curves

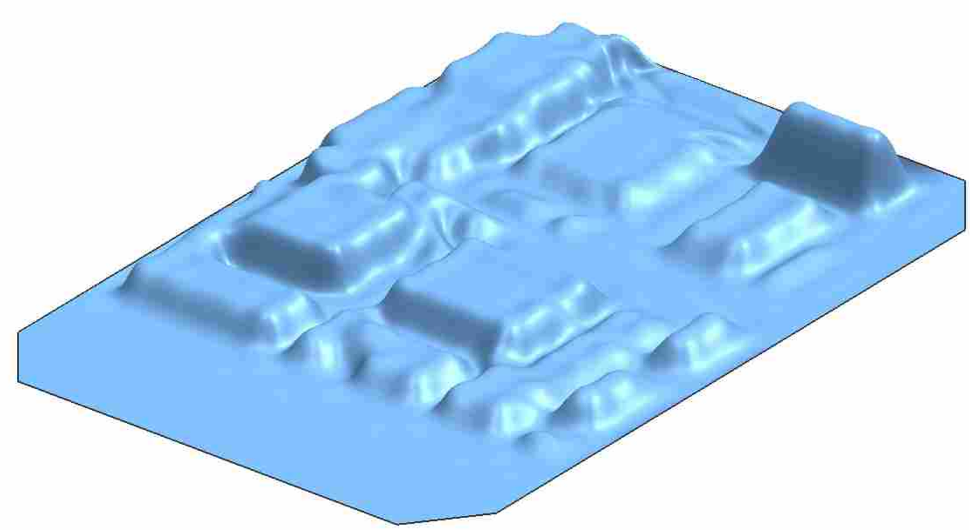


Figure 4-26: Trimmed B-Surface

The next step is to connect the vertices to form edges. The lower and higher vertices are connected by straight lines. These lines are the face edges and with them, the inner envelope of the enclosure is fully defined (Figure 4-27).

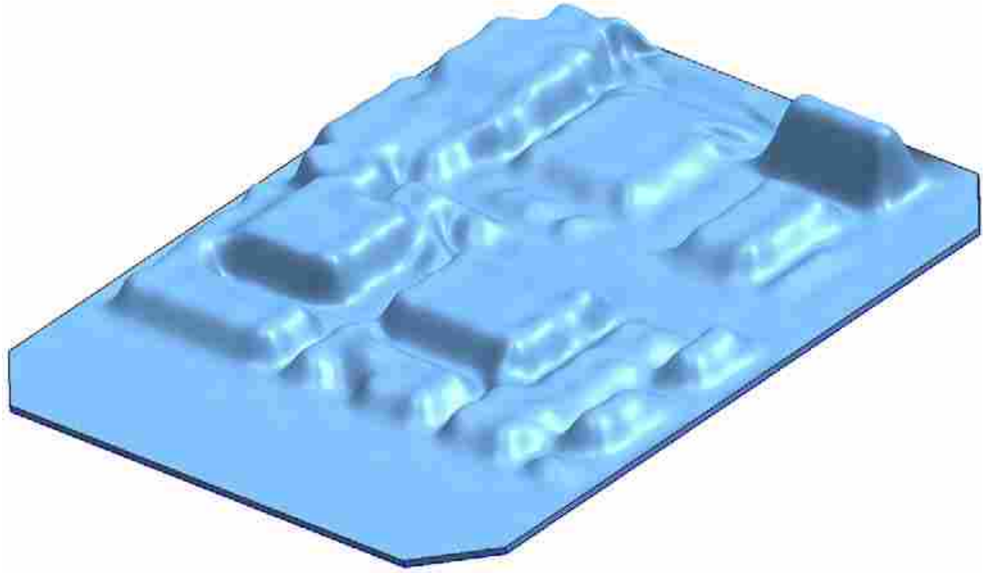


Figure 4-27: Inner Envelope Polyhedron

4.4.3 Outer Shell Creation

The outer shell of the enclosure is much easier to construct. To make the outer shell of the enclosure the structural offset curves vertices are used as the lower vertices. The roof surface for the PCB is a simple planar face that has the same geometry and topology as the structural offset curves. To create the roof surface the structural offset curves are offset in a direction normal to the splitting plane. The offset distance is equal to the maximum clearance zone height, plus the wall thickness (Figure 4-28).

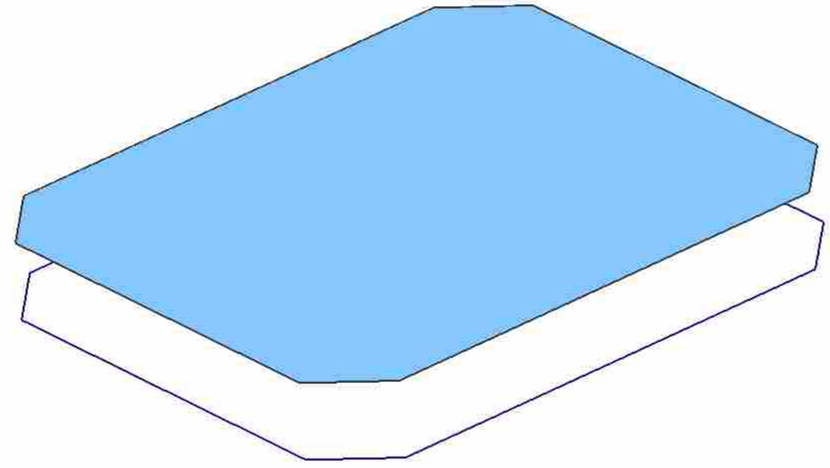


Figure 4-28: Roof Face Offset

As with all polyhedron creation, the vertices are connected by edges, and the edges enclose the faces. The outer shell and the inner envelope make up the inner and outer walls of the enclosure and define a solid body, according to Euler's equation (Figure 4-29).

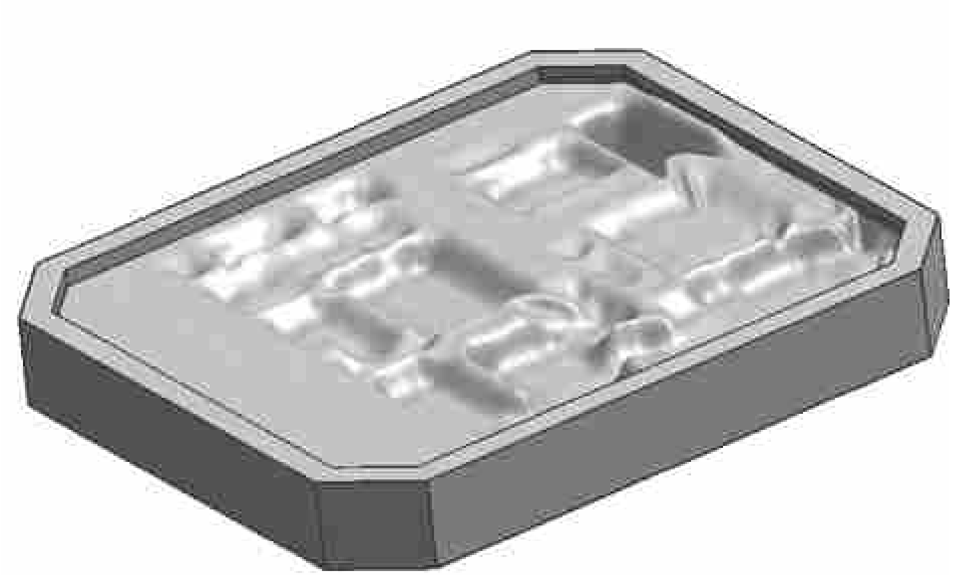


Figure 4-29: Completed Enclosure Design

4.5 Button 4: Heat Sink Design

Buttons 4 and 5 are both used as examples to show the extensibility of the overall process. The main body of the enclosure, created in Button 3, needs much more detailed features to be a fully defined enclosure. The purpose of this research is not to automate the design fully, but to show that it can be accomplished. Therefore, two features are created to enhance the enclosure design discussed in this thesis.

The first feature discussed is a heat sink, which is very important to an enclosure design. It assures that enough heat is transferred out of the enclosure so that the PCB does not overheat. The heat sink feature of the enclosure was chosen for this thesis to again show how knowledge from the ECAD system can determine the design in MCAD. To show how mathematics can be used to generate geometry, the equation shown below is used to determine the number of heat sink fins needed to extract sufficient heat.

$$N = \frac{\frac{1}{R_{t,o}h} - A_b}{A_f \eta_f} \tag{4-5}$$

In the above equation N is the number of fins, Rt,o is the thermal resistance of the fin, h is the convection coefficient, A_b is the base area, A_f is the fin surface area, and η_f is the fin efficiency. The derivation of the thermal equation used to calculate the number of fins is complicated and lengthy and is not shown here. For a full derivation of the equation refer to appendix A. The inputs needed for this equation were gathered from the user by way of a GUI.

In order to solve the equation, the total amount of heat that is generated from the PCB must be calculated. The total amount of heat is the summation of the heat generated by each component. During the file parsing process the thermal output of each component was gathered from the ECAD package. This information is read from the part file attributes, summed together and the number of fins is calculated.

When the required number of fins is known, the geometry can be created. The fin geometry is created to be a simple rectangular polyhedron. This geometry was preferred to simplify the mathematics of the heat transfer equations. The lower vertices of each fin are placed on the flat surface of the roof of the enclosure and extruded to a height specified by the user in a direction normal to the roof face. The fin geometry is shown below (Figure 4-30).. The overall enclosure design is more functional

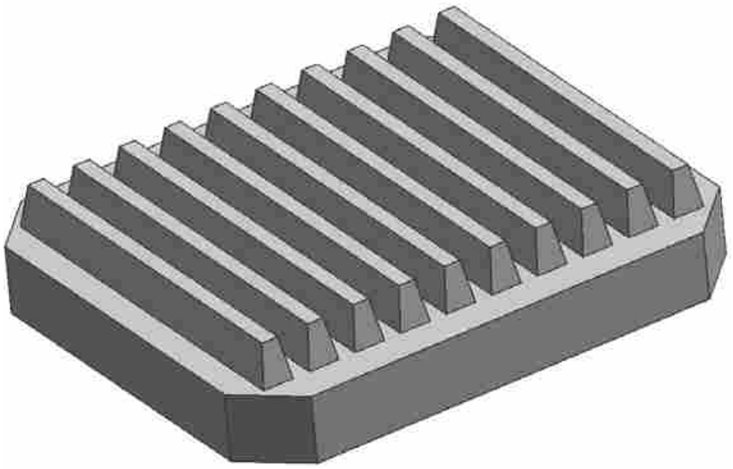


Figure 4-30: Enclosure with Heat Sink Fins

4.6 Button 5: Mounting Brackets

Another extensible feature to add to the enclosure design is the mounting bracket. Mounting brackets can be any shape or size, but for this research only one bracket design is described. To begin the bracket design, the user is prompted by a GUI to choose the X, Y, and Z coordinates for the center of the bracket hole. The user is then prompted to enter values for the hole diameter, the bracket wall thickness and a height.

The program then uses this information to create the vertices and curves of the bracket shape on a 2D plane (Figure 4-31 (a)). This geometry is extruded to the height specified by the user and the bracket solid body is formed (Figure 4-31 (b)).

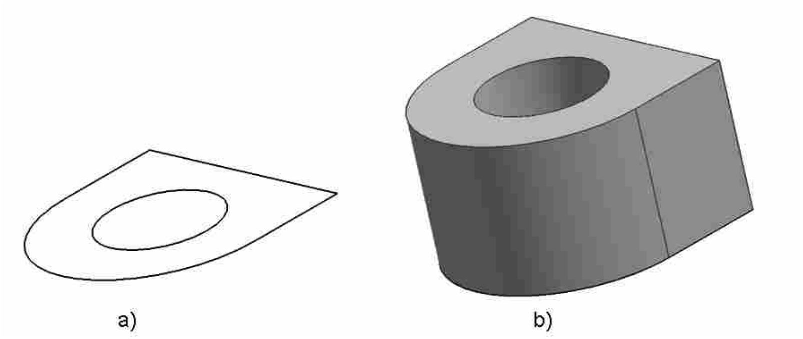


Figure 4-31: Mounting Bracket Geometry Creation

The final step to this process is to perform a Boolean unite function on the bracket solid body and the enclosure solid body. This joins the two geometries and makes one single solid body (Figure 4-32). If the method is followed correctly, a Boolean operation will always result in geometry that obeys Euler's equation.

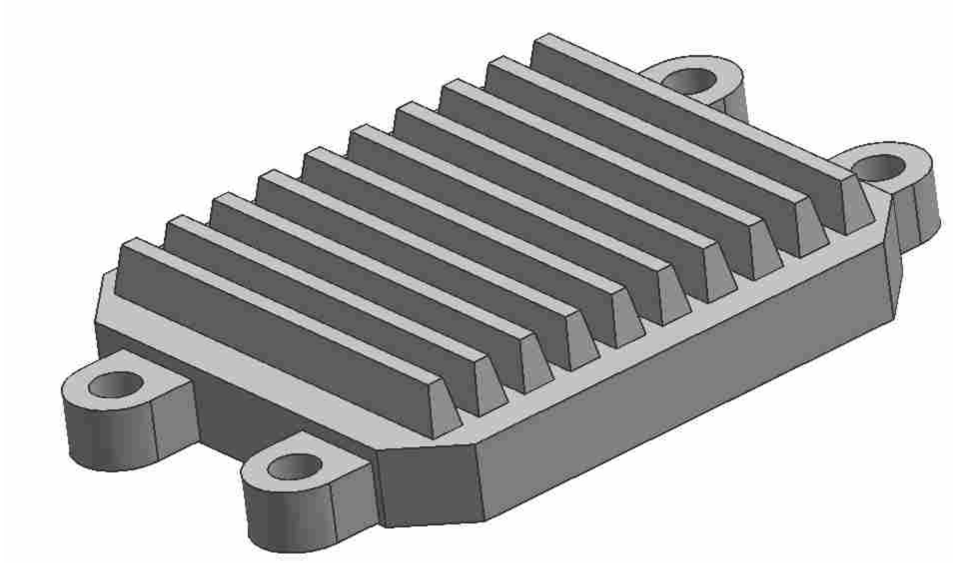


Figure 4-32: Enclosure with Mounting Brackets

4.7 Conclusion of Implementation

The enclosure design method that was implemented in this chapter created a comprehensive system having mechanical, electrical, and thermal rules built-in to the enclosure design. The process to achieve the preliminary design was performed automatically with little input from the user, thus greatly decreasing the design time. With the decrease of time, the designer has more time to optimize and perfect the full enclosure design. The framework of the automated design process allows for other rules to be added into the design. For example, vibration, fluid flow, magnetic, etc. rules can also be implemented to further define a working enclosure design.

The implementation discussed in this chapter was to show that rules extracted from an ECAD package can be used to create a preliminary design. To more fully design a complete enclosure, input is needed from experienced designers who better understand the design process.

5 Results

To reiterate what was stated in Chapter 1, the objective of this thesis is to overcome the two major issues facing enclosure design.

1. The loss of information when transferring data from ECAD to MCAD
2. Designing a safe electronic enclosure design more efficiently

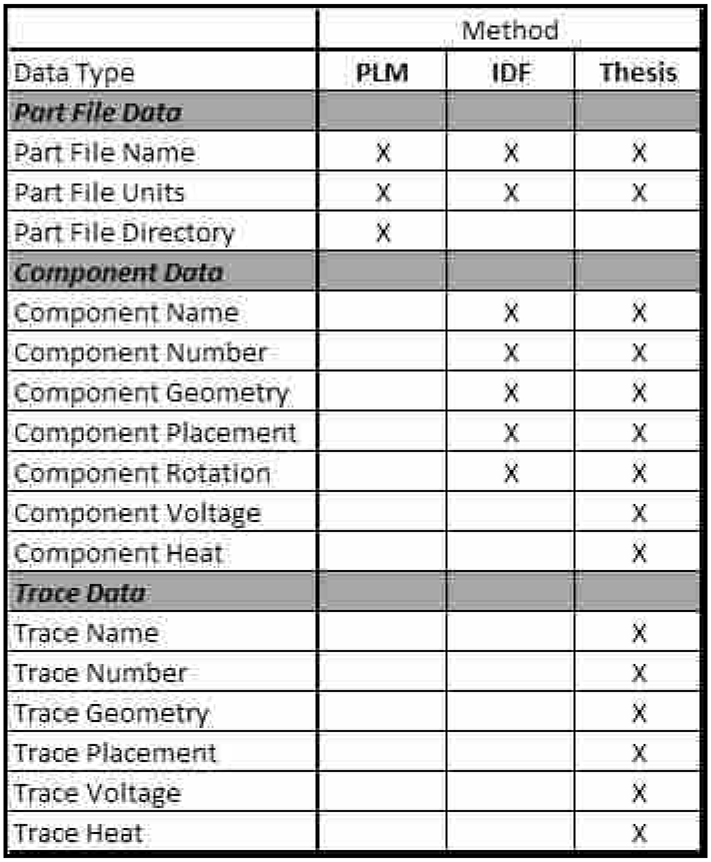
In order to determine the success of this thesis, the three criteria, given in Chapter 1, will be evaluated and discussed below. The three criteria are:

1. The amount of information communicated between the two CAD packages
2. The reduction of time it takes to create a preliminary case design
3. The quality of the enclosure design to ensure safety

5.1 Information Exchange

The success of the amount of information communicated between the two CAD packages will be determined by a comparison of the amount of data transferred described in the thesis to the amount of data transferred using conventional data transferring practices. As was discussed in Chapter 2, in today's industry most companies will either use PLM packages to exchange information, or use IDF file transferring methods. The table below compares the amount of data each method can transfer from the ECAD system to the MCAD system (Table 5-1).

Table 5-1: Data Transfer Comparison



	Method		
Data Type	**PLM**	**IDF**	**Thesis**
Part File Data			
Part File Name	X	X	X
Part File Units	X	X	X
Part File Directory	X		
Component Data			
Component Name		X	X
Component Number		X	X
Component Geometry		X	X
Component Placement		X	X
Component Rotation		X	X
Component Voltage			X
Component Heat			X
Trace Data			
Trace Name			X
Trace Number			X
Trace Geometry			X
Trace Placement			X
Trace Voltage			X
Trace Heat			X

The results from the table are conclusive. The method described in the thesis significantly transfers more data from ECAD to MCAD. The ability to transfer trace information is a great advantage when designing an enclosure. Many enclosure designs are poor because the inner envelope of the enclosure is designed too closely to the traces. The root cause of this error comes from not having the capability to transfer the information into MCAD.

The ability to transfer voltage and heat information is invaluable to the enclosure design process. As was described in the thesis, this information was not used for analyzing the enclosure design, but to generate it.

Another added benefit the thesis method provides, is the ability to adapt the data transfer to specific procedures. The electrical engineer can add more attributes to be exported from the ECAD package that would be useful for designing the enclosure in MCAD. For example tolerances, vibration endurance, cost of parts, stress yields, materials, etc. all affect enclosure design. By following the file parsing method, it would be easy to also map this additional information to the MCAD system.

5.2 Time Reduction

The measure of success for the time reduction criteria will be determined by the time saved in automating the enclosure design process. To accomplish this, a simple PCB (Figure 5-1 (a)) was provided to 4 students with the instruction to manually build an enclosure to house it (Figure 5-1 (b)). The steps to create the enclosure were clearly defined and given to each student. The results of the experiment are provided in the table below.

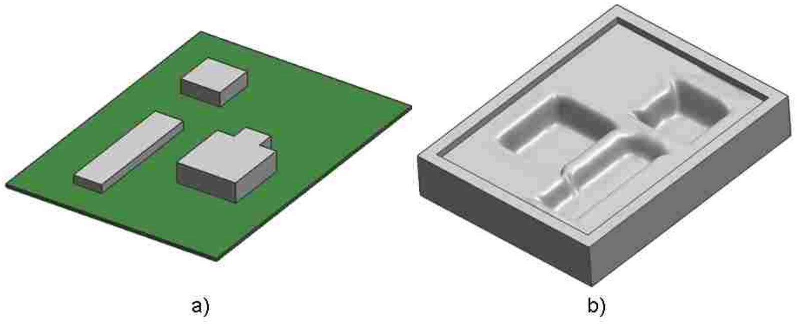
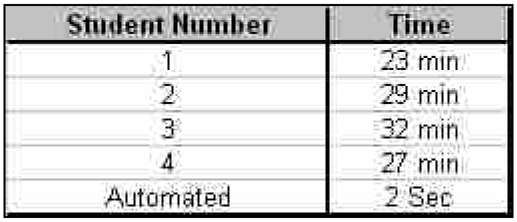


Figure 5-1: Simple PCB Test Case

Table 5-2: Time Results



Student Number	Time
1	23 min
2	29 min
3	32 min
4	27 min
Automated	2 Sec

As the table shows, the automated process is approximately 700 times faster than manually creating the geometry. These time savings are shown only for a simple board. The time savings will greatly increase when working with larger boards. It should also be noted that the exact steps were provided to each student, meaning they did not need to conceptualize the design before implementing it. The automatic case generation not only saves time in creating an enclosure, but saves time in conceptualization.

5.3 Enclosure Design Quality

The main objective of this thesis is to create an enclosure design that ensures a safe design for the customer to use. The two main risks when making an enclosure design is that of arc flashing, and overheating. The traditional process of enclosure creation involves designers using their best judgment to design the model. Once this is done, analyses are performed to see if the design is valid. The method described in this thesis is the reverse of the traditional process. The analysis constraints (safe voltage distance, heat transfer) are used to create the design, resulting in an enclosure that is safe.

Since the design was defined by voltage and heat transfer rules, it would be illogical to base the success of the design by these same standards. Therefore, to measure the success of the methodology, two additional PCBs were tested to verify that the process will work with any PCB layout. The results of the tests are shown in the figures below.

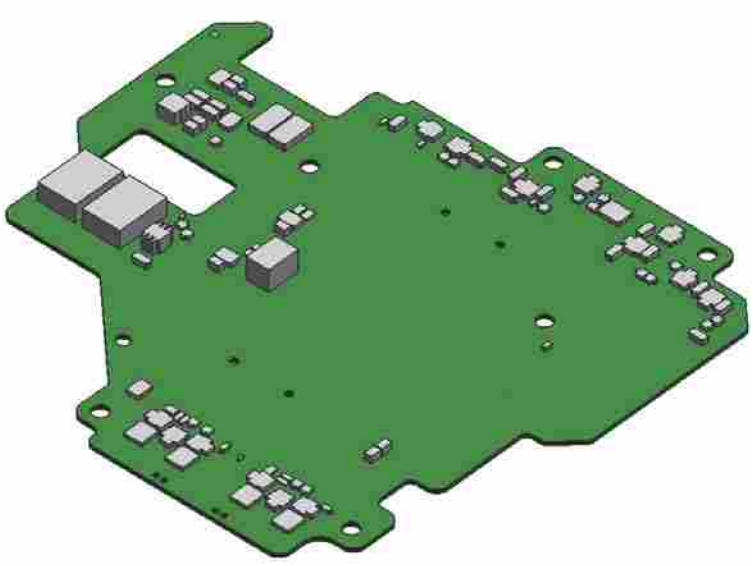


Figure 5-2: PCB Test 1

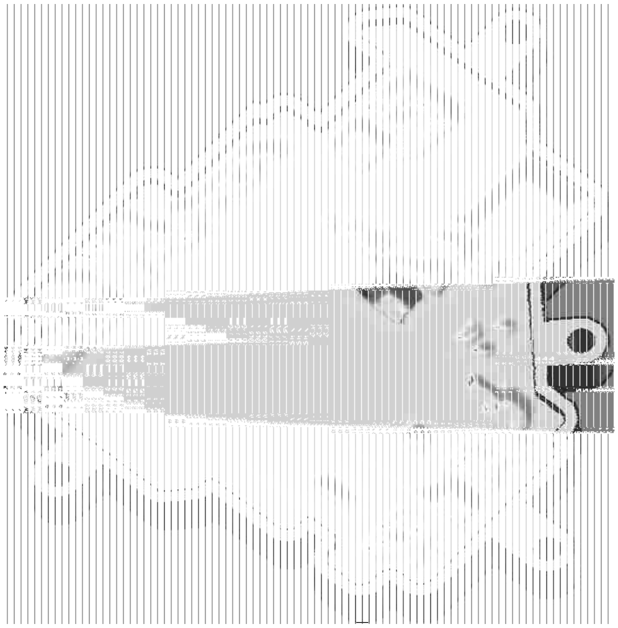


Figure 5-3: Enclosure Test 1

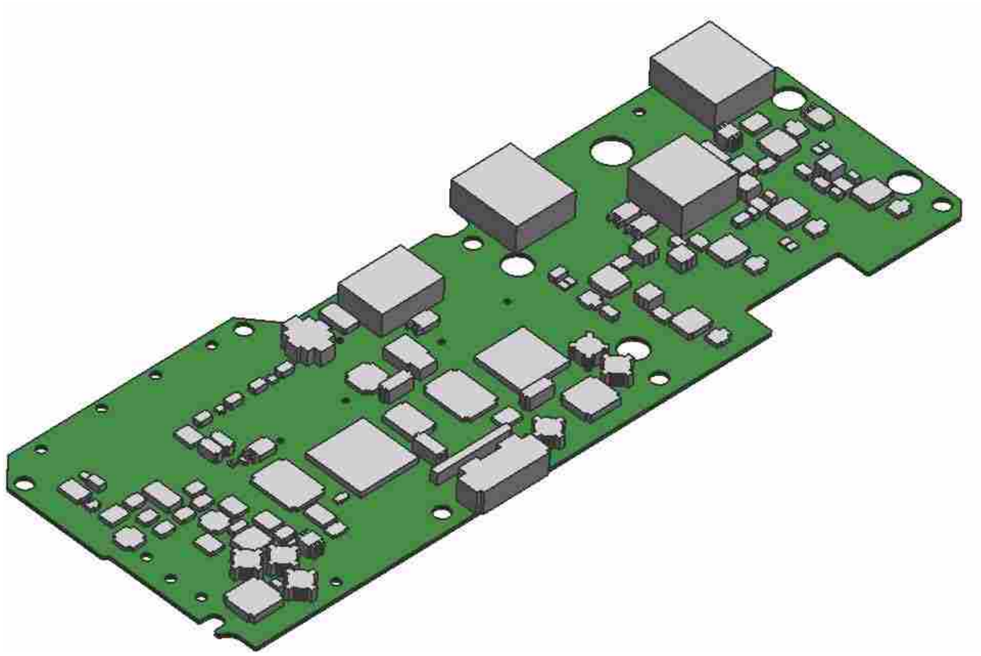


Figure 5-4: PCB Test 2

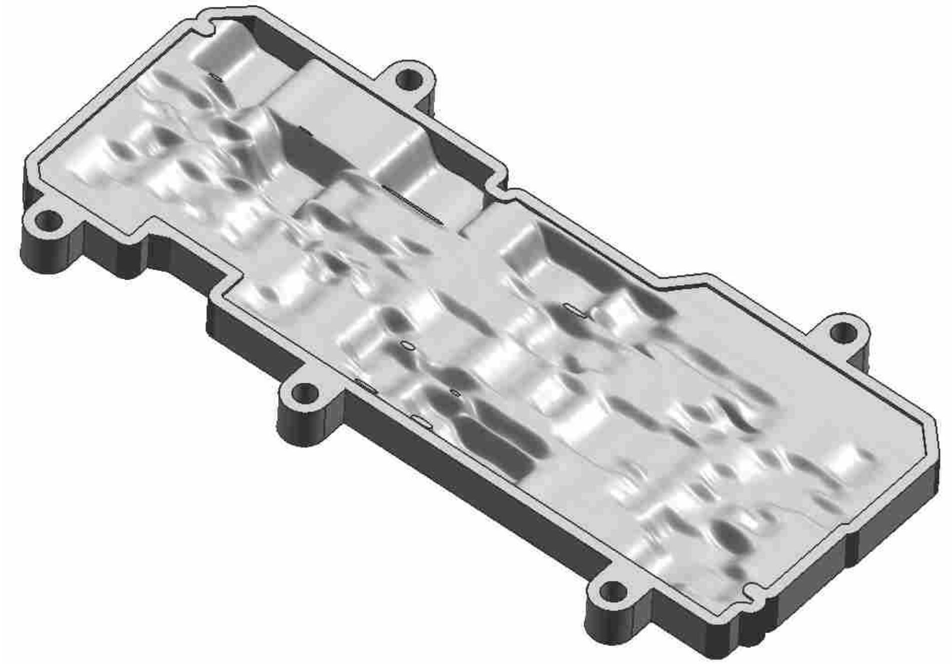


Figure 5-5: Enclosure Test 2

The two PCBs tested were chosen because of their differing complex shapes and assemblies. The first test case shows that the ceiling b-surface can be created on PCBs where there are groupings of components. This means that the process can handle areas where more definition is needed and still create a relatively smooth surface. The first test case also shows a very complex wiring board design. The automated process was able to create an enclosure to fit and house even complex shapes. The second test case was chosen for its rectangular shape. The process shows that enclosures with elongated board designs can still be automatically generated.

6 Conclusions

The objectives of this thesis were to improve the communication from ECAD to MCAD, and then use the added information to automatically generate an enclosure. The reason this is needed is to eliminate the problems that occur with miscommunication between engineering disciplines.

One of the first issues discovered in researching the current design cycle was the generalized nature of existing file transfer methods (IDF). To overcome this problem, a file parsing method was discussed in detail in Chapter 3. The method was implemented in Chapter 4 to customize the data extracted from the ECAD system for the use of automated enclosure design. This customization can be used on any CAD/CAE system that uses an ASCII file format, making it applicable to any product design process using multiple CAD/CAE packages. Chapter 5 demonstrated that the file parsing process described for this thesis has vast improvements over existing data transfer methods.

Chapter 3 also discussed basic modeling methods used by CAD packages to create preliminary enclosure designs. The methodology behind enclosure creation was automated using C style programming and was discussed in Chapter 4. The program developed not only created the enclosure automatically, but also used the physical rules from ECAD to conceptualize the overall design of the housing. The automation led to a 700X speed-up in the design cycle. These results prove that physical knowledge and mathematics can be used to generate a design, and that enclosure model creation can be streamlined to expedite the design process.

The last section in chapter three discussed a method to make the overall process extensible. This is done be performing Boolean operations on the enclosure. Two separate features were added to the design as an example to prove the method. The heat sink fins were created using mathematics to determine the number of fins needed to extract enough heat away to prevent overheating. The mounting brackets were created to further show how easily features can be added to or subtracted from the enclosures main body to better resemble a final design.

6.1 Recommendations

Chapter 1 stated that this thesis was not intended to create a fully designed enclosure for all types of enclosures. The purpose of this thesis was to prove a concept. The concept is that knowledge contained in one CAD/CAE system can be completely transferred to another CAD/CAE system and used to automate the design of the enclosure.

Chapter 5 showed that the results are promising; however, there are many improvements that can be made. The following recommendations are suggested to increase the power of this tool and create a more defined and detailed enclosure.

The first recommendation is to implement the idea of parametrics and associativity into the design. A parametric model is one that allows the user to change a dimension and have the model update to accommodate it. This allows the designer to explore more design options and optimize the model. Associativity means that geometry of one solid object is associated or linked to the geometry of another object. For example, the clearance zones created around the component models could be associated to the components themselves. If an ECAD designer needs to shift the component the clearance zone will follow. Similarly the ceiling b-surface could be associated to the clearance zones and its form would change when changes are made to the clearance zones.

Another recommendation would be to more fully incorporate tolerance stack-up methods. A tolerance analysis can be very useful for optimization of parts and processes. It would also be useful to combine the various stack-up methods for a better prediction of the standard deviation limits (see Appendix B).

Another recommendation is to use the physical rules and the automated design process to optimize the enclosure design. Optimization techniques can be used to increase thermal transfer, decrease weight, decrease material used, increase manufacturability, etc. Optimization techniques can be included directly into the automation process.

The next recommendation would be to create smoother b-surfaces for the inner envelope ceiling. It would be difficult to manufacture a surface like the ones shown in this thesis. There are mathematical equations that can be used to smooth b-surface definition to create more manufacturable enclosure designs.

It is also recommended that further research be made in creating enclosures that house more than one PCB. This process would be more difficult, but would lend itself useful to more enclosure designs needed in today's industry.

Another recommendation is to use detailed 3D geometry to define the building blocks. Many components are made of non-conductive material and therefore have no risk of shorting. If only the conductive parts of components were used to create clearance zones, the inner envelope space could be greatly optimized.

It is also recommended to incorporate testing and simulation into the automation process. The process could also link other CAE software tools to perform thermal and electrical analysis of the enclosure design. This could potentially locate hotspots or areas where flash arcing is more likely to occur. Simulation of the manufacturing process could also be included into the design process to optimize the manufacturability of the design.

One last benefit needed for this type of tool is to create a bi-directional form of communication between ECAD and MCAD. This research only dealt with communicating from ECAD to MCAD; however it would be just as valuable to have the ability to communicate more information from MCAD to ECAD. With this information the ECAD designer could optimize component layout for a more efficient use of board space thus creating a better electrical design.

This is an initial study of problems and challenges facing automated enclosure design. Guidelines have been established for the development of a comprehensive system, which will permit effective communication between MCAD and ECAD software with commercial CAD systems, with capability to organize and store the full range of data, from geometry, to material property, to design rules. Customized systems will permit enclosure design processes to be faster, more efficient, leading to superior products at lower cost.

7 References

Ahn, W. and Agonafer, D. (2004). "Methodology for an Integrated (Electrical/Mechanical) Design of PWBA" *ASME*

Breedveld, P. C. (2004). "Port-Based Modeling of Mechatronic Systems" *Mathematics and Computers in Simulation*

Chase, K. W. (2004). "Chap. 7 –Basic Tools for Tolerance Analysis of Mechanical Assemblies" *Manufacturing Engineering Handbook*. Geng, Hwaiyu (editor), McGraw-Hill.

Feldman. K. and Franke, J. (1993). "Computer-Aided Planning Systems for Integrated Electronic and Mechanical Design" *IEEE Transactions on Components, Hybrids, and Manufacturing Technology.*

Garfinkel, G. A. (1998). "Integrating Mechanical Computer-Aided Engineering Tools into the Printed Wiring Board Design Cycle" *IEEE InterSociety Conference on Thermal Phenomena*

Gayretli, A. (2007). "20DE-IPD: An Object-Oriented Design Environment For Robust and Reliable Interdisciplinary Product Design" *Key Engineering Materials*

Hsiao, S. W. (1999). "Integrated Concurrent Engineering Based Approach for Electric-Fan Design" *Journal of Integrated Computer-Aided Engineering*

Kowalski, W. P. and Roman, M. (2004). "Autodesk Inventor Professional as Common Platform CAD for Designer from Mechanic and Elecromechanic Profession" *International Conference TCSET'2004*

Manufacturing Business Technology (2007). "A United Framework" *White Paper Series*

Mentor Graphics (2007). "A Strategy for Transformation" *White Paper Series*

Odell, D. and Wright, P. (2002). "Concurrent Product Design: A Case Study on the Pico Radio Test Bed" *7th Design for Manufacturing Conference*

PTC (2008). "Overcoming the Top Five Challenges in Electro-Mechanical Product Development" *White Paper*

Salzman, H. (1989). "Computer-Aided Design: Limitations in Automating Design and Drafting" *IEEE Transactions on Engineering Management*

Staton, M. (1985). "Computer-Aided design" Occupational *Outlook Quart., Spring* 1985

Wang, F. and Wright, P. (1996). "A Multidisciplinary Concurrent Design Environment for Consumer Electronic Product Design" *21st Annual Design Automation Conference*

Zhou, W. X. (1997). "CAD-Based Analysis Tools for Electronic Packaging Design" *Innovations in CAD/CAE Integration in Electronic Packaging, INTERpack'97, Kohala, HW.*

Appendix A. Fully Derived Fin Number Calculations

The picture below shows the setup and variables used for a thermal heat transfer equation. The variable names are also given below in Table A-1.

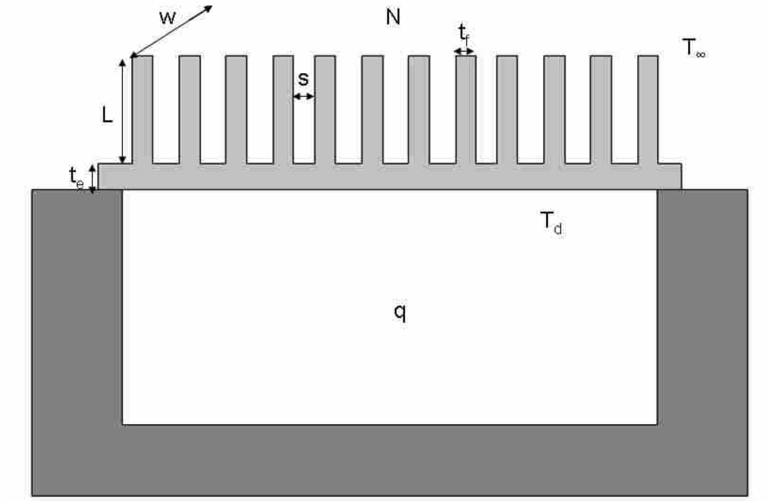
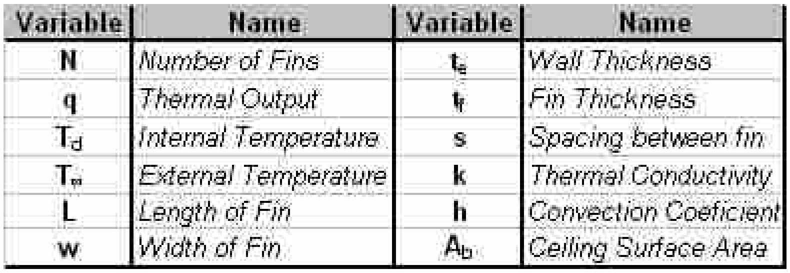


Figure A-1: Heat Transfer Fin Setup

Table A-1: Variables



Variable	Name	Variable	Name
N	*Number of Fins*	t_e	*Wall Thickness*
q	*Thermal Output*	t_f	*Fin Thickness*
T_d	*Internal Temperature*	s	*Spacing between fin*
T_∞	*External Temperature*	k	*Thermal Conductivity*
L	*Length of Fin*	h	*Convection Coeficient*
w	*Width of Fin*	A_b	*Ceiling Surface Area*

Standard Fin Math Equations

Thermal Resistance Equation: $q = \frac{T_d - T_\infty}{\sum R}$ **where**: $\sum R = R_{t,c} + R_1 + R_{t,o}$

Contact Resistance: $R_{t,c} = Constant$

Wall Resistance: $R_1 = \frac{t_e}{kA_b}$

Fin Thermal Resistance: $R_{t,o} = \frac{1}{\eta_o h A_t}$

Fin Area: $A_f = 2wL_c$

Total Fin Area: $A_t = NA_f + A_b$

Corrected Fin Length: $L_c = L + \left(\frac{t_f}{2}\right)$

Fin Efficiency: $\eta_f = \frac{\tanh mL_c}{mL_c}$ **where**: $m = \sqrt{\frac{2h}{kt_{f\,c}}}$

Overall Fin Efficiency: $\eta_o = 1 - \frac{NA_f}{A_t}(1 - \eta_f)$

Fin Number Derivation

The goal of this derivation is to find an equation to solve for N (number of fins).

1) To start the Fin Thermal Resistance equation is rearranged to get the variables with N on one side:

$$\eta_o A_t = \frac{1}{R_{t,o} h} \qquad \textbf{(A-1)}$$

2) Then substitute the Overall Efficiency equations in:

$$\left[1-\frac{NA_f}{A_t}(1-\eta_f)\right]\left[A_t\right]=\frac{1}{R_{t,o}h} \tag{A-2}$$

3) The next step is to multiply A$_t$ through:

$$A_t - NA_f(1-\eta_f)=\frac{1}{R_{t,o}h} \tag{A-3}$$

4) Now substitute in the Total Fin Area equation and multiply NA$_f$ through:

$$\left(NA_f + A_b\right)-(NA_f - NA_f\eta_f)=\frac{1}{R_{t,o}h} \tag{A-4}$$

5) Through basic algebra the equation becomes:

$$A_b + NA_f\eta_f = \frac{1}{R_{t,o}h} \tag{A-5}$$

6) Next subtract A$_b$ from both sides of the equation:

$$NA_f\eta_f = \frac{1}{R_{t,o}h} - A_b \tag{A-6}$$

7) The last step is to divide both sides by A$_f\eta_f$:

$$N = \frac{\frac{1}{R_{t,o}h} - A_b}{A_f\eta_f} \qquad \textbf{where:} \qquad R_{t,o} = \frac{T_d - T_\infty}{q} - R_{t,c} - R_1{}^{*} \tag{A-7}$$

***** **-** This is derived from the Thermal Resistance Equation defined above

Appendix B. Discussion on Tolerance Stack-Up

Combination of Methods

One of the benefits of tolerance analysis is that the statistical methods described in Section 3.4 may be combined together. When the tolerances are well defined, the RSS method is desirable. Similarly, when there is not much data on a set of tolerances the UD method is recommended. The proposed equation for the overall tolerance stack-up can therefore be modified as follows, where each type of variation is grouped into one of two categories.

$$\sigma_{total} = \sqrt{\left(\frac{\Sigma(T_i)}{3}\right)^2 + \left(\frac{\Sigma(T_i)}{\sqrt{3}}\right)^2} \qquad \textbf{(B-1)}$$

This method results in a more realistic representation of the variation within the assembly. It avoids the more costly WC constraints.

Mean Shift

When calculating the overall clearance for enclosure design, the tolerances are combined together and their statistical average becomes the nominal overall clearance value. There are times however, when the mean value (average) needs to be adjusted. This adjustment is called a mean shift. For example, when determining the clearance for the widget used in Chapter 3 the thermal expansion of the widget causes the overall clearance value to increase. In other words, because the widget expands with temperature, there needs to be more clearance allotted for the enclosure design. This adjustment shifts the average clearance and can decrease the product yield.