2008-03-19

Robust Parameterization Schema for CAx Master Models

Courtney L. Berglund
Brigham Young University - Provo

ROBUST PARAMETERIZATION SCHEMA

FOR CAX MASTER MODELS

by

Courtney L. Berglund

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

April 2008

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Courtney L. Berglund

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	Walter E. Red, Member
Date	Kenneth W. Chase

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Courtney L. Berglund 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

Matthew R. Jones
Graduate Coordinator

Accepted for the College

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

ABSTRACT

ROBUST PARAMETERIZATION SCHEMA

FOR CAX MASTER MODELS

Courtney L. Berglund

Department of Mechanical Engineering

Master of Science

Today's engineering companies rely heavily on an engineer's ability to use computers to analyze and optimize designs. With this use of computers in the design process, products undergo multiple design iterations between preliminary concept and final form. This in turn results in Computer Aided Design (CAD) models being passed from one discipline to the next. In attempts to keep consistency within the design process, an industry wide shift towards the use of CAD master models is taking place.

With this change to master models, manufacturing and engineering development companies are attempting to more fully employ the use of parametrics in their initial CAD models. This is in hopes that the initial models handed downstream are robust enough to be used throughout the entire design loop. Unfortunately, current parameter definitions are often not robust enough to incorporate all the design changes from the various analyses and manufacturing operations. To address this problem, we present a

more robust parametric methodology that broadens the current definition of parametrics as currently employed on CAx master models within CAD packages.

ACKNOWLEDGMENTS

I wish to thank my family, for their support, Dr. C. Greg Jensen for his guidance and Pratt & Whitney for funding and supporting all of my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Today, engineering companies rely heavily on their engineers' ability to use computers to analyze and optimize designs. From a preliminary concept to final form, products undergo multiple analysis and manufacturing iterations. Because of these multiple design and analyses iterations, the design process can be thought of as being cyclic as outlined in Figure 1-1 [1].

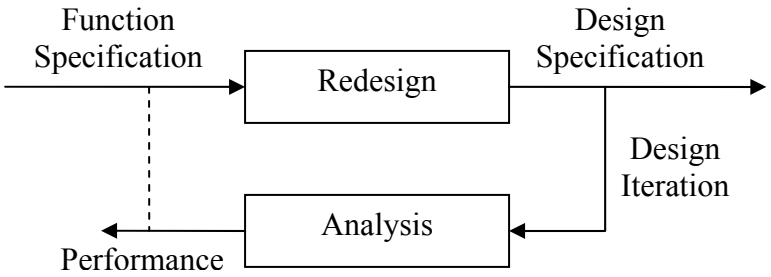

Figure 1-1 The cyclic design process.

With an increase in the number of design iterations, Computer Aided Design (CAD) models are constantly being passed from one discipline to the next. In attempts to keep consistency within the design process, current trends are showing a shift towards the use of CAD master models (CADmm). CADmm are intended to be used in multiple, if not all, phases of the design process. With the change to CADmm schema, manufacturing and engineering development companies are attempting to more fully employ the use of

parametrics within their initial CADmm so instantiated models handed downstream are robust and can be used throughout the entire design loop.

This CADmm shift is however, often thwarted. Commonly, current parameter definitions are not robust enough to incorporate all the design changes inherent with the design and manufacturing processes. Current parametric methodology involves single value representation parameters to define part dimensions. Because of this single value definition, these parameters do not allow for easy integration of the changes from the various analyses and manufacturing operations. As a work around, the various disciplines will manually edit the CADmm for their respective operations. They will change the topology and single value parameters to fit their specific objectives and, in the process, lose the topology associativity and previous parametric knowledge stored within the CADmm.

1.1 Objective

To address these problems, the objective of this thesis is two fold. First, a methodology that redefines and extends the parameter definition currently employed within CAD packages will be developed. This new parametric methodology will transition traditional single value representation to a multiple value representation incorporates the use of functional equations with design and manufacturing factors integrated within the functional parameter definition equation.

The second objective of this thesis will be to demonstrate the validity and robustness of this new parametric definition. This will be accomplished by testing the methodology on two different example parts and comparing the results with industry

parts not employing the modified parameter definitions. In testing the methodology, a user defined function will be developed to implement the method within a commercial CAD package.

In particular, this thesis will answer the following questions:

- What specific factors should be included in the parameter definition equation?
- How can the various factors be included in the parameter definition equation to increase factor independence and reduce or eliminate factor interaction?
- What are the steps and how can the new parametric schema be implemented to work within commercial CAD packages?

1.2 Proposed Benefits

Extending the definition of parametrics will yield multiple benefits to the current design process. First, there will be greater flexibility added into the parametric modeling process. Designers will be better able to cope with the various uncertainties inherent with the design process, creating more robust CADmm. Also, by including greater flexibility into CADmm from the onset, less time will be spent reparameterizing and updating CADmm, allowing designers time for what they do best, creating more designs and products.

Secondly, expanding the definition of parametrics will increase the design knowledge stored within CADmm to be used and interpreted in downstream operations. Additionally, the process of incorporating this new methodology within commercial CAD packages will allow users to automatically create driving parameters based on more robust parameter definitions.

Finally, by expanding the definition of parametrics, the CADmm concept can be more completely applied and incorporated within the current design process. The resultant model consistency will greatly reduce confusion and compatibility issues inherent with passing different models to and from disciplines through different development phases. Also, by having this parameter knowledge incorporated into the models from conception, higher fidelity analysis and research can be conducted on CADmm. Specifically, analyses concerning how parts will actually perform after the various manufacturing are completed on the product can be conducted.

1.3 Thesis Outline

The remainder of this thesis will be organized as follows:

- Literature review addressing similar research in this area
- Development of the new parametric method
- Comparison of the new method against current parametric methods
- Conclusions and recommendations

2 Literature Review

The following chapter includes the results of a literature review regarding the history, development and some of the weaknesses and issues associated with the parametric method as it is currently employed in industry and academia. The specific concepts that will be addressed are as follows:

- History of CAD
- Parametrics
- Master model concept
- Limitations of current parametric methodology

2.1 History of CAD

When one looks at the majority of implemented parametric methodologies and applications, they reside within modern CAD packages. More importantly, the development of CAD over the years is of particular importance to this research because early CAD packages did not have the capability of supporting high level parametric operations.

The onset of CAD systems started as electronic 2D-drafting devices in the 1960s with the Sketchpad System [2]. From these humble beginnings, CAD packages have evolved from 2D drafting tools, to wire frame modeling systems, to surface modelers and

finally into the 3D modeling and surface packages of current. Various advances in technology played major roles in this evolution, namely, numerical control of machines, surface modeling, computer graphics, finite element analysis (FEA) and general computer hardware/software improvements [3]. The development of CAD packages has resulted in many benefits for the engineering design process, such as the ability to simulate performance aspects early in the design process. Early simulations are especially beneficial, as they save time and in turn, money for design companies [4]. Additionally, improvements with computer technology has paved the way for third generation CAD systems to be better able to integrate the parametric techniques used to benchmark the methodology of this thesis.

In addition to parametric applications, these improvements have also allowed CAD systems the potential to apply the method of CAD master models (CADmm). There are however, inherent problems with the current methods of applying parametric principles to CAD parts and assembly files. The next sections will address current parametric practices, CADmm implementation as well as current limitations inherent with current parametric methodologies as applied within third generation CAD systems.

2.2 Parametrics

Parametrics have revolutionized the capabilities of CAD systems. They have opened the door for CAD systems to be programmed and allowed for the reuse of CAD models for multiple purposes [5]. The idea of parametrics can be traced back to the onset of CAD systems with the Sketchpad System that incorporated the use and implementation of graphical modification per geometric constraints [2]. From its

beginnings, parametrics has evolved to define a model "…in terms of [its] key dimensions and association between these dimensions" [6]. Proposed initially by Pratt and Wilson in conjunction with a CAM-I project investigating feature-based design [7], research in parametric methods continued to increase with the introduction of feature-based design systems in various university labs such as Minor and Gossard's system at MIT [8], Dixon's system at the University of Massachusetts [9], the First Cut system from Stanford [10], and the QTC system at Purdue [11].

Features, or any entities within the CAD model belonging to a semantic order higher than the geometric one, opened the door to parametric research. This is because features necessitate the implementation of a methodology to manage the combinatorial relations associated with the resultant models [12]. In CAD models, the creation of features is maintained within a feature tree similar to the part navigator screen shot shown in Figure 2-1. Some examples of features include a sketch, fillet, blend, revolve, extrude, slot, boss, hole, etc.

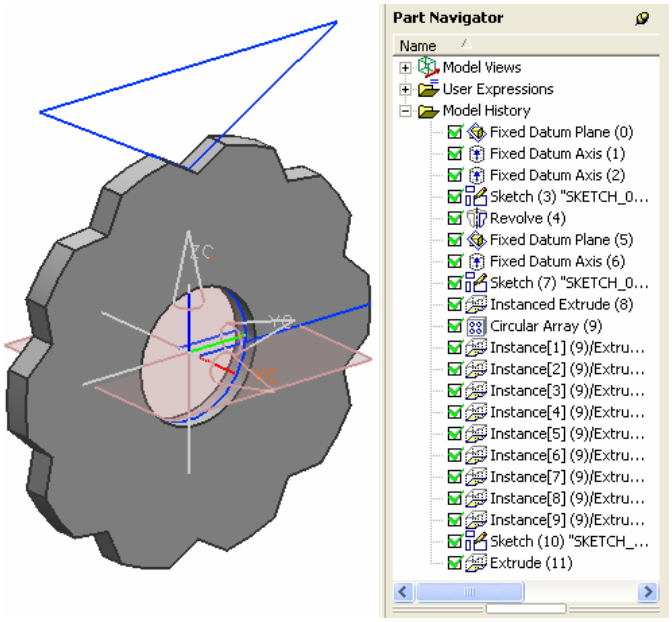

Figure 2-1 Feature tree, outlining the creation order of the features in the model.

The proper application of parametrics has the potential to reduce cycle time, improve end designs and allow for more innovation [15]. However, to fully apply parametrics to a CAD model, a definition of parametrics must first be established. In the context of engineering design, for a CAD model to be parametric it must be reusable and robust. Giullian et. al. further defined the basic criteria surrounding a reusable component to include being easy to learn, maintain and keep accurate [16].

Along with being reusable, a parametric CAD model must also be robust. Webster's defines robust as being capable of performing without failure under a wide range of conditions [17]. The degree of robustness measures the number of valid models possible versus the total number of models possible. With this definition of parametrics, it becomes clear there is more required for a CAD model to be parametric than simply naming expressions or having fully constrained part sketches. As can be seen, parametric models involve much more planning and imbedded information.

While every CAD model will be different in terms of complexity and number of parameters and features, there are certain characteristics common to all parametric models. First, parametric models will have their geometric modeling features described in terms of key parameters and their relationships. These key parameters and their relationships make up a series of key characteristics. According to Lee and Thornton, these key characteristics are "product features, manufacturing process parameters, and assembly features that significantly affect a product's performance, function and form [18]." These key characteristics define product requirements and the relationships between the various part parameters and are critical to proper parametric implementation.

Second, parametric models also contain a set of input and/or structural parameters that accurately describe the CAD model. Input parameters are variables or relationships defining key dimensions in a model, while structural parameters define the topological and hierarchical similarities between models [13]. All other parameters will be based off of these key parameters, inputs and there respective relationships and key characteristics.

Current parametric methodologies can be implemented using an interactive approach by using the native parametric capabilities imbedded within the CAD system (i.e. NX, CATIA, Pro/E) to create flexible reusable models. For major CAD applications, parameterization happens automatically in background processes, while the designers create the model [14]. These automatically generated parameters can then be manually modified and changed to incorporate engineering and design knowledge through applying the changed parameters to different definition equations (see Figure 2-2).

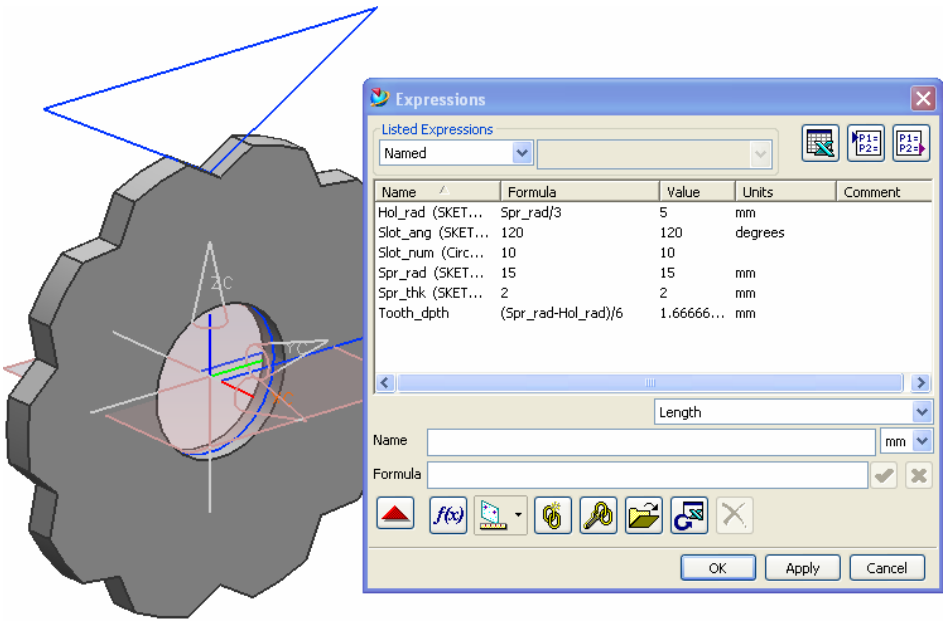

Figure 2-2 Parametric variables and equations to control the definition of a CAD model.

Even with correctly parameterized CAD models, engineering knowledge must be supplied to the model to fully utilize the reusability and understand the limits inherent with any parameterization scheme. Take for example a simple block part with a hole in the center as shown in Figure 2-3. Without proper understanding of the robustness of the parameterization scheme employed by the CAD model, the hole can easily become larger than the block feature, resulting in update errors, or no block solid.

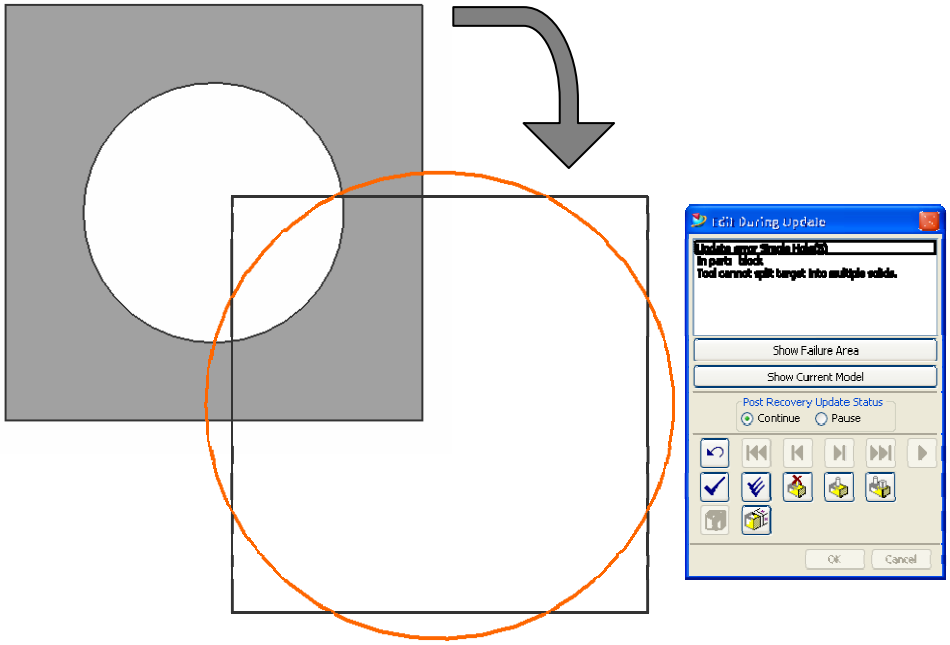

Figure 2-3 Even with a fully parameterized model, the parameters can be edited in such a way to make the resulting model no longer viable.

As can begin to be seen from Figure 2-3, part update based on parameter modifications is a definite issue associated with commercial CAD packages. This issue however, can be dealt with in various ways and will not be directly addressed by this thesis, but left for possible future work.

The onset and greater availability of parametrics has had a huge impact on industry as it relates to design and production processes. Robust parametrics has been conclusively proven to improve design time and end designs [20]. When parametrics were applied to the design of a raw materials blending yard, the design efficiency was improved more than ten times, shortening design cycle time and therefore costs [21]. By applying parametric methods early in the design process, downstream design iterations can be easily incorporated into the design. This can be done by importing new parameter values into the CAD package either manually or from a common parameter database or parameter definition spreadsheet. The CAD program can then automatically regenerate or update the model based on the new parameter values [19].

2.3 Master Model Concept

As previously stated, in attempts to keep consistency within the design process, current industry trends are showing a shift towards the use of CADmm. CADmm are intended to be used in multiple, if not all, phases of the design process, with respective clients including but not limited to, the CAD system and the various domain-specific application subsystems. These subsystems could deal with a myriad of issues such as manufacturing process planning, geometric dimensioning and tolerancing, cost estimation, performance evaluation etc [22]. Of particular importance in this area is the process of removing the segregation commonly employed between the design feature CAD model and the CAD model used for manufacture and machining process plans. By fully implementing the CADmm concept, design and manufacturing models can be one CADmm and not separate representations prone to update and consistency errors [23].

2.4 Additional CAD System Capabilities

Along with the ability to create model instantiations based on input parameters, third generation CAD packages also allow for certain scaling operations. High end CAD packages such as NX, CATIA and Pro/E all contain scaling methods allowing for both uniform and non-uniform scaling (see Figure 2-4).

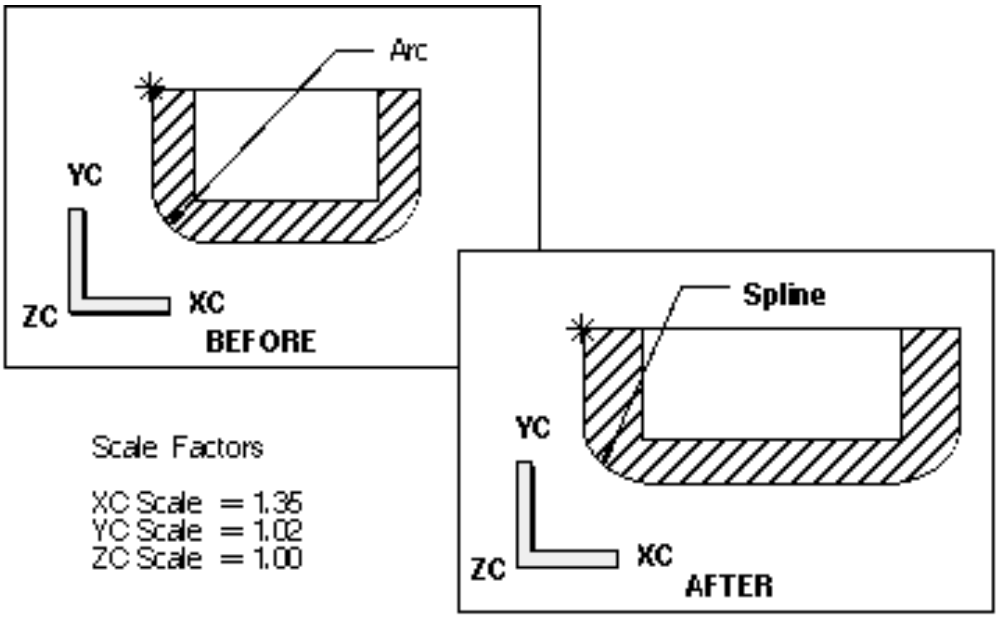

Figure 2-4 Non-uniform scaling operation applied to the various scaling factors [24].

In NX for example, you can choose to scale a part based on a uniform scale, or a non-uniform scale based on specific values in the X, Y and Z coordinate directions. Both scaling operations can then be applied via a user-specified point or plane (see Figure 2-5). For the major CAD systems, there is currently no easy way to scale assemblies other than scaling each member independently based on a common reference point and scale factors for the various parts. While the lack of an easy method of accomplishing assembly-based

scaling is an issue associated with commercial CAD packages, this issue will not be addressed by this thesis, and only part-based scaling will be discussed.

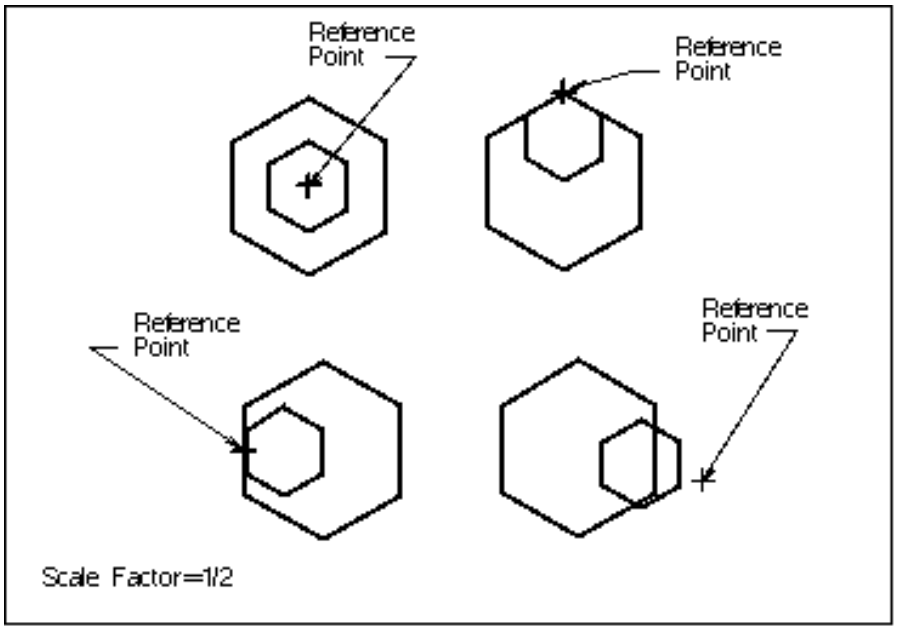

Figure 2-5 Uniform scaling operation applied via various reference points [24].

Along with having built-in operations for scaling, third generation CAD packages also have built-in functions to include tolerance information in the CAD model. Until only recently, the addition of tolerancing data was only available within CAD generated engineering drawings (see Figure 2-6). This tolerance data is included only in the drawing realm of the CAD system. The tolerance information does not appear in the expression of the CADmm.

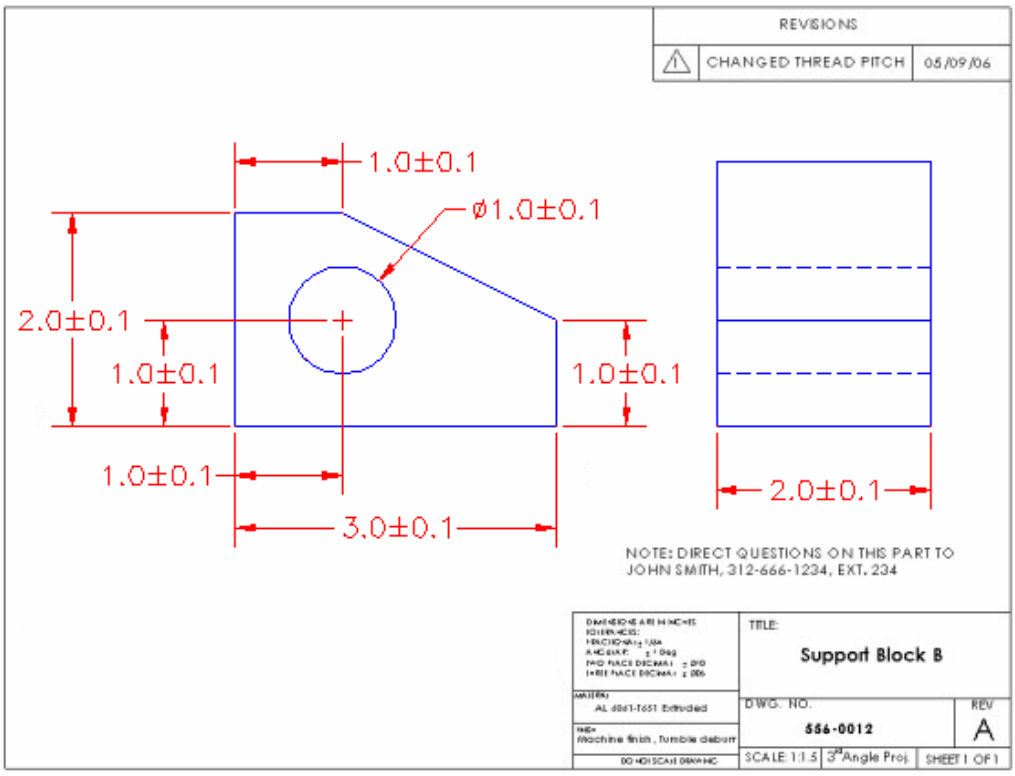

Figure 2-6 Tolerance information included on a typical engineering drawing.

Recently, the addition of model based geometric tolerancing has been introduced within commercial CAD packages. Major CAD packages are able to associate tolerance information for geometric objects via tolerance features, such as datums and geometric dimensioning and tolerancing (GD&T) symbols [25]. It is important here to differentiate between GD&T and the single value based tolerancing addressed in this research. There has been much research done in the area of GD&T, particularly tolerancing based on characteristics, such as parallelism, perpendicularity, concentricity, flatness etc. For this research, the tolerancing discussed is that associated with the individual parameters and features of the CADmm. Addressing GD&T could however be an area for future research in applying a more robust parametric scheme to this arena. As with scaling

characteristics, this thesis only addresses part-based tolerancing and not assembly-based tolerancing.

2.5 Current Parametric Methodology Limitations

The use of parametrics has resulted in great successes and improvement to the design process. There are however, inherent limits associated with the parametric techniques and methodologies currently used within commercial CAD packages. One example of limits associated with current parametric methods is the lack of tolerance information included in parametric model representations. As briefly discussed in the previous section, commercial CAD packages currently have the capability of including tolerance information by means of a +/- tolerance value inclusion to the drawing dimensions. Also, CAD systems are able to apply some forms of GD&T based on model characteristics. They cannot, however, automatically incorporate value independent tolerances to the various parameters in the CAD model. Current commercial CAD packages have no easy way to apply different value tolerances to the individual part parameters anytime within the design and manufacturing process. This lack of information can result in expensive and time intensive iterations downstream in the development process [26]. Along with not including tolerance information, current parametric methods cannot easily represent physical and in particular, exponential phenomena, such as heat transfer or exponential growth and decay [26].

Another example of limits from the current parametric process is the difficulty involved in representing products in the initial design process. Current parametric methodologies only allow the storage of single value representations to describe model

parameters. Because of this, CAD systems are not able to fully incorporate and seamlessly update for all of the engineering uncertainties inherent with the initial design process [27, 28]. Quantities such as tolerances, feature scaling values and possible exponential growth or decay factors are all examples of factors that, while not dominant in the early design phase, become crucial in later phases of the design process. This becomes particularly problematic when attempting to scale CAD models for design or manufacturing reasons. As discussed in the previous section, third generation CAD packages have some built-in scaling operations for both uniform and non-uniform scaling of CAD parts. The problem arises, however, as such operations generally un-parameterize the part, resulting in dumb solids with little or no parametric knowledge stored in the part. Figure 2-7 graphically depicts the results of completing a uniform scaling operation to a simple sprocket part in NX 4.0.

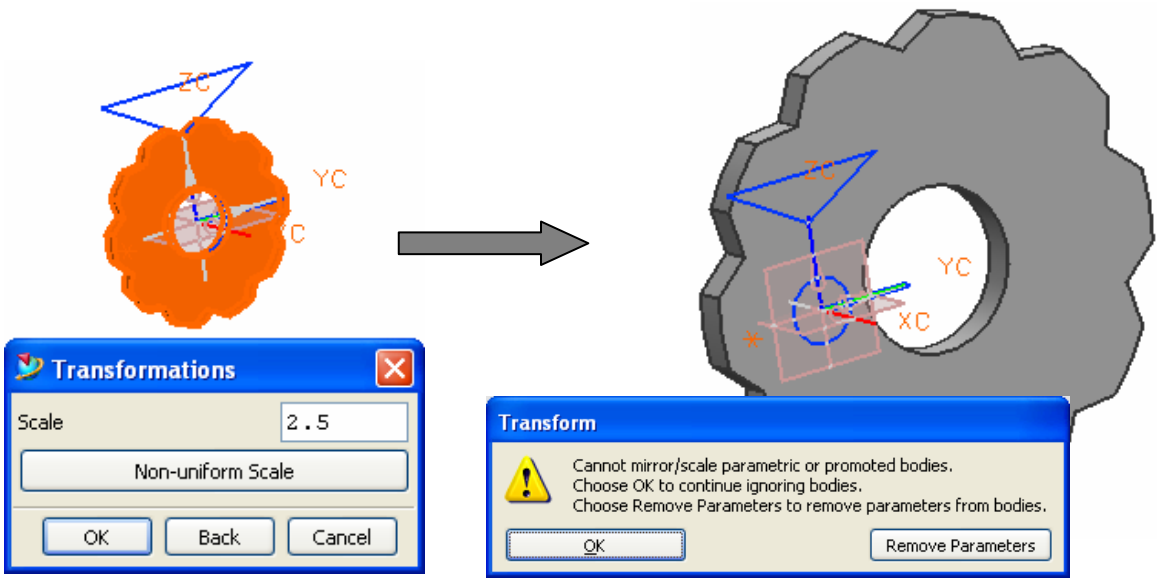

Figure 2-7 Resulting model after a uniform scaling operation is applied within NX 4.0.

Notice how the resulting model has no driving parameters, and the previous parametric model is replaced with an un-parameterized solid that cannot be edited through the parameters previously created to define the part. Along with resulting in an un-parameterized part, these methods do not easily allow for partial scaling, for example a scaling scheme applied only to certain part features or parameters.

Various methods for representing these uncertainties, such as using a probabilistic-based approach or fuzzy-set based approach, have been researched and explored [29]. While these methods have their distinct advantages and disadvantages, this thesis presents a methodology to broaden the current definition of parametrics, expanding the methodology to support and encourage multiple value representations for model parameters.

3 New Robust Parametric Methodology

Because of the multiple limitations inherent with current parametric practices, a more robust parametric method is needed to more completely realize the CAD master model (CADmm) concept as previously outlined. This new parametric method will be developed and tested in the following manner. First, the robust parameter equation will be defined. This will consist of identifying the appropriate factors included in the parameter equation. The factors will then be organized in such a way to maximize the robustness and inter-factor independence of the resultant parameter definition equation. Once a parametric scheme has been developed, the scheme will then be tested to demonstrate its robustness in handling the various parameter changes inherent with the multiple iterations and phases of the design process.

To demonstrate the parameter equation definition, a Robust Parametric Plug-In (RPPI) will be developed to be run in conjunction with NX 4.0. Through the use of the RPPI, the robustness of the parameter equation will be demonstrated when applied to two different example scenarios. For example scenarios, a jet engine turbine case and an automotive internal combustion engine block will be used. The first of these examples is an existing product with well documented design histories outlining the design and manufacturing steps currently employed by Pratt & Whitney Jet Engine Company (PW). The second example is taken from the current PACE collaborative engineering project at

Brigham Young University (BYU PACE). For the BYU PACE project, the components for a formula race car are being designed collaboratively at 20 different universities worldwide. Being such a common part, the engine block, in a similar manner to the PW example, also has well documented design and manufacturing processes that will be applied in the implementation of the new parametric methodology.

Using this history information, both examples will be taken through the various design and manufacturing steps twice. The first time will be with a model using a traditional parametric approach. The second time will be with a model incorporating the new robust parametric method. A comparison between the resulting models of the two examples per specified metrics, namely, ease of use and time to implement changes will then be made. Through this comparison, it will be demonstrated that the test CAD models employing the new robust parametric methodology can update to the various design and manufacturing changes with a higher level of accuracy than those of the original test parts provided by PW and BYU PACE. This will verify that the developed methodology is a more robust definition of parametrics.

The following sections describe the steps followed and functions created in developing and demonstrating the robust parametric schema.

3.1 Robust Parametric Equation Definition

As addressed in the previous chapter, the current parametric methodology uses single value representation formulas to represent parameter definitions. In the simple example of a sprocket, a standardized parametric definition could be employed to define the sprocket parameters as outlined in Figure 3-1 [30].

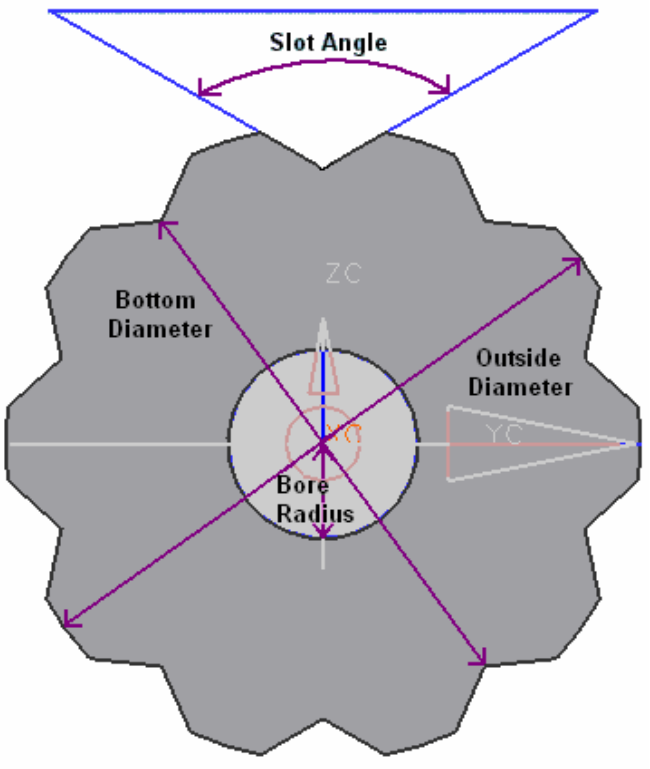

Figure 3-1 Graphical view of a simple sprocket design.

There are many standards employed in sprocket design, including radius tooth, involute spline and straight tooth forms [31]. To maintain simplicity, for this example, the straight sided serration standard is used. Additionally, the parameterization scheme employed within the NX model is shown in the expression editor shot in Figure 3-2.

As can be seen from the expression editor screen shot on the following page, the various parameters are represented as a set of single value representations, such as:

$$Hol\_rad = 4.50mm \qquad \textbf{(3-1)}$$

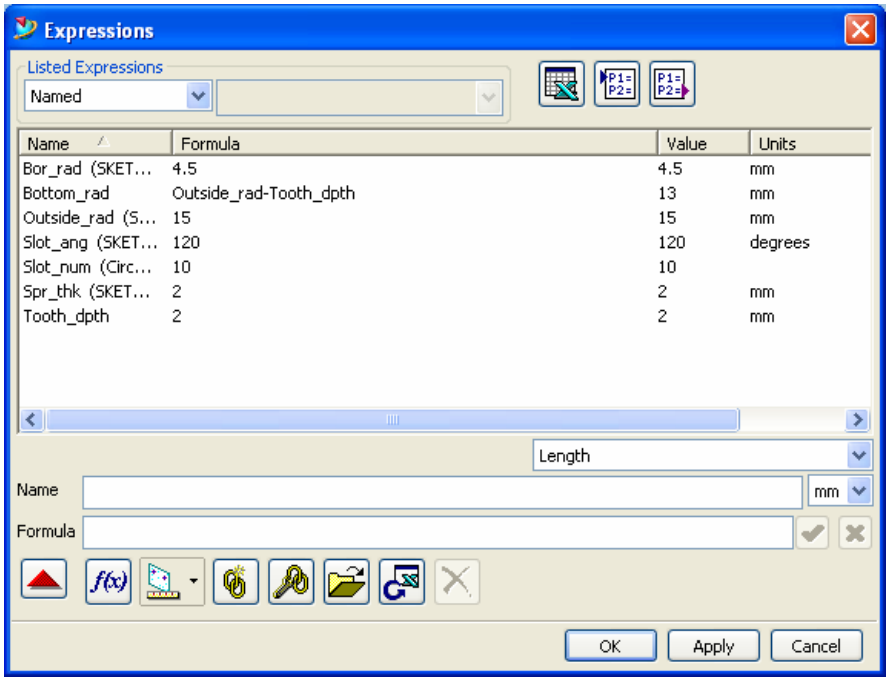

Figure 3-2: Possible parameterization scheme and values for a simple sprocket design.

While this single value representation parameter may be adequate for final print parts that no longer undergo any parameter revisions, most parts will undergo multiple parameter revisions. Through these changes, the parametric knowledge surrounding the value change would be lost using the current parametric technique. The following sections will further address the development of a more general parametric equation and focus particularly on the incorporation of the different factors within the robust parametric definition equation.

3.1.1 Scaling

Whether to create an only slightly modified product or a smaller or larger instance of a previous part, scaling of 3D CAD models is an issue that many companies deal with. As discussed in the previous chapter, certain routines exist for uniform and non-uniform

scaling of CAD parts, but generally un-parameterize the part, resulting in dumb solids with little or no parametric knowledge stored in the part. Along with resulting in an un-parameterized part, these methods do not easily allow for partial scaling, for example a scaling scheme based only on certain part features or parameters.

Returning to the simple sprocket example, if for various design constraints, this sprocket was to be scaled based on an overall scaling factor of 2.0 for the main body of the sprocket, with a factor of 1.2 for the features of the sprocket, such as the hole radius and tooth definition. Using the current parametric method would require the part parameters to be updated with new values, either manually or through the import of new values from a spreadsheet or text file. This type of update would result in a loss of the engineering knowledge relating to why the parameter was changed.

Rather than just changing values, the addition of a scaling parameter to the parameter value definition would allow for the flexibility of scaling CAD models without losing specific values concerning the scaling operation. Reorganizing the original user-specified parameter to be a part of a driving parameter definition equation that would now control the original dimension, would allow for just that. Equation 3-2 showcases how the parameter definition equation would be organized for a radius dimension. Of importance to note is how the control of the dimension is transferred from the user specified parameter to a robust equation parameter consisting of the original parameter and a scaling factor parameter.

$$R\_Radius = S\_Radius \times Radius \quad \textbf{(3-2)}$$

In Equation 3-2, the robust equation parameter is denoted by *R_Radius*, the scaling parameter is denoted by *S_Radius*, and the original parameter as specified by the user, is denoted simply by *Radius*. Furthermore, incorporating CAD model scaling by the use of a methodology like that outlined in Equation 3-2 would address the issue of different portions of the model being scaled at different ratios. Because each driving parameter would be reorganized to define the parameter value by means of an equation rather than a single value representation, implementing a scaling scheme based on certain part features or parameters is more easily realized.

3.1.2 Tolerancing

In addition to modifying models based on scaling parameters, models are often tweaked and changed based on different tolerances based on the various manufacturing processes completed on the part throughout the design cycle. In similar manner to the manual update of model scaling, tolerance values and discrepancies are often entered manually. This again loses the information regarding the accuracy and results of the various manufacturing processes.

For example, if a part radius is specified to be manufactured within a tolerance of 0.01 mm, tolerance values could be added to the parameter definition as outlined in Equation 3-3. As with incorporating the scaling parameter, notice how the control of the dimension is transferred from the user specified parameter to a robust equation parameter consisting of the original parameter and a tolerance factor parameter.

$$R\_Radius = Radius + T\_Radius \quad \textbf{(3-3)}$$

Where the tolerance value is represented by the *T_Radius* parameter. If the scaled radius parameter in the previous section also had a tolerance value associated with it, the resulting robust parameter definition equation would be as follows:

$$R\_Radius = S\_Radius \times Radius + T\_Radius\,. \qquad \textbf{(3-4)}$$

Of particular importance to note here is the independence of the scaling and tolerance parameters. As a general rule, larger parts have larger tolerance values. However, the relation between the two factors is highly process dependent and are often not related based on a consistent factor or amount. For example, doubling the diameter of a part turned on a lathe does not double the process variation. To incorporate this phenomenon into the robust parameter definition equation, an additional scaling factor can be added to the equation as follows:

$$R\_Radius = S_1\_Radius \times Radius + S_2\_Radius * T\_Radius\,. \qquad \textbf{(3-5)}$$

With the inclusion of the additional scaling factor, the designer or manufacturing engineer adjusting the CADmm will be able to apply appropriate values to account for the interactions between the scaling and tolerance values. Alternatively, if the tolerance values do not directly scale with the scaling factor, the designer can still adjust the tolerance parameter value independently of the scaling factor and set the second scaling factor to the default value of 1.0.

3.1.3 Exponential Growth and Decay

Yet another phenomenon affecting dimensions and CAD model parameterizations is that of exponential growth or decay. There are many situations where the increase or decrease of a variable over a fixed time interval will be proportional to the magnitude of the variable at the beginning of that time interval. Such examples include chemical dipping and etching processes that vary based on the amount of treatment time. These processes react based on various functions, which when included in the previous radius dimension example, is denoted by *f(t)* in the resulting parameter definition equation:

$$R\_Radius = (S_1\_Radius \times Radius + S_2\_Radius \times T\_Radius) \times f(t) \qquad \textbf{(3-6)}$$

Including exponential processes within the parameter definition equation by means of process functions allows for increased flexibility in the parametric method. This function could represent a myriad of processes, including, but not limited to power series operations, logarithmic processes or polynomial operations. For this thesis, one example of these types of processes will be explored and demonstrated to illustrate the robustness of this methodology. The exponential process used is a naturally occurring growth and decay process. This process occurs naturally in nature as well as in many manufacturing processes currently employed by industry. This particular process responds according to the following equation:

$$N = N_0 \times e^{kt} \qquad \textbf{(3-7)}$$

N is the parameter value after a time t; N_0 is the initial parameter value and k is the exponential constant (growth if positive or decay if negative). Returning once again to the radius dimension example, consider the final part being subjected to a chemical dipping process with exponential growth properties. In this case, the resulting robust parameter definition equation for the radius parameter would be as follows:

$$R\_Radius = (S_1\_Radius \times Radius + S_2\_Radius \times T\_Radius) \times e^{(k\_Radius)(t\_Radius)} \quad \textbf{(3-8)}$$

where *R_Radius*, *k_Radius*, and *t_Radius* correspond to the *N*, *k*, and t variables in Equation (3-5), respectively. Additionally:

$$N_0 = (S_1\_Radius \times Radius + S_2\_Radius \times T\_Radius). \quad \textbf{(3-9)}$$

Important to note here is the inherent interaction between the scaling factors and the *k* and *t* process parameters. While the scaling factors could have the potential to effect the exponential factors as well, this interaction is not addressed in this thesis and is left for possible future work. Because of this, it is assumed the designer or engineer adjusting the scaling factor would also understand and adjust the exponential process factors accordingly.

3.1.4 Fully Integrated Robust Parametric Equation Definition

With the simple sprocket example, after the various processing changes are incorporated manually into the model, the original design is very hard to deduce from only looking at the part parameters (see Figure 3-3).

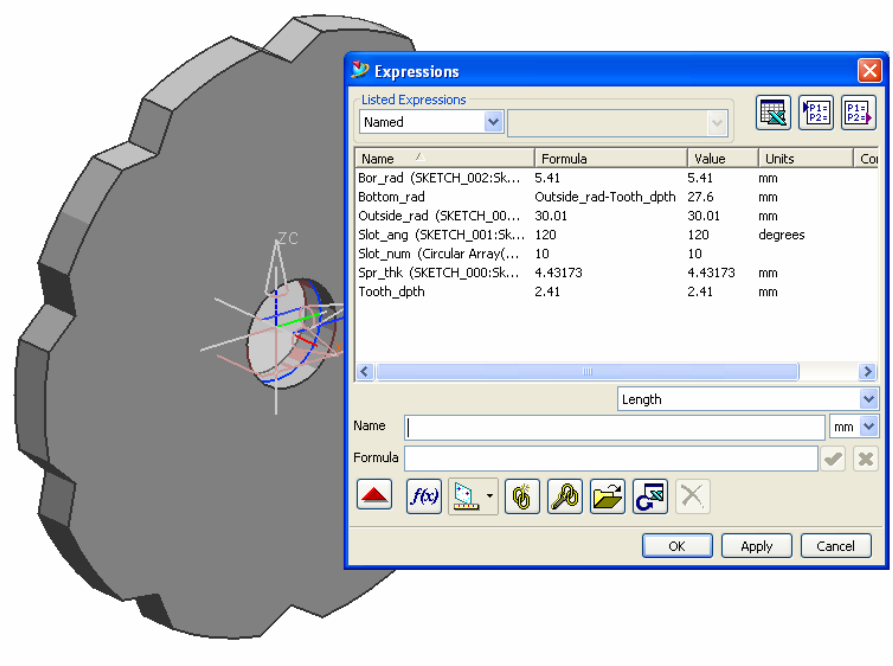

Figure 3-3 Manually updated parameters to include scaling, tolerances and exponential growth factors for the simple sprocket example.

For a simple example like this, the amount of time required to manually update the part to incorporate these changes is not an overly time-intensive process. However, the amount of time required to implement changes by hand later in the design process for more complex models quickly grows. This further reduces designers and manufacturing engineers time spent actually designing and manufacturing the respective parts.

As opposed to simply updating the parameters by hand, a more robust parametric scheme incorporated early in the design process could save update time and help preserve

the engineering knowledge and decisions stored within the CAD model. The following scheme, when applied correctly, can do just that. The Robust Parametric Master Model Scheme (RPS) consists of a multiple value representation equation. The resulting equation defines the parametric relationship between the main part driving parameters and process parameter values such as scaling factors, tolerance values and exponential growth and decay factors. This equation is defined as:

$$P_n = (s_{1n} p_0 + s_{2n} \delta_n) \times f(t) \quad \textbf{(3-10)}$$

Where the individual parameters and their relationships are defined as follows:

- P_n: new defining parametric equation
- p_0: part parameter, as entered originally by user
- s_{1n}: scaling parameter associated with the original parameter
- s_{2n}: scaling parameter associated with the tolerance perturbation
- δ_n: tolerance perturbation value
- *f(t)*: specific process to be applied to the part

As discussed in the previous section, the exponential process used to demonstrate this methodology is a natural exponential process as outlined in Equation (3-7). When the general function in Equation (3-10) is replaced with this process, the robust parameter definition equation is defined as:

$$P_n = (s_{1n} p_0 + s_{2n} \delta_n) \times e^{kt}. \quad \textbf{(3-11)}$$

This Equation is the one used to demonstrate the methodology of this thesis. The additional exponential process parameters are defined as follows (see Figure 3-4):

- k: exponential growth or decay factor
- t: time associated with exponential process

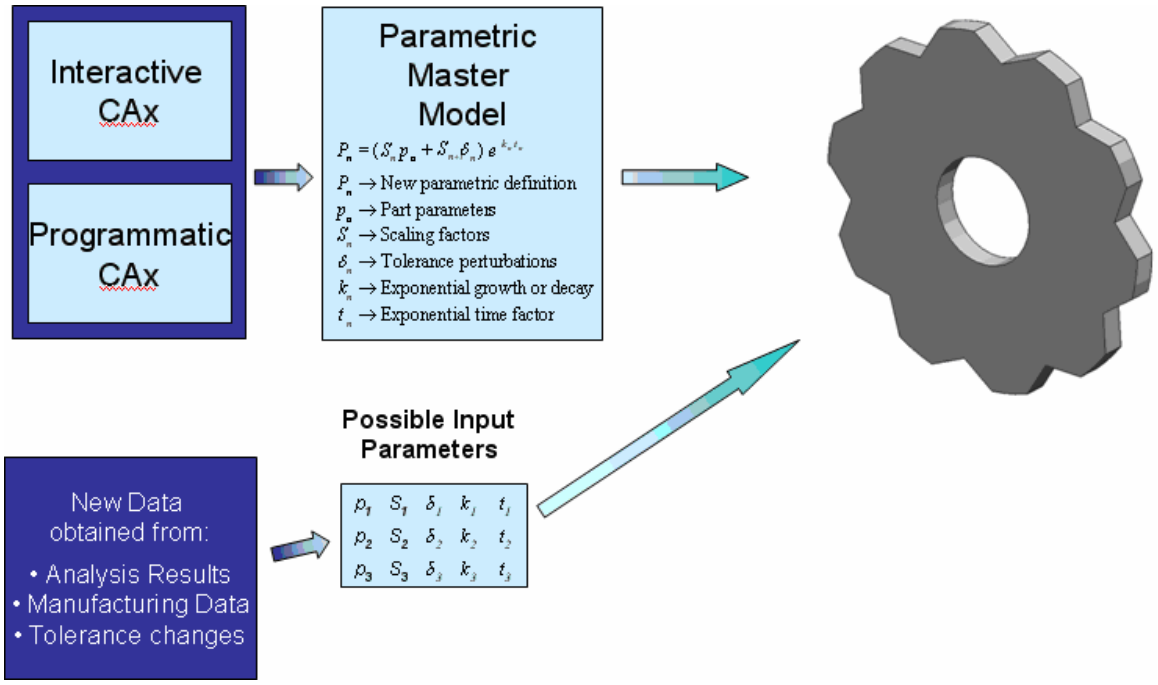

Figure 3-4 Graphical representation of the RPS outlining the parameter relationships and their flow.

Using the RPS transitions the traditional single value representation parametric scheme to a multiple value representation system, increasing the flexibility of the parametric master model. While not all engineering parts will necessarily see all of the different processes addressed by the RPS, it is still beneficial to include all of the parameters into the CAD models at the beginning of the design process. For example, some parts will only see manufacturing processes that do not directly scale and are based solely on tolerance information. In a situation like this, it may seem overkill to include

the scaling parameters to the parameter definition equation and cause unnecessary complexity to the model. Even so, in the initial stages of design the actual processes a CADmm may see are often modified and subject to change. A different department or division may choose to use the CADmm as a starting point for an entirely different product plan. This new application may need to be scaled up or down for design reasons. For these reasons, all of the different robust parameters within the RPS methodology are recommended to be included in the initial modeling process, regardless of their initial viewed applicability.

3.2 Robust Parametric Plug-In

To apply the RPS methodology to CAD models, a Robust Parametrics Plug-In (RPPI) was developed to be used in conjunction with a specific CAD package, namely NX 4.0. The RPPI allows users to automatically create driving parameters based on scaling, tolerance and exponential growth and decay factors starting from a master model, with traditional parametrics applied. More specifically, the RPPI automatically creates scaling, tolerance and exponential growth and decay parameters for all named parameters within the NX 4.0 parametric master model part.

3.2.1 Code Development

To demonstrate the RPS methodology, a RPPI was developed using a high level programming language and a third-generation CAD system applied programming interface (API). The programming language chosen for this implementation was C/C++, applied through the use of the Open C API for NX 4.0. In developing the RPPI, different algorithms were designed to redefine the parameterization scheme for each named,

driving parameter within the respective CAD model. The necessary code fragments were then organized according to their individual function, and necessary storage sequences were designed to properly modify and apply the new robust parametric definition equations. Within the RPPI, there are three main algorithms that perform perspective functions and sub routines. They are:

- create_default_RPPIexp: creates the default process factor parameters
- RPPI: organizes the robust parameter definition equations
- main: controls the entire RPPI application and initiates the program

The RPPI's controlling function, *main*, is responsible for calling the *RPPI* algorithm that controls the robust parameter definition equation creation and manages the resulting new parameters through smaller sub-function routines and by calling the *create_default_RPPIexp* algorithm. The following sections give brief descriptions of each algorithm listed above, along with explanations and examples of key elements within each algorithm. For the complete RPPI code, see Appendix A.

3.2.1.1 Create_default_RPPIexp

In this algorithm, an existing expression value and name are entered into the function. The expression information is then transformed into the resulting Robust Parameter Definition Equation (RPDE) for that parameter. Default values for the various factors are then assigned and the RPDE is created.

The following C/C++ code is taken from *create_default_RPPIexp*, showing how the code changes the current parameter from single to multiple point value representation. Comments are included with ***bold italicized font*** to explain sub-routine organization.

```
.
.
.
//Create Default RPPI expression
//Input the expression value (val) and the expression name (exp_name)

//Update the CAD model
UF_MODL_update();

//Print expression name to default scaling, tolerance, and exponential
parameters and create the new expressions
sprintf(S_new_exp,"S1_%s=1.0",exp_name);
sprintf(S_new_exp,"S2_%s=1.0",exp_name);
sprintf(T_new_exp,"T_%s=0.0",exp_name);
sprintf(k_new_exp,"k_%s=1.0",exp_name);
sprintf(tk_new_exp,"t_%s=1.0",exp_name);
UF_MODL_create_exp ( S1_new_exp );
UF_MODL_create_exp ( S2_new_exp );
UF_MODL_create_exp ( T_new_exp );
UF_MODL_create_exp ( k_new_exp );
UF_MODL_create_exp ( tk_new_exp );

//Print expression name to default robust parameter definition parameter
and store it in the new_exp variable
sprintf(new_exp,"R_%s",exp_name);

//Switch parameter dependency by use of a temp variable, making the main
driving parameter be the Robust Parameter Definition Equation, new_exp
UF_MODL_rename_exp (exp_name, new_exp);
sprintf ( temp, "%s=%lf", exp_name, val );
UF_MODL_create_exp ( temp );

//Print equation values to default robust parameter definition parameter
sprintf(new_exp,"R_%s=(S1_%s*%s+S2_%s*T_%s)*exp^(k_%s*tk_%s)",exp_name,
exp_name,exp_name,exp_name, exp_name,exp_name,exp_name);
std::cout<<"new_exp: "<<new_exp<<endl;

//Update Robust Parameter Defintion Equation parameter to complete
iteration
UF_MODL_edit_exp( new_exp );
.
.
.
```

3.2.1.2 RPPI

In this algorithm, a unique part identifier is entered into the function. The expression information for the part is then found and transformed into the resulting Robust Parameter Definition Equation for that parameter.

The following C/C++ code is taken from the *RPPI* algorithm. This code illustrates how the code identifies the current expression information for the selected part and then loops through the parameters using the *create_default_RPPIexp* algorithm to apply the RPS methodology completely to the part. Again, comments are included with ***bold italicized font*** to explain sub-routine organization.

```
.
.
.
//Create all RPPI expressions for CAD model
//Input the current CAD part identifying information (part_tag)

//Find all the expressions within the current CAD part
ask_part_exp (part_tag, num_exps, exps);


//Loop through all the expressions, applying the RPS methodology
for (i=0; i < num_exps; i++) {
      //Retrive the expressions name and value
      val = get_exp_val2 ( exps[i] );
      ask_exp_string ( exps[i], exp_string );

      //Grab just the name from the exp_string (char *) and store it in
exp_name
      j = strcspn (exp_string, str2);
      std::cout<<"j: "<<j<<endl;
      for (k=0; k<j; k++) temp[k] = exp_string[k];
      temp[j]='\0';

      //Create the Robust Parametric Definition Equation expression
      create_default_RPPIexp (val, temp, sc1, sc2, tol, exp_k, exp_t);

      }
.
.
.
```

3.2.1.3 Main

As previously mentioned, the *main* algorithm is responsible for calling the *RPPI* algorithm, hence, initializing the implementation of the RPS methodology.

The following C/C++ code is taken from *main*, outlining how the code combines together and gathers the necessary inputs for the previous algorithms. As before, comments are included with ***bold italicized font*** to explain sub-routine organization.

```
.
.
.
//Run the RPPI to apply the RPS methodology to a preexisting CAD model
//No inputs are required for this function

//Get the current part identifier (part_tag) for the RPPI algorithm
num_parts = UF_PART_ask_num_parts();

for (curr_part=0 ; curr_part < num_parts ; curr_part++ ) {
      part_tag = UF_PART_ask_nth_part( curr_part );
}

//Call the RPPI algorithm
RPPI (part_tag);
.
.
.
```

Main is then called within the *ufusr* function, where the Open C API interfaces with and calls NX 4.0, allowing the program to run and update the parameterization scheme of the respective CAD model.

3.2.2 Naming Convention

After the initial parameters are created, the RPPI reorganizes the parameters and creates a RPDE. The RPDE is based on the original named parameter and the newly created scaling, tolerance and exponential growth and decay parameters. The naming convention employed by the RPPI is defined as follows:

- Parameter_name: Original parameter name as defined by the user, i.e. Spr_thk
- Scaling parameter: S_Parameter_name, i.e. S_Spr_thk
- Tolerance perturbation value: T_Parameter_name, i.e. T_Spr_thk
- Exponential growth or decay factor: k_Parameter_name, i.e. k_Spr_thk

- Exponential time factor: tk_Parameter_name, i.e. tk_Spr_thk
- Robust parameter definition equation: R_Parameter_name, i.e. R_Spr_thk

3.2.3 Plug-In Implementation

The robust parameter definition equation allows the user to change the main driving parameter (i.e. Spr_thk) through the expression editor in the traditional manner, but also allows the flexibility and ease of including scaling, tolerance and exponential growth and decay factors. This is all accomplished without losing the original engineering knowledge concerning the driving parameter. In addition to preserving the engineering knowledge for driving parameter values, the RPS also maintains engineering and manufacturing knowledge about the scaling, tolerance and exponential growth and decay factors associated with the various analyses and manufacturing processes preformed on the part throughout the product lifecycle.

In the previous sprocket example, using the RPPI to apply the RPS methodology, the parametric equation defining the Spr_thk parameter could be defined as outlined in Equation (3-9) and shown in the expression editor screen shot in Figure 3-5:

$$R\_Spr\_thk = (S\_Spr\_thk \times Spr\_thk + T\_Spr\_thk) \times e^{k\_Spr\_thk \times tk\_Spr\_thk} \quad \textbf{(3-12)}$$

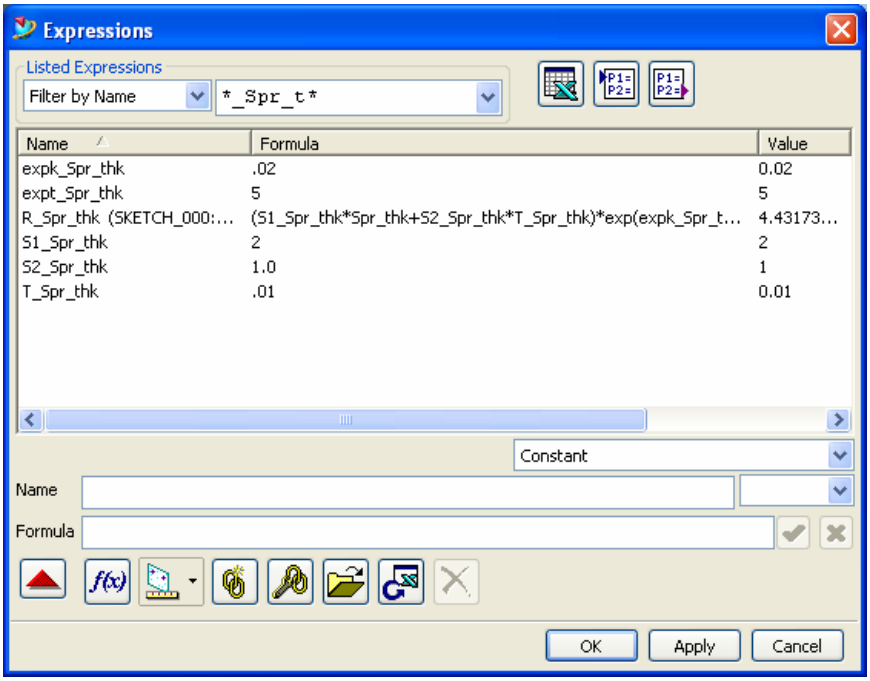

Figure 3-5 Expression editor screen shot of programmatically updated parameters transformed by RPPI.

Where Spr_thk = 2.0, S_Spr_thk = 2.0, T_Spr_thk = 0.01, k_Spr_thk = 1.0 and tk_Spr_thk = 1.0, respectively.

3.3 Test Robustness

To test the robustness of the developed parametric methodology, it is compared with current parametric practices through two different examples, an aerospace and an automotive part. Both examples are complex parts, requiring a large number of defining parameters. In addition to these examples, there are many other applications where the RPS methodology would be very beneficial to the design process. One example of these would be in the after-market segment.

Engineering companies receive large amounts of revenue in the area of rework and repair. By having CADmms with the various process information included via part

parameters, engineers could quickly and easily determine how thick the recoating process for a jet engine should be or how much material should be taken off in an etching process to repair a part.

In addition to the after market segment, there is particular application to the design side of engineering with the scaling parameters. Often, new projects will be taken on that are very similar to previous projects worked by a company. An example of this would be to take a large commercial sized engine and scale it down to a smaller engine, for a single-aisle airplane. This type of design process occurs frequently in industry and has the potential to take hundreds of man hours using current parametric methods. By applying the RPS methodology developed in this thesis, huge savings could be accomplished and end designs could be realized in a much timelier manner.

In testing the RPS methodology, the robust methodology will be deemed a more robust definition of parametrics if the test CAD models employing the new robust parametric methodology can update to the various design and manufacturing changes with an equal or higher level of accuracy than those of the original test parts provided by PW and BYU PACE. To verify this accuracy, the resultant models will be rated based on two different metrics. The first will be a Boolean metric, based on whether or not the parametric change is straightforward to implement or not. This metric, referred to as the ease of use metric, will be rated by the following dimension:

- -1: No easy way to implement the parametric change, a lot of outside user work and manipulation is required to implement these changes.
- 0: No easy way to implement the parametric change, but little outside user work and manipulation is required to implement these changes.

- 1: Easy way to implement the parametric change, requires little to no outside user work and manipulation.

The second metric will consider the time required to implement the parametric change into the various test case models, represented in minutes and seconds.

With the two different metrics, it will be demonstrated that the test CAD models employing the new robust parametric methodology perform better than the baseline current parametric models.

4 Results and Discussion of Results

This chapter showcases the results of the application of the Robust Parameter Master Model Scheme (RPS) methodology outlined in Chapter 3 on two separate examples. In both examples, the RPS methodology was demonstrated by comparing two CAD model representations of each example, one showcasing the new robust parameter equation methodology and the other employing traditional, single value representation parametrics. The two different models were then manipulated to incorporate the various design and manufacturing operations consistent with an example product process plan. The results of how each model preformed against one other, as well as specific descriptions of each example follow.

4.1 Example 1

The first example part used to demonstrate the RPS methodology is a jet engine turbine case. In particular, it is a high pressure turbine case provided by Pratt and Whitney jet engine company (PW).

4.1.1 Example Description

Traditionally, the turbine is a later stage of a jet engine. In the example part this is no exception as the turbine case used follows directly behind the combustor portion of the jet engine (see Figure 4-1).

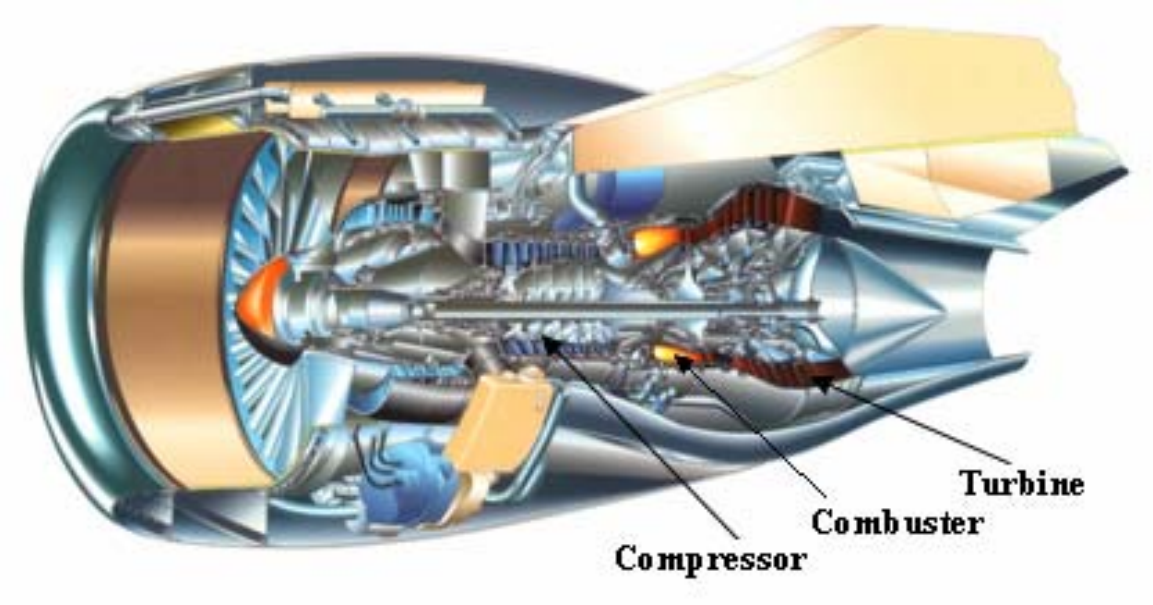

Figure 4-1 A cut out PW6000 jet engine featuring the turbine location.

The turbine is made up of bladed discs that gain energy from the hot gases leaving the combustor. As with all cyclic heat engines, a higher combustion temperature means greater downstream engine efficiency. The limiting factor, however, is the ability of the engine parts to withstand the extra heat and pressure. Because of the nature of the heat encountered in the turbine stage of a jet engine, turbine cases are subject to very high temperatures that require special manufacturing considerations (see Figure 4-2).

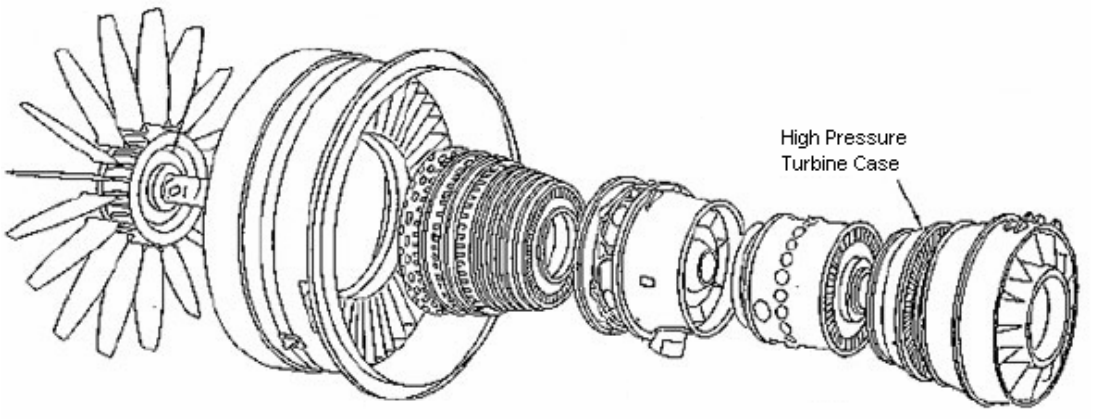

Figure 4-2 Screen shot showcasing a high pressure turbine case.

In the design of the turbine case, the primary controlled parameters deal with the case thickness as it changes from front to the aft of the turbine section. In this example, the beginning and ending axial positions of the turbine case are set based on the particular engine configuration and are not varied during the demonstration of the RPS methodology (see Figure 4-3). Additionally, Figure 4-4 shows a screen shot of the actual turbine case CAD model used to demonstrate the RPS methodology. It represents the cross section of the case, as viewed in an axial plane through the center of the engine. The vertical projections shown in Figure 4-4 are stiffener rings.

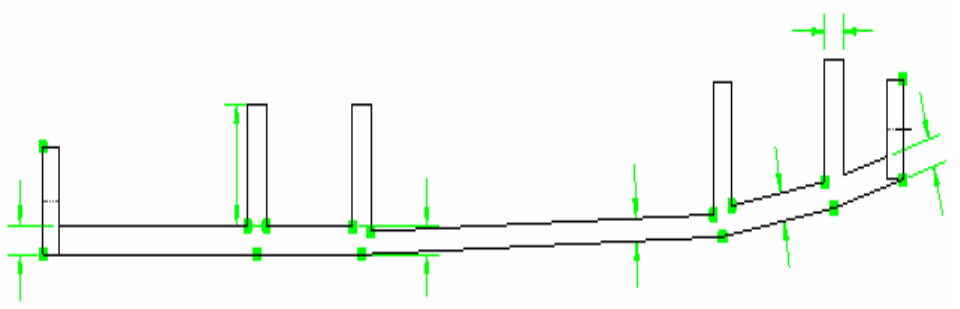

Figure 4-3 Turbine case cross section view showcasing the key dimensions controlled by the parametrics of the CAD model.

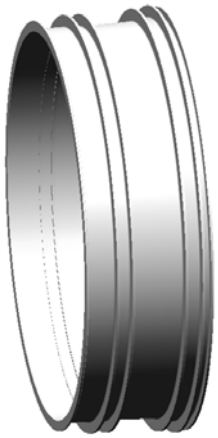

Figure 4-4 Jet engine turbine case study model used to demonstrate the RPS methodology.

4.1.2 Product Process Plan

The extreme heat conditions encountered by the turbine case require special manufacturing considerations to be taken. To deal with these conditions, different materials are chosen and special insulating coatings and machining processes that have been developed and implemented on the turbine case. Table 1, outlines the ten different processes that were applied to the two different CAD models in the demonstration of the RPS methodology against current parametric techniques.

Table 1 Basic machining processes completed in the manufacture of a high pressure turbine case.

		Applicable Factors		
		Scaling	Tolerances	Exponential
1	Semi-Finish Turn Outer Rails	N	Y	N
2	Semi-Finish Turn Inner Rails	N	Y	N
3	Finish Turn Front Rails	N	Y	N
4	Finish Turn Rear Rails	N	Y	N
5	Coating Process	N	Y	Y
6	Debur Front Flanges	N	Y	N
7	Debur Rear Flanges	N	Y	N
8	Debur Outer Hooks	N	Y	N
9	Debur Inner Hooks	N	Y	N
10	Final Coating Process	N	Y	Y

Also shown in Table 1 are the applicable parametric factors associated with the various processes. As you can see, all of the manufacturing processes incorporate tolerance factors and two of the processes integrate exponential factors for the turbine case.

4.1.3 Results of RPS Example 1 vs. Control Model

In demonstrating the RPS methodology, the turbine case model was updated by means of the RPPI to incorporate the robust parameter definition equation methodology to its parametrics. This new model was then compared on the ability to easily incorporate the desired parameter change as well as the actual time taken to implement the change on the various models. Due to the proprietary nature of the turbine case provided by PW, actual values for the various processes will not be explicitly stated. Rather, the different models will only be compared to demonstrate the robustness of the parametric method established in this thesis. Table 2 shows the results of the ease of use comparison metric for the two different parametric methodologies for the turbine case example.

Table 2 Results of the ease of use comparison metric for the turbine case.

		Ease of Use Comparison		
		Factors	Metric Value	
			RPS Model	Control Model
1	Semi-Finish Turn Outer Rails	T	1	0
2	Semi-Finish Turn Inner Rails	T	1	0
3	Finish Turn Front Rails	T	1	0
4	Finish Turn Rear Rails	T	1	0
5	Coating Process	T and E	1	-1
6	Debur Front Flanges	T	1	0
7	Debur Rear Flanges	T	1	0
8	Debur Outer Hooks	T	1	0
9	Debur Inner Hooks	T	1	0
10	Final Coating Process	T and E	1	-1
		Total	***10***	***-2***

Based solely on the ease of use metric, the RPS methodology outscores the traditional parametric method approach. This demonstrates the RPS methodology's robustness and superiority on the ease of use level.

When comparing the amount of time required implementing the different parameter changes, similar results were found (see Table 3).

Table 3 Results of the time required to implement parametric changes for the turbine case.

		Time to Implement Comparison (sec)			
		Factors	RPS Model	Control Model	Time Difference
1	Semi-Finish Turn Outer Rails	T	8	42	*-34*
2	Semi-Finish Turn Inner Rails	T	6	25	*-19*
3	Finish Turn Front Rails	T	7	34	*-27*
4	Finish Turn Rear Rails	T	5	26	*-21*
5	Coating Process	T and E	26	92	*-66*
6	Debur Front Flanges	T	9	20	*-11*
7	Debur Rear Flanges	T	7	18	*-11*
8	Debur Outer Hooks	T	9	35	*-26*
9	Debur Inner Hooks	T	8	20	*-12*
10	Final Coating Process	T and E	19	108	*-89*
				Total	**-316 s**

These times were recorded based on an experienced designer with knowledge of the parametric workings and schemes employed by both the RPS model and the control model for the turbine case example. As can be see from the results outlined in Table 3, the time required to implement the various parametric changes is much less for the RPS methodology than for the traditional parametric approach. Specifically, when implementing changes based on exponential processes the time saved by using the RPS model is on the order of a five time speed up. As can be imagined, this potential time savings becomes particularly significant when multiple operations and tolerances are applied to different CAD models on a regular basis.

Perhaps of more importance than the ease of use and time saved with the RPS method is the amount of information stored in the CAD model at the end of the ten different operations. The control model employs traditional single value representation parametrics as shown in Figure 4-5.

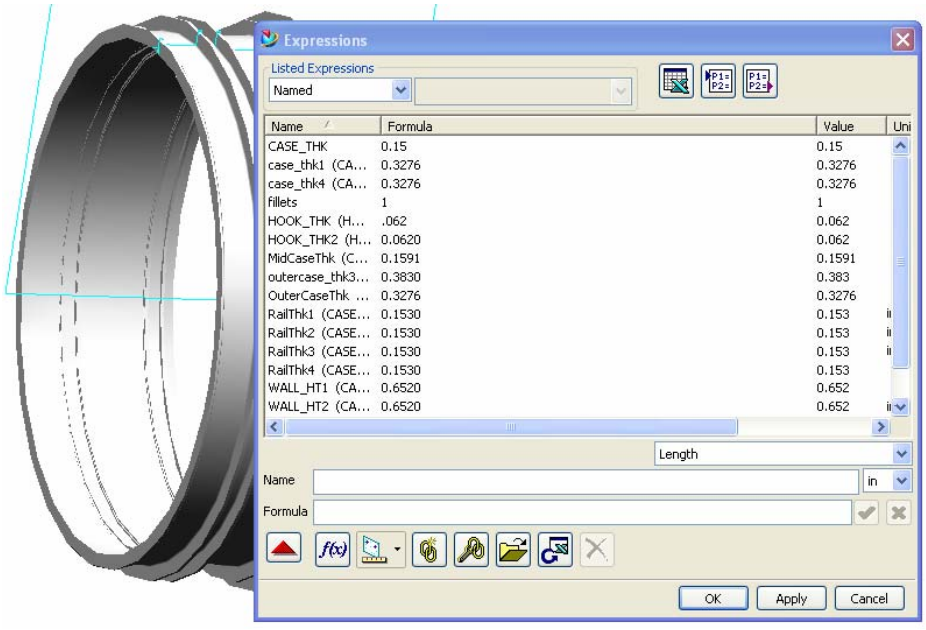

Figure 4-5 Screenshot of control model expression editor showcasing single value representation.

Alternatively, the RPS model shown in Figure 4-6 has RPDE made up of the various aspects including the actual tolerance perturbations experienced from the different processes and the specific exponential factors and times of exposure that brought the CAD model to its final state. Not shown in Figure 4-6 are the other individual parameters, such as the tolerance value applied to the MidCaseThk. These expressions could be easily found within the expression editor by scrolling through the editor interactively. In addition, the expression editor allows searching and filtering of viewed expressions based on user specifications, as shown in Figure 4-7.

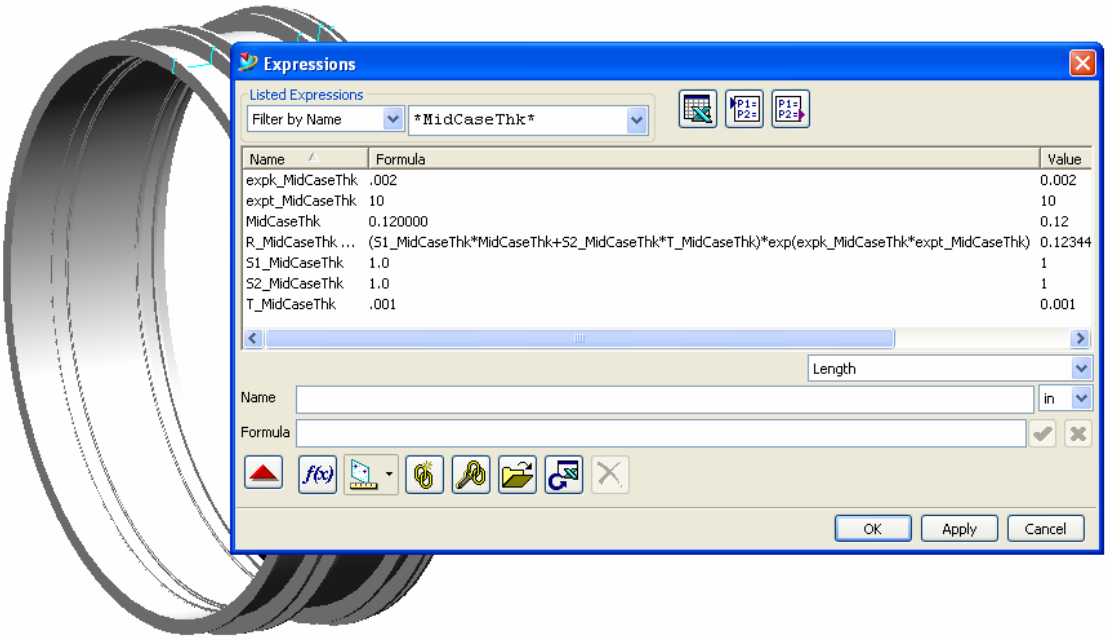

Figure 4-6 Screenshot of expression editor for the RPS model showcasing the multiple value representation.

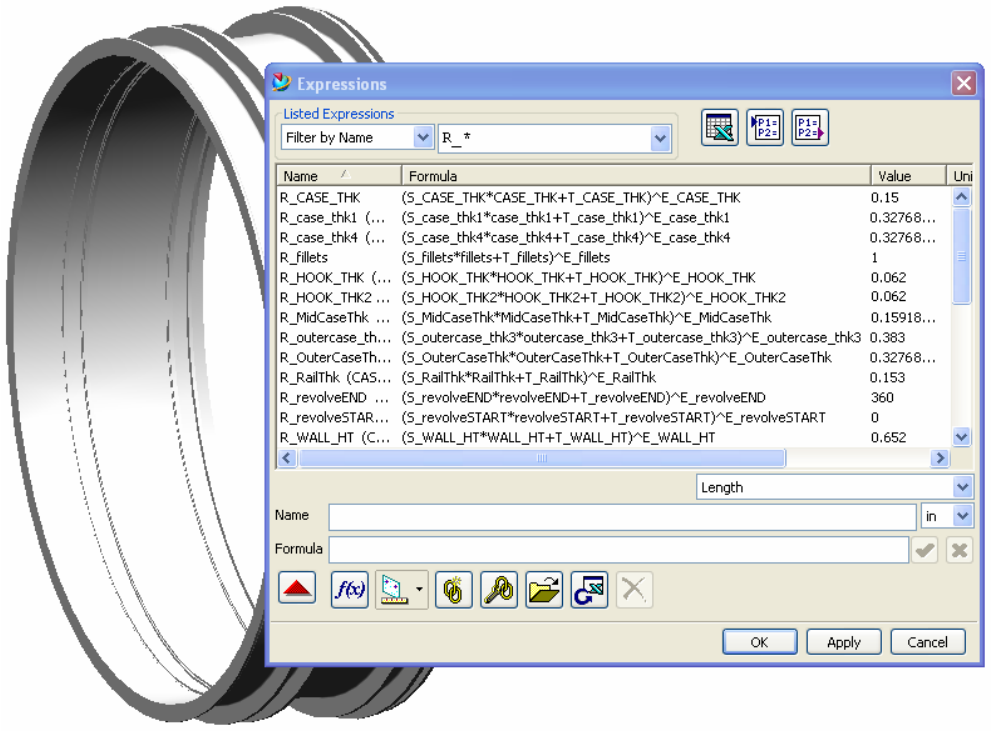

Figure 4-7 Screenshot of expression editor for the RPS model showcasing the expression filtering option for the parameter MidCaseThk.

The above figures show that the RPS model contains much more information regarding the processes and tolerances applied to the different parameters. Say for example an engineer was interested in how the overall case thickness had changed throughout the different processes applied to the case. Looking through the scaling, tolerance and exponential parameters of the RPS model would quickly show what factors played a role throughout the different processes applied. Alternatively, an engineer could also easily determine original design intent concerning a specific dimension prior to any tolerances being added or manufacturing processes preformed. This information could be very beneficial in doing backwards analysis as well as determining how dimensions change from the various manufacturing processes preformed on the part and how the resulting changes could affect overall product performance. Contrast these opportunities with that of looking at the single value representations of the control model. It would be very difficult, if not impossible to easily deduce these types of information for various parameters using this parameterization scheme.

4.2 Example 2

The second example used to demonstrate the new parametric method is an engine block for an automotive internal combustion engine. Figure 4-8 illustrates the complexity of a typical engine block. This example is based on the engine block used for the BYU PACE international collaborative design project.

Figure 4-8 Example picture of a 4-cylinder engine block.

4.2.1 Example Description

The engine block is a machined casting consisting of machined cylinders for the pistons of the engine to operate and other machined mating surfaces. The engine block model used for this demonstration is a 4-cylinder engine block. For design considerations of the BYU PACE collaborative project, this model is to be scaled up to a larger 6-cylinder size engine block. Because of this step, the engine block model acts as an interesting example in demonstrating the RPS methodology.

In many cases, the engine block is a very complicated part with multiple additions and adaptations allowing various parts, such as the crankcase, coolant passages, engine mounts, etc, to be incorporated into the engine design. For the engine block CAD model supplied by the BYU PACE team, this was no exception, with multiple complex surfaces and features all throughout the model (see Figure 4-9).

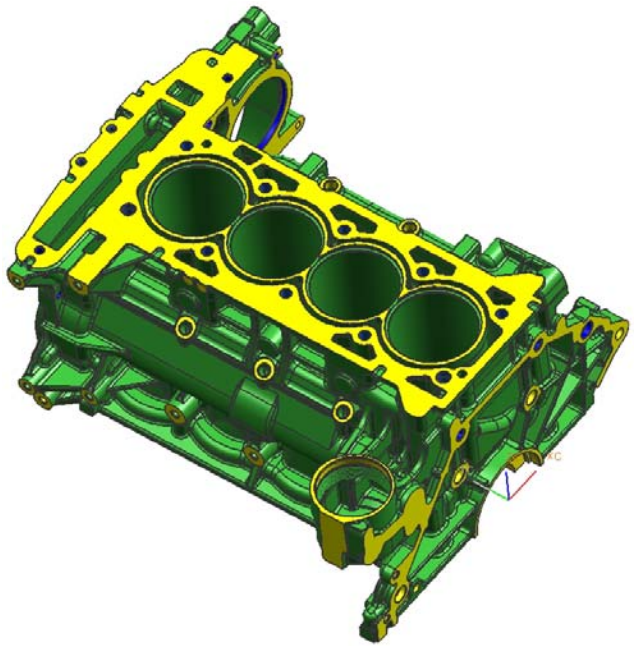

Figure 4-9 Actual engine block part model as supplied by the BYU PACE team.

To make the parameterization of the engine block model more straight forward, a simplified CAD model was created incorporating the original design intent. For the simplified engine block test part, the following eight independent main driving parameters make up the parameterization scheme (see Figure 4-10 for a visual parameter representation):

- CYL_count: Controls the number of cylinders.
- CYL_depth: Controls the depth of the cylinder holes and height of the connecting rod clearance cavity.
- CYL_rad: Controls the cylinders radius size.
- CYL_tol: Controls the distance between the cylinders.
- EB_height: Controls the height of the engine block.
- EB_length: Controls the length of the engine block.

- EB_width: Controls the lower width of the engine block.
- EB_width2: Controls the upper width of the engine block.

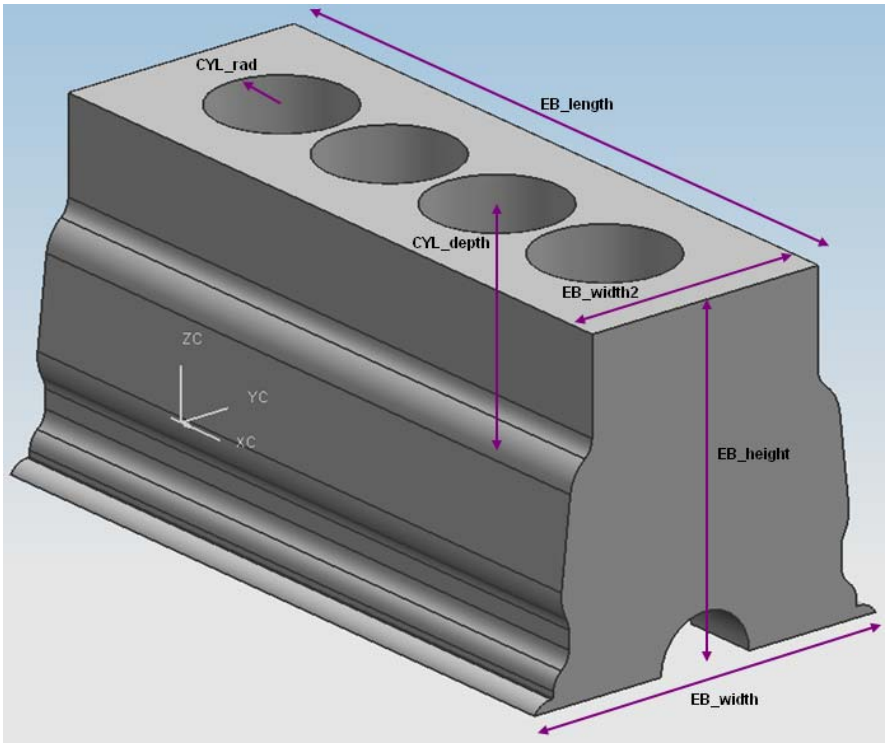

Figure 4-10 Graphical view of the engine block parameterization scheme.

4.2.2 Product Process Plan

Manufacturing an engine block consists of first, casting the main shape and then, machining and performing various exponential processes on the casting to arrive at the final desired material specifications and product dimensions. Table 4 outlines ten different processes that were applied to the two different engine block CAD models to demonstrate the validity of the RPS methodology against current parametric techniques.

Table 4 Basic machining processes completed in the manufacture of an engine block.

		Applicable Factors		
		Scaling	Tolerances	Exponential
1	Scale engine from 4 cylinders to 6 cylinders	Y	N	N
2	Cast part	N/A	N/A	N/A
3	Rough Mill cylinders	N	Y	N
4	Finish Mill cylinders	N	Y	N
5	Carburize cylinders	N	N	Y
6	Rough Bore cylinders	N	Y	N
7	Finish Bore cylinders	N	Y	N
8	Etching process	N	N	Y
9	Anodizing process	N	N	Y
10	Final Coating Process	N	N	Y

As with the previous case study, tolerances continue to play a major role in the manufacturing of the engine block. More than the previous example, however, exponential processes play a role in the manufacturing process. Also, unlike the first case, the engine CAD model will first be scaled up from a 4-cylinder to a larger 6-cylinder size engine. This additional step will allow for a demonstration of the scaling principle incorporated into the RPS methodology, as applied to a non-uniform scaling operation.

4.2.3 Results of RPS Example vs. Control Model

In demonstrating the RPS methodology, the engine block model was also updated in a similar manner to that of the turbine case. The developed RPPI was used to incorporate the robust parameter definition equation methodology to its parametrics. The new RPS model was then compared by the same metrics through the implementation of the ten different design and manufacturing operations specified in Table 4. As with the proprietary nature of the turbine case provided by PW, actual part dimensions and process values will not be used for the engine block. Instead, dummy dimension values

will be used to dictate the changes undergone through the different manufacturing processes.

In applying the different processes to the CAD model, the various processes were researched and general process data was used to approximate the actual machining processes undergone by typical engine blocks. As with the proprietary nature surrounding the actual part dimension values, contrived values were used to verify the RPS methodology. They were also generalized and not exact, based on industry standards. The values used are, however, true to function and react in a manner true to the actual processes. For the manufacturing processes applied on the engine block test part outlined in Table 4, the contrived values applied to the different processes are outlined in Table 5.

Table 5 Non-proprietary values used in applying the various manufacturing processes to the engine block test case.

		Scaling	Tolerance	Exponential
1	Scale engine from 4 cylinders to 6 cylinders	1.5 for applicable features	N/A	N/A
2	Cast part	N/A	N/A	N/A
3	Rough Mill cylinders	N/A	0.005	N/A
4	Finish Mill cylinders	N/A	0.001	N/A
5	Carburize cylinders	N/A	N/A	k: 0.001 , t: 10
6	Rough Bore cylinders	N/A	0.005	N/A
7	Finish Bore cylinders	N/A	0.001	N/A
8	Etching process	N/A	N/A	k: 0.005 , t: 15
9	Anodizing process	N/A	N/A	k: 0.0025 , t: 30
10	Final Coating Process	N/A	N/A	k: 0.004 , t: 5

In applying the different manufacturing processes to the engine block part, the ease of use comparison metric was again tabulated to compare the different parameterization methodologies based on usability as outlined on the following page in Table 6.

Table 6 Results of the ease of use comparison metric for the turbine case.

		Ease of Use Comparison		
		Factors	Metric Value	
			RPS Model	Control Model
1	Scale engine from 4 cylinders to 6 cylinders	S	1	-1
2	Cast part	N/A	---	---
3	Rough Mill cylinders	T	1	0
4	Finish Mill cylinders	T	1	0
5	Carburize cylinders	E	1	-1
6	Rough Bore cylinders	T	1	-1
7	Finish Bore cylinders	T	1	-1
8	Etching process	E	1	-1
9	Anodizing process	E	1	-1
10	Final Coating Process	E	1	-1
		Total	***9***	***-7***

As with the aerospace example, the RPS methodology definitely outscores the traditional parametric method approach in the arena of the ease of use metric, further demonstrating the RPS methodology's robustness and superiority on this level.

Before the times required to implement the parameter changes for the different models can be compared, a more detailed explanation of the scaling process must be defined. As graphically represented in Figure 4-10, the CAD model used for the engine block case study is driven by eight different independent parameters. To maintain consistency and simplicity, the scaled up 6 cylinder model would remain of an inline configuration similar to the 4 cylinder model. Also, the actual cylinder radius size would not scale larger, but would remain constant. However, the number of cylinders would change thus increasing the length and the height of the engine block. This scaling operation is very simplified compared to scaling an actual engine block. The operation was simplified however to make it possible to implement by hand. Even with its inherent simplicity, the operation does, however, show the potential of the RPS methodology to

more quickly and efficiently scale parametric CAD models with a more sophisticated method and reasoning behind the scaling process.

With these considerations in mind, results were found for the time required to implement the various parameter changes on the engine block CAD models (see Table 7).

Table 7 Results of the time required to implement parametric changes for the turbine case.

		Time to Implement Comparison (sec)			
		Factors	RPS Model	Control Model	Time Difference
1	Scale engine from 4 cylinders to 6 cylinders	S	27	95	*-68*
2	Cast part	N/A	---	---	---
3	Rough Mill cylinders	T	25	41	*-16*
4	Finish Mill cylinders	T	23	39	*-16*
5	Carburize cylinders	E	41	96	*-55*
6	Rough Bore cylinders	T	20	27	*-7*
7	Finish Bore cylinders	T	21	30	*-9*
8	Etching process	E	10	28	*-18*
9	Anodizing process	E	53	153	*-100*
10	Final Coating Process	E	45	123	*-78*
				Total	***-367 s***

As with the first case study, the implantation times for the engine block example were recorded by an experienced designer, with knowledge of the parametric workings and schemes employed by both the RPS model and the control model. Even more so than in the previous case study, the results outlined in Table 7 show that the time required to implement the various parametric changes is much less for the RPS methodology than for the traditional parametric approach. Important to note here is the simplicity involved in the scaling of this case study may downplay the potential time savings of using the RPS methodology for scaling applications. Even so, the over three time speed up achieved here is significant when scaling highly complex and detailed parts with hundreds of driving parameters taking weeks to scale accurately.

The engine block example definitely showcases the potential benefit of the RPS methodology in being able to quickly and accurately scale CAD models based on specific scaling factors (see Figure 4-11). As previously mentioned, these times were gathered under the assumption that an experienced designer, with knowledge of the parametric workings of the model was implementing the changes. This assumption results in much more favorable time results for the control model, which implements the traditional, single value representation parametric approach. If, as is often the case in industry, an engineer or designer was tasked to scale a CAD model they have little or no experience with, these times could potentially be much larger, further demonstrating the superiority of the RPS methodology when compared to the traditional single value representation approach.

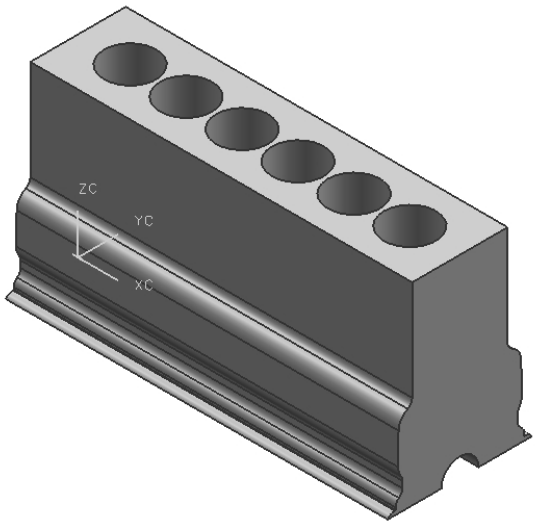

Figure 4-11 Scaled CAD model for the engine block case study.

Finally, as with the previous case study, the amount of engineering and manufacturing knowledge stored within the CAD model at the end of the ten different operations is much greater for the model employing the RPS methodology. Take for

example the parameter controlling the length of the engine block (EB_length), for the control model, at the conclusion of all of the operations, the parameter is defined as:

$$EB\_length = 700.5 \quad \textbf{(4-1)}$$

This definition gives no background information or insight to the engineering and manufacturing based decisions playing roles in the end parameter value. Compare this with the RPDE of the length parameter in the model incorporating the RPS methodology:

$$R\_EB\_length = (S\_EB\_length \times EB\_length + T\_EB\_length) \times e^{(k\_EB\_length)(tk\_EB\_length)} \quad \textbf{(4-2)}$$

This definition gives much needed insight into how the parameter changed and where it began. It also allows the designer to look at the various scaling, tolerance and exponential factors to determine exactly how the parameter evolved. Very easily a designer could find that the length of the engine block was scaled by a factor of 1.5 and various exponential growth factors with varying tolerances were applied to arrive at the final dimension value of 700.5. As previously mentioned, the scaling operation for this demonstration was simplified, assuming a uniform scale in the length of the engine block. A more rigorous method of scaling would scale the length by a ratio that increased the length only by two times the distance between the centers of two cylinders. Alternatively, the designer could also easily determine what the engine block length was before the various processes and tolerances were incorporated into the model by looking at the

EB_length parameter. In the RPS methodology, this parameter would not have been altered from the original value, or 466.7 in this example.

5 Conclusion

This thesis has developed a new parametric scheme to be used for CAD master models (CADmm) and has compared this new method with traditional parametric methods. Based on the metrics laid out in this thesis, namely ease of use and overall time required implementing various parametric changes, it has been shown that the proper application of the Robust Parametric Master Model Scheme (RPS) provide the following:

- More complete incorporation of the CADmm concept within the current design process.
- More effective and versatile incorporation of parametric changes based on scaling operations.
- Tolerancing applied to the CADmm on a feature-by-feature basis.
- Exponential growth or decay processes incorporated into CADmm.

The results discussed in the previous chapter show how the RPS method allows users to easily incorporate parametric changes based on scaling operations. Current CAD packages allow for only uniform or non-uniform scaling, resulting in non-parameterized bodies that can no longer be altered or modified by the previous driving model parameters. Beyond the built-in CAD system un-parameterizing scaling method, using traditional parametric methods make scaling CAD models an awkward and

time-consuming process. Contrast this with the RPS method of scaling parts. Using the RPS method allows CAD models to be easily scaled simply by changing the scaling parameters for the applicable features and parameters. No longer do CAD models have to be scaled by a method where the designer only has control of the magnitude of the scaling factors in the various coordinate directions. Instead, designers can scale models accurately based on feature specific scaling factors, and have the resultant scaled part still maintain the parameterization scheme and flexibility it did prior to the scaling process.

In addition to being able to better incorporate scaling abilities to CAD models, this thesis also showed how using the RPS method better allows for tolerancing to be applied to the CAD model on a feature-by-feature basis. As opposed to current CAD system and parametric method abilities, the RPS method allows case specific tolerance values to be applied to different parameters in the design space, and not just as a side note on the engineering drawing.

Finally, the results from this thesis have shown that the RPS method allows for much more design flexibility when it comes to incorporating exponential growth or decay processes into a design. As it currently sits, there is no out of the box method for applying this data into the CAD model, other than manually calculating and implementing the changes inherent with each respective process by hand. This tedious method is prone to human error which is unacceptable when compared with the ease of including the different process information associated with specific CAD models with the RPS method.

All of these factors surrounding the RPS methodology result in greater model consistency. This will reduce the confusion and compatibility issues involved with passing different models to and from disciplines through different development phases.

Proper application of this method also has the potential to have significant advantages for design and process integration.

Additionally, by programmatically applying the multiple value representation RPS to current CADmm via the Robust Parametrics Plug-In (RPPI), designers will be better able to plan for and incorporate the various uncertainties and changes inherent with the design process in the creation of CADmm. Proper application of the RPS will also reduce the amount of time spent downstream by designers, analysts and manufacturing engineers reparameterizing and updating the CADmm to fit and apply new product specifications.

Along with being able to easily implement changes to CAD models based on scaling, tolerance or exponential factors, proper application of the RPS also increases the amount of design, analysis and manufacturing knowledge automatically stored within the CADmm. Ideally, the RPS would be implemented within CAD packages at a system level, allowing for proper parameter information storage and update to be handled by the CAD operating system. This CAD system level incorporation of the RPS will allow users to automatically and seamlessly create robust driving parameters based on factors of scaling, tolerance and exponential growth and decay factors and change their respective values as applicable. Also, if the RPS methodology is directly incorporated within commercial CAD systems, proper application of the method will be much easier and the time spent updating CADmm and editing driving parameters could be reduced even further.

This research opens the door to multiple possibilities in the area of future work. First, a small scale implementation on actual production parts would provide valuable

insight into how the RPS method compares with current parametric techniques. Also, a great deal of research could be conducted in terms of implementing the methodology within a commercial CAD system. Within the implementation realm, possible additions could be included to the methodology such as, error checking, undo possibilities, security and access privileges, as well as automatically compiling or saving back-up or editing files. Yet another area in which significant future work could be conducted would be in applying the RPS methodology to other applications such as Computer Aided Engineering, Computer Aided Machining and inspection processes.

References

[1] Taylor, D. L. (1992). *Computer-Aided Design*. Addison-Wesley.

[2] Sutherland, I. (1963). "Sketchpad, a man-machine graphical communication system." PhD thesis, Massachusetts Institute of Technology.

[3] Requicha, A. G. (1980). "Representation for rigid solids: Theory, methods and systems." *ACM Computing Surveys*, 12(4), pp. 427-464.

[4] Onwubiko, C., (1989). *Foundations of Computer-Aided Design*. West.

[5] Lund, J. G. (2006). "The storage of parametric data in product lifecycle management systems." Masters thesis, Brigham Young University.

[6] Shah, J. J., Sen, S., Ghosh, S. (1991). "An Intelligent CAD environment for routine mechanical design." *Computer in Engineering,* 1, pp. 111 – 117.

[7] Pratt, M. J., Wilson, P. R. (1987). "Conceptual design of a feature-oriented solid modeler. Draft Document 3B." *General Electric Corporate R&D*.

[8] Miner, R. H. (1985). "A method for the representation and manipulation of geometric features in a solid model." Masters thesis, Massachusetts Institute of Technology.

[9] Cunningham, J., Dixon, J. R. (1988). "Design with features: The origin of features." *ASME Computers in Engineering Conference,* pp. 237 – 243.

[10] Cutkosdy, M., Tenenbaum, J. M., Muller, D. (1988). "Features in process based design." *ASME Computers in Engineering Conference,* pp. 557 – 562.

[11] Turner, G., Anderson, D. C. (1988). "An object oriented approach to interactive, feature based design for quick turnaround manufacturing." *ASME Computers in Engineering Conference.*

[12] Monedero, J. (2000). "Parametric design: a review and some experiences." *Automation in Construction*, 9, pp. 369 – 377.

[13] Loukipoudis, E. N. (1996). "Object management in a programming-by-examples, parametric computer-aided-design system." *The Visual Computer*, 12, pp. 296-306.

[14] Hoffman, C. M., Kim, K. (2001). "Towards valid parametric CAD models." *Computer-Aided Design*, 33(1), pp. 81-90.

[15] Jensen, C. G., Tucker, S. S., Jones, C. J., Rohm, T. (2000). "Concurrent engineering with parametric design." *Proceedings of the Third World Congress on Intelligent Manufacturing Processes & Systems*, Cambridge, MA, pp. 267-273.

[16] Giullian, N., Berglund, C., Jensen, C. G. (2007). "Applying parametric design to global collaboration projects." *2007 PACE Global Annual Forum*, Darmstadt, Germany, July 23-28.

[17] "Merriam-Webster Online Dictionary." http://www.merriam-webster.com/

[18] Lee, D. J., Thornton, A. C. (1996). "The identification and use of key characteristics in the product development process." *Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, Irvine, CA, August 18-22.

[19] Rohm, T., Tucker, S., Jones, C. L., Jensen, C. G. (2000). "Parametric engineering design tools and applications." *ASME Design Automation Conference*, Baltimore, MD, September 10-13.

[20] Anderl, R., Mendgen, R. (1996). "Modeling with constraints: theoretical foundation and application", *Computer-Aided Design*, 28, pp. 155-168.

[21] Chen, M., Huang, D., Zhang, D., Zhang, J. (2004). "Three-dimensional geometric modeling based on parametric design and its application in enterprise production." *Proceedings of the 2004 International Conference on Intelligent Mechatronics and Automation,* 4(2), pp. 733 – 738.

[22] Hoffman, C. M, Joan-Arinyo, R. (1998). "CAD and the product master model." *Computer-Aided Design*, 30(11), pp. 905-918.

[23] Zhengming, C., Ji, G., Shuming, G., Qunsheng, P. (2001) "An incremental approach to converting design feature model to machining feature model." *CAD/Graphics*, August 22-24.

[24] "Scale." *NX Help*.

[25] "NX Design Brochure."
http://www.plm.automation.siemens.com/en_us/products/nx/design/. pg. 8.

[26] Ilies, H. T., Shaprio, V. (1997). "An approach to systematic part design." *Proceedings of the Fifth IFIP TC5/WG5.2 International Workshop on Geometric Modeling in Computer Aided Design on Product Modeling for Computer Integrated Design and Manufacture,* pp. 17 – 31.

[27] Pan, Y., Geng, W., Tong, X. (1996). "An intelligent multi-blackboard CAD system." *Artificial Intelligence in Engineering,* 10, pp. 351 – 356.

[28] Nahm, Y. E., Ishikawa, H. (2006). "A new 3D-CAD system for set-based parametric design." *The International Journal of Advanced Manufacturing Technology,* 29, pp. 137–150.

[29] Nahm, Y. E., Ishikawa, H. (2005). "Representing and aggregating engineering quantities with preference structure for set-based concurrent engineering." *Concurrent Engineering,* 13(2), pp. 123 – 133.

[30] American Chain Association (2006). *Standard Handbook of Chains: Chains for Power Transmission and Material Handling*. CRC Press.

[31] American Society of Mechanical Engineers standards. ASME B 29.17 m – 1983.

Appendix A. Full RPPI code

Header Files:

RPPI_header.h

```
#include "sub_func_header.h"

#ifndef RPPI_header_h
#define RPPI_header_h

int create_default_RPPIexp(double val, char exp_name[50], char
S1_new_exp[50], char S2_new_exp[50], char T_new_exp[50], char
expk_new_exp[50], char expt_new_exp[50]);
int RPPI (tag_t part_tag);

#endif RPPI_header_h
```

sub_func_header.h

```
#ifndef sub_func_header_h
#define sub_func_header_h

#define UF_CALL(X) (report( _FILE_, _LINE_, #X, (X)))
#define WINDOW

#include <algorithm>
#include <cstring>
#include <ctype.h>
#include <fcntl.h>
#include <fstream>
#include <io.h>
#include <iostream>
#include <istream>
#include <iterator>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

```
#include <vector>
#include <windows.h>
using namespace std;

#include <uf_part.h>
#include <uf_obj.h>
#include <uf_modl.h>
#include <uf_curve.h>
#include <uf_disp.h>
#include <uf_attr.h>
#include <uf.h>
#include <uf_defs.h>
#include <uf_styler.h>

int report( char *file, int line, char *call, int irc);

int ask_part_exp(tag_t part, int &num_exps, tag_t *& exps);
int create_exp(tag_t t_input_pt, char *left, char *right);
double get_exp_val(char* exp);
double get_exp_val2 (tag_t exp_tag);
int ask_exp_string (tag_t exp_tag, char *& exp_string);



#endif sub_func_header_h
```

C++ Files:

dr_RPPI_v02.cpp

```
// RPPI_v02.cpp : Defines the entry point for the DLL application.

#include "sub_func_header.h"
#include "RPPI_header.h"

int main (void) {

	tag_t part_tag;
	int curr_part, num_parts;

	//get current part tag
	num_parts = UF_PART_ask_num_parts();

	for (curr_part=0 ; curr_part < num_parts ; curr_part++ ) {
		part_tag = UF_PART_ask_nth_part( curr_part );
	}

	RPPI (part_tag);

	char exp_name[] = "waka=5.0";
	UF_MODL_create_exp ( exp_name );

	return 0;
```

```
}

extern void ufusr (char *param, int *retcode, int rlen)
{
    if ( ( UF_initialize() ) != 0 )
           return;

      ::AllocConsole();
      FILE *fp;
      fp = freopen("conout$", "w", stdout);

      std::cout<<"this prints to the debug window"<<endl;

      main();

    UF_terminate();
    return;
}

extern int ufusr_ask_unload (void)
{
     /* unload immediately after application exits*/
     return ( UF_UNLOAD_IMMEDIATELY );

     /*via the unload selection dialog... */
     /*return ( UF_UNLOAD_SEL_DIALOG );   */
     /*when UG terminates...              */
     /*return ( UF_UNLOAD_UG_TERMINATE ); */
}

extern void ufusr_cleanup (void)
{
    return;
}
```

RPPI_sub_func.cpp

```
#include "sub_func_header.h"
#include "RPPI_header.h"

int create_default_RPPIexp(double val, char exp_name[50], char
S1_new_exp[50], char S2_new_exp[50], char T_new_exp[50], char
expk_new_exp[50], char expt_new_exp[50]){

//    char S_new_exp[130];
//    char T_new_exp[130];
//    char E_new_exp[130];
      char new_exp[130];
      char temp[130];

      UF_MODL_update();

      sprintf(S1_new_exp,"S1_%s=1.0",exp_name);
      sprintf(S2_new_exp,"S2_%s=1.0",exp_name);
```

```
	sprintf(T_new_exp,"T_%s=0.0",exp_name);
	sprintf(expk_new_exp,"expk_%s=1.0",exp_name);
	sprintf(expt_new_exp,"expt_%s=0.0",exp_name);

	UF_MODL_create_exp ( S1_new_exp );
	UF_MODL_create_exp ( S2_new_exp );
	UF_MODL_create_exp ( T_new_exp );
	UF_MODL_create_exp ( expk_new_exp );
	UF_MODL_create_exp ( expt_new_exp );

	sprintf(new_exp,"R_%s",exp_name);
	UF_MODL_rename_exp (exp_name, new_exp);
	sprintf ( temp, "%s=%lf", exp_name, val );
	UF_MODL_create_exp ( temp );

	sprintf(new_exp,"R_%s=(S1_%s*%s+S2_%s*T_%s)*exp(expk_%s*expt_%s)",
exp_name,exp_name,exp_name,exp_name,exp_name,exp_name,exp_name);
	std::cout<<"new_exp: "<<new_exp<<endl;
	UF_MODL_edit_exp( new_exp );

	return true;
}

int RPPI (tag_t part_tag) {

	int i, j, k;
	int num_exps;
	double val;
	tag_t *exps;
	char *exp_string;
	char sc1[50];
	char sc2[50];
	char tol[50];
	char exp_k[50];
	char exp_t[50];
	char str2[] = "=";
	char temp [50];

	ask_part_exp (part_tag, num_exps, exps);

	for (i=0; i < num_exps; i++) {
//	i=0;
		//get the exp name and value
		val = get_exp_val2 ( exps[i] );
		ask_exp_string ( exps[i], exp_string );

		//grab just the name from the exp_string (char *) and store
it in exp_name
		j = strcspn (exp_string, str2);

		std::cout<<"j: "<<j<<endl;

		for (k=0; k<j; k++) temp[k] = exp_string[k];
		temp[j]='\0';

		std::cout<<"exp_string: "<<temp<<endl;
```

```
            create_default_RPPIexp (val, temp, sc1, sc2, tol, exp_k,
exp_t);

            //change the name
//          robust_exp_update (exp_name, sc, tol);
      }

      return true;
}
```

sub_func.cpp

```
#include "sub_func_header.h"

int report( char *file, int line, char *call, int irc){

      if (irc)
      {
            char    messg[133];
            printf("    %s, line %d:  %s\n", file, line, call);
            (UF_get_fail_message(irc, messg)) ?
                  printf("        returned a %d\n", irc) :
            printf("        returned error %d:  %s\n", irc, messg);
      }

      return(irc);
}

int ask_part_exp(tag_t part, int &num_exps, tag_t *& exps){

      UF_MODL_update();

      UF_MODL_ask_exps_of_part ( part, &num_exps, &exps );

      return true;
}

int create_exp(tag_t t_input_pt, char *left, char *right){

      logical exists;
      char new_exp[130];

      UF_MODL_update();
//    printf("checking if %s expression exists...", left);
      UF_MODL_is_exp_in_part ( t_input_pt, left, &exists );
//    printf("\n%d %d", exists, TRUE);
      sprintf(new_exp,"%s=%s",left,right);
//    printf("\nnew_exp is %s  \n\t", new_exp);

      if (exists) {
            UF_MODL_edit_exp( new_exp );
      } else {
            UF_MODL_create_exp ( new_exp );
      }
```

```
      return true;
}

double get_exp_val(char* exp)
{
       double val;
       UF_MODL_eval_exp(exp,&val);
       return val;
}

double get_exp_val2 (tag_t exp_tag){

      double val;
      UF_MODL_ask_exp_tag_value (exp_tag, &val);
      return val;
}

int ask_exp_string (tag_t exp_tag, char *& exp_string){

      UF_MODL_ask_exp_tag_string (exp_tag, &exp_string);

      return true;
}
```