

Automated Iterative Tolerance Value Allocation and Analysis

by

Deepanjan Biswas

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved May 2016 by the
Graduate Supervisory Committee:

Jami Shah, Chair
Joseph Davidson
Yi Ren

ARIZONA STATE UNIVERSITY

August 2016

ABSTRACT

Tolerance specification for manufacturing components from 3D models is a tedious task and often requires expertise of “detailers”. The work presented here is a part of a larger ongoing project aimed at automating tolerance specification to aid less experienced designers by producing consistent geometric dimensioning and tolerancing (GD&T). Tolerance specification can be separated into two major tasks; tolerance schema generation and tolerance value specification. This thesis will focus on the latter part of automated tolerance specification, namely tolerance value allocation and analysis. The tolerance schema (sans values) required prior to these tasks have already been generated by the auto-tolerancing software. This information is communicated through a constraint tolerance feature graph file developed previously at Design Automation Lab (DAL) and is consistent with ASME Y14.5 standard.

The objective of this research is to allocate tolerance values to ensure that the assemblability conditions are satisfied. Assemblability refers to “the ability to assemble/fit a set of parts in specified configuration given a nominal geometry and its corresponding tolerances”. Assemblability is determined by the clearances between the mating features. These clearances are affected by accumulation of tolerances in tolerance loops and hence, the tolerance loops are extracted first. Once tolerance loops have been identified initial tolerance values are allocated to the contributors in these loops. It is highly unlikely that the initial allocation would satisfy assemblability requirements. Overlapping loops have to be simultaneously satisfied progressively.

Hence, tolerances will need to be re-allocated iteratively. This is done with the help of tolerance analysis module.

The tolerance allocation and analysis module receives the constraint graph which contains all basic dimensions and mating constraints from the generated schema. The tolerance loops are detected by traversing the constraint graph. The initial allocation distributes the tolerance budget computed from clearance available in the loop, among its contributors in proportion to the associated nominal dimensions. The analysis module subjects the loops to 3D parametric variation analysis and estimates the variation parameters for the clearances. The re-allocation module uses hill climbing heuristics derived from the distribution parameters to select a loop. Re-allocation Of the tolerance values is done using sensitivities and the weights associated with the contributors in the stack.

Several test cases have been run with this software and the desired user input acceptance rates are achieved. Three test cases are presented and output of each module is discussed.

ACKNOWLEDGEMENTS

I would like to express sincere gratitude to my advisor Dr. Jami J. Shah, for his valuable guidance, teachings, support and believing in me with the task at hand. I am also equally thankful to Dr. J.K. Davidson and Dr. Yi Ren for serving on my committee, sharing valuable knowledge and showing the way with their expertise with difficult problems.

I am thankful to all the current researchers at DAL especially my project team mates for being supportive and involve in brainstorming solutions when approached with a problem.

Finally I am thankful to DMDII (Digital Manufacturing and Design Innovation Institute, grant number - 14-02-05) for financial support throughout the project term.

TABLE OF CONTENTS

	Page
LIST OF TABLES	vii
LIST OF FIGURES	viii
ABBREVIATIONS, NOMENCLATURE AND SYMBOLS	x
CHAPTER	
1. INTRODUCTION	1
1.1 Problem Definition	1
1.2 Background	2
1.3 Scope of Work	3
1.4 Problem Decomposition	5
1.5 Overview of Approach	6
1.6 Organization of Thesis	7
2. LITERATURE REVIEW	9
3. AUTO-TOLERANCING FRAMEWORK	24
3.1 Auto-Tolerancing Framework	24
3.2 Pre-processor Modules	25
3.2.1 STEP to SAT Translation and Input Geometry Refinements	26
3.2.2 Assembly Feature Recognition (AFR)	27
3.2.3 Pattern Feature Recognition (PFR)	30

3.3 Schema Generator	31
3.4 Tolerance Allocation and Analysis Module.....	32
3.5 Tolerance Conversion Module.....	34
4. AUTOMATIC LOOP DETECTION.....	35
4.1 Assembly Stacks	35
4.2 Automatic Loop Detection.....	35
4.2.1 Conceptual Design.....	36
4.2.2 Exhaustive Breadth First Search for loop detection	37
5. TOLERANCE ANALYSIS	41
5.1 Selection of Tolerance Analysis Method for Auto-tolerancing	41
5.2 3D Parametric Variation Analysis	43
5.3 Determination of Sensitivities.....	45
6. TOLERANCE ALLOCATION	47
6.1 Allocation Procedure for Designers.....	47
6.2 Tolerance Allocation for Auto-tolerancing.....	52
6.2.1 Domain Independent Iterative Re-design	53
6.2.2 Initial Tolerance Allocation	55
6.2.3 Tolerance Re-Allocation.....	61
7. TEST CASES.....	77
7.1 Cam Follower.....	78

7.2 Radio Car	79
7.3 Cylinder Body Cap.....	81
8. CLOSURE	85
8.1 Contributions.....	85
8.2 Limitations	86
8.3 Future Work	87
9. REFERENCES.....	88

LIST OF TABLES

Table	Page
2- 1: Tolerance Chart.....	14
3- 1: ASU DAL Pattern Feature Library	31
6- 1: Initial Tolerance Distribution Tab-Slot Feature from Figure 6- 3	59
7- 1: Tolerance Analysis Verification Test Case Time Statistics.....	77

LIST OF FIGURES

Figure	Page
2- 1: Radio Car Support Brackets	13
2- 2: Variation Algorithm for Cylindrical Feature	16
2- 3: Global GD&T Model	19
2- 4: Section A in CTF: File Directory	19
2- 5: Section B in CTF: Part and Member Features.....	19
2- 6: Section C in CTF: Assembly Constraints and Basic Dimensions.....	20
2- 7: Section D in CTF: Tolerances and Associated Degree of Freedom.....	20
2- 8: Section E in CTF; Assembly Hierarchy	20
2- 9: Proposed Auto-Tolerancing Framework under AVM program	21
2- 10: Auto-tolerance Pre-processing Modules under AVM program	23
3- 1: Auto-Tolerancing Framework	25
3- 2: Current Assembly of Auto-tolerance Pre-processing Modules.....	26
3- 3: Assembly Feature Recognition Schematic	29
3- 4: ASU DAL Assembly Feature Library	30
3- 5: General Overview of 1st Order GD&T Schema Generator Module.....	32
3- 6: Tolerance Allocation and Analysis Module	33
3- 7: Data Translation Module	34
4- 1: Example of Geometric Dimensioning and Mating Constraint Graph	36
4- 2: Illustration of Loop Detection Search Tree	38
5- 1: 3D Parametric Variation Analysis.....	43

Figure	Page
6- 1: Cost vs Precision	47
6- 2: Inception of Allocation after Loop Detection	52
6- 3: Tab and Slot Assembly.....	58
6- 4: Initial Allocation.....	60
6- 5: Tolerance Re-allocation Framework	62
6- 6: Cost vs Machining Cost and Scrap rates	63
6- 7: Loss Function	64
6- 8: Distribution of Stacks after Initial Allocation	65
6- 9: Loss Function Discretized	65
6- 10: Acceptance Rates Distribution After 1 st Phase of Re-Allocation	72
6- 11: Acceptance Rate Distribution after Satisficing of Tolerances	74
7- 1: Cam Follower Assembly in Exploded View.....	78
7- 2: Automated GD&T Output for Right Support of Cam Follower	79
7- 3: Final distribution for Critical Stacks of Cam Follower	79
7- 4: Radio Car Assembly Exploded View	79
7- 5: Automated GD&T Output for Radio Car Chassis and Rear Cross Beam.....	80
7- 6: Final distribution for Critical Stacks of Radio Car Assembly.....	80
7- 7: Body Cap Assembly Exploded View	82
7- 8: Automated GD&T Output for Cylinder Body and Cap	84
7- 9: Final Distribution for Critical Stacks of Cylinder Body Cap Assembly	85

ABBREVIATIONS, NOMENCLATURE AND SYMBOLS

1-D: One Dimensional, alternatively written as 1D

2-D: Two Dimensional, alternatively written as 2D

3-D: Three Dimensional, alternatively written as 3D

Assemblability: Term to describe that manufactured component parts successfully fit together

Assembly Feature: Mating feature pairs on distinct parts in an assembly

Basic Dimension: Nominal dimensions defined to identify the controlled dimensions and help locate the tolerance zones

Controlled Dimension: Dimension exclusively specified to have specific tolerance value and not dependent on other tolerances

DoF: Degree of freedom are the position, orientation and size of geometric entities when treated as rigid bodies in 3D

Feature of Size (FOS): Any 3 dimensional feature with a size dimension is a feature of size

GD&T: Geometric Dimensioning and Tolerancing

Master DRF: Feature that is datum to another feature but does not have any position or orientation tolerance

Local DRF: Feature that is datum for another feature and itself has position and orientation tolerances

Primary Tolerances: Position and size tolerances

Product Manufacturing Information (PMI): Manufacturing information like GD&T and surface finish

Secondary Tolerances: Orientation and form tolerances

Tolerance Schema: Specification of tolerance frame parameters like datums, modifiers and material modifiers

Tolerance: Extent of variation permitted from nominal form and size

Tolerancing: Includes tolerance specification, representation, inspection, synthesis and analysis

Uncontrolled Dimension or Sum Dimension: Dimensions not exclusively specified by designer but can be derived from the specified dimension and is not exclusively assigned any tolerance but has a derived tolerance zone from its contributors

Variability: Extent of variance associated with entity

MBD: Core definition of a product in Model Based Enterprise

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

Product Specification is an important part of New Product Development (NPD) process. Engineers design components along with a list of specifications that will determine the quality of the product. These specifications are specific, measurable, attainable and results-oriented. One such field which has enormous scope for automation is tolerance specification. Generating proper GD&T for assemblies takes years of experience and knowledge. Designers often generate incomplete, inconsistent and low-quality GD&T because of either lack of tools or expertise. Such delays in product specification prolong the total product development time. Commercial CAT systems are available that assist designers and detailers to do tolerance analysis, but none of these systems synthesize tolerances. There has been work in progress at ASU Design Automation Lab to fill this gap by trying to automate tolerance synthesis and scale down the reliance on designer's skill and cognition. An automated system can potentially mitigate incomplete and inconsistent GD&T, reduce overall product development time, improve the overall quality of GD&T schema and maximize allowable tolerances conserving assemblability and functionality.

In the desire towards automating tolerance synthesis Haghghi et al [5] classifies the undertaking in 2 primary tasks, schema generation, and allocation/verification. Schema generation primarily involves identifying required tolerance controls, basic dimensioning, datum selection and assigning material modifiers based on some heuristics. It is an under-development tool-set in its advanced stages at ASU Design Automation Lab (DAL). This research here suggests an architecture and execution of

automated iterative 1st order tolerance allocation/verification on the statistical fit basis using a method founded on some fundamental heuristics. The development of a toolset for systematic tolerance value allocation and satisficing those for required acceptance rate for assemblies will be discussed. It aims to automate the 1st order tolerancing and therefore targets tolerancing only for assemblability and does not directly implement production economics or consider functionality in any section of its implementation.

1.2 Background

Tolerances are associated with the nominal dimensions to specify an acceptable variability to the manufactured features of a part. Attaining exact nominal dimension is almost impossible with the current state of technologies and is not required. Components can meet the required performance characteristics even with acceptable variations due to deformation and wear, causing changes in dimension. The designers do not know the environmental conditions under which the component will be used. It is therefore neither possible nor necessary to manufacture parts having nominal dimensions.

Geometric dimensioning and tolerancing (GD&T) is the universal way to convey the design intent and dimensional requirement to the downstream stages till product realization and provides the vital link between products function and process capability. Apart from design intent and manufacturing aspects of GD&T, it is also necessary for metrological purposes i.e. formulating inspection procedures. GD&T is communicated over different stages as mentioned before and therefore underscores the need for a standard practice to specify them. There are two standards in use for specifying tolerances on drawings, ISO 1101 and ANSI/ASME Y14.5M. The

standards have defined different tolerance classes for dimensional variations: Size, and geometric variations: form, orientation, profile, position, and runout as shown in Figure 1- 1. Functionality and assemblability requirements drive the types of variation that will be controlled by the tolerance classes.

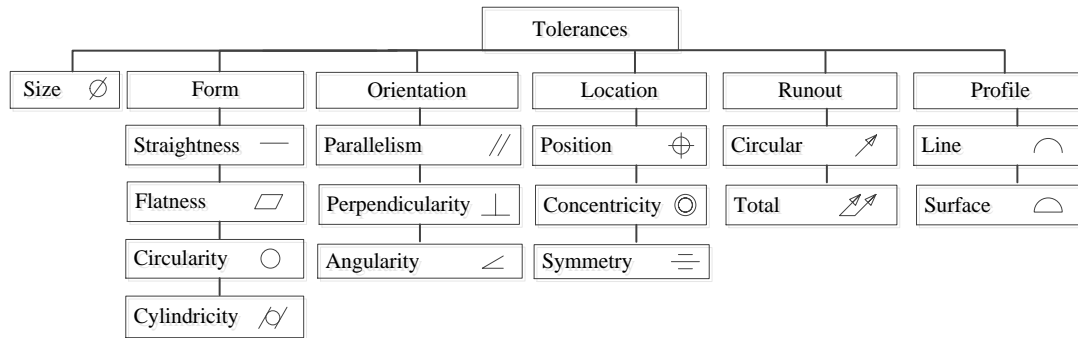


Figure 1- 1: Classes of Tolerances According to the Standards

For example, the form needs to be controlled for surface variations; perpendicularity is critical for insertion of the orientation of a long feature; feature size and location must be controlled for proper assembly. Whereas, datums define a reference system on which these tolerances are applied. A group of tolerance classes, tolerance values, and datums specified in design that control the variations in a part to be manufactured. This entire set of information is commonly referred to as a tolerance scheme. It is not possible to give a review of the entire standards here.

1.3 Scope of Work

The primary objective is to develop and implement a flexible architecture for the tool so that auxiliary features or new tolerance optimization algorithms can be easily added to the tool.

Primary tasks under the research

- Develop modularized architecture to ensure flexibility, adaptability, and scalability for automatic iterative tolerance allocation and analysis

- Design and implement tolerance allocation tool for systematic tolerance allocation and satisficing. Also, develop robust tolerance convergence strategies that ensure convergence or gives suggestive feedback to the user. The research does not aim to formulate a sophisticated optimization algorithm as the primary goal is to explore the possibilities of automating tolerancing and develop the primitive version of auto-tolerancing tool
- Investigate suitability of different tolerance analysis methods for auto-tolerancing and adapt it for the automatic tolerance allocation/verification tool
- Record the outputs of various test cases and discuss the output at different stages.

Tolerancing based on process capability data and functional tolerances i.e. 2nd order and cost optimization tolerancing i.e. 3rd order is not in the scope of this project. Additional toolsets or user feedbacks can be useful for automating 2nd and 3rd order tolerancing and will be tackled in the future. The current state of the art technologies developed at ASU DAL can be used to develop the 1st order tolerancing i.e. tolerances for assemblability automation, and here we describe the value allocation/verification technology for 1st order tolerancing.

To confirm this, the results of test runs on CAD assemblies is also shown here. The tolerances are assigned systematically based on weights and sensitivities associated with the contributor. To support allocation/verification, a separate tool to detect all 1D/multidimensional stacks is also developed and is discussed in detail with outcomes. Future work can include a tolerancing based on cost model and can be included as an additional feature in the current tool-set, but for now, we limit ourselves to satisfying the tolerance allocation for required acceptance rates of

assemblability. It is likewise significant to highlight that profile and runout tolerance under the geometric tolerances in current implementation are not covered as these are important from a functional position and not assemblability. The current implementation aims to surpass or match the quality of GD&T generated by most designers with five years of experience.

1.4 Problem Decomposition

The objective of tolerance allocation and analysis module is to assign values once the tolerance frames have been set up by schema generator. Assigning tolerance is an iterative process and requires simultaneously satisfying multiple overlapping loops. Logically, it was identified that tolerancing for assemblability can be achieved by seeking solutions to the following problems and integrating them in a single system.

Loop (Tolerance stack) Detection: For assemblability, clearances at mating features should be a positive value. The clearances at the mating features are affected by accumulation of tolerances in the loop. Loops are a continuous series of controlled dimensions between mating features that form closed chains.

Tolerance Analysis: Before proceeding with tolerance allocation, an appropriate tolerance analysis method should be selected that can be used to verify the tolerance values that satisfy the assemblability requirements. The analysis method should be able to deal with multi-dimensional loops and give a good approximation of the variation of the analyzed dimension.

Initial Tolerance Allocation: In practice it is easier to assign some initial tolerance values and then improve the values iteratively with analysis. Therefore, good initial

tolerance values should be assigned as a start point. Otherwise as mentioned earlier, it is hard to satisfy all inter-related loops simultaneously.

Tolerance Re-allocation: The tolerance analysis may indicate that the initial allocation does not meet the assemblability requirements. In such situation the tolerances should be adjusted methodically in iteration with tolerance analysis.

1.5 Overview of Approach

Problems mentioned under section 1.4 can be addressed as follows

Loop Detection: The clearances at the assembly features are affected by accumulation of tolerances in the loop. The primary task is to automatically identify these loops and control them. The mating constraints and geometric dimensions relate to the mating features and directly controlled dimensions respectively. This information can be derived from a previously generated tolerance schema. The relation between features, constraints and dimensions forms a graph structure with nodes as the features and the links as the dimensions and constraints. Exhaustive search is carried out on the graph to extract the loops associated to the clearances.

Tolerance Analysis Method: For the purpose of automated tolerance analysis standard tolerance analysis methods could be used like worst case, Root Sum Square (RSS), variation analysis and T-Maps. Since the aim is to control acceptance rates of assemblies, exploring statistical tolerance analysis methods is of main interest. The selected tolerance analysis method is used to compute acceptance rates for positive clearance from the variation parameters of clearance.

Initial Tolerance Allocation: Once the tolerance loops are identified, the values are allocated in such a way that the clearance variations can be controlled. Assigning very

tight tolerances may result in 100% assemblability. But in several cases achieving tight tolerances could be impractical and can raise the cost of product significantly. Therefore, initial tolerance allocation should aim to allocate some realistic values. This may be achieved by considering the geometric dimension related to the contributors. For example a pin with bigger diameter is prone to larger size variation than a pin with a much smaller diameter. Therefore, larger tolerance should be assigned to the pin with the bigger diameter.

Tolerance Re-allocation: The initially allocated tolerance values are analyzed and multiple iterations are carried out to bring all the loops within the desired acceptance rates. Some practical considerations should be made for re-allocation of tolerances just like initial allocation. So consideration may be given to geometric dimensions while re-allocating. Re-allocation should also ensure that the tolerances allocated are within achievable limits.

1.6 Organization of Thesis

In Chapter 2, a detailed literature review for automatic stack detection and tolerance allocation methods. Chapter 3 gives an overview of the auto-tolerancing framework and briefly discusses all the modules of the tool. Chapter 4 describes two alternative automatic tolerance stack detection method, conceptual design, and the implementation. Chapter 5 discusses past DAL work in Tolerance analysis and also discusses the three most important tolerance analysis methods, pros and cons of each method, adaption for auto-tolerancing, conceptual design, and implementation. Chapter 6 discusses some initial tolerance allocation guidelines for designers and the tolerance allocation procedure adapted for auto-tolerancing including the different user input methods, the conceptual design, and implementation. Chapter 7 documents

output from various test assemblies and discusses them to make interpretations.

Chapter 9 concludes the work and addresses limitations and possible future work.

CHAPTER 2

LITERATURE REVIEW

Automated Loop Detection

A tolerance chain is a sequence of features that accumulate to augment variability of the sum dimension. They are also commonly referred as tolerance stacks. To synthesize adequate GD&T these stacks should be analyzed and qualified within a confidence limit. Lai and Yeun [9] developed a vector based transformation scheme. Mohan et al[4] has presented a loop detection method that does not require GD&T schema and can be done directly on a CAD model. But these loops are not exactly stacks, instead assembly feature loops. These do not include the intermediate datum flow chains between the mating features on a part. Therefore these loops cannot be directly used for tolerance analysis. The above methods are difficult to incorporate for allocation/verification purposes as it needs extraction of all loops with defined datum paths. The Loop detection presented in this research is a supporting tool for allocation/verification that extracts all 1D/multidimensional stacks using the constraint graph.

Tolerance Allocation and Optimization

Tolerance allocation encompasses schema generation (datum selection, assigning modifiers and material modifiers), tolerance values and optimization. Tolerance allocation is classified as 1st order allocation, 2nd order allocation and 3rd order allocation. Haghghi et al [5] established this classification based on the intent behind GD&T.

1st order tolerance allocation: Allocation based on assemblability of parts

2nd order tolerance allocation: Allocation based on assemblability and functionality

3rd order tolerance allocation: Allocation based on assemblability, functionality and manufacturing

Several methods have been proposed for tolerance value allocation. Bjork [6] suggests few approaches for designers to establish initial tolerance values. He suggests equal tolerance allocation, tolerance allocation proportional to dimension and tolerance allocation proportional to process deviation. All these approaches first require identifying stacks associated to sum dimension. The initial step common to these approaches is to determine the allowed deviation of sum dimension or tolerance budget based on assemblability and/or functionality requirements. Equal tolerance allocation method suggests distributing this tolerance budget equally over all dimensions in the stack. It has no consideration of any trend in manufacturing variation. Proportional to dimension approach distributes tolerance budget proportional to the dimensions associated to the tolerances. The rationale behind this is that higher process deviation should be associated with bigger dimensions. Proportional to process deviation distributes the tolerance budget proportional to the process deviation associated with dimension. This approach requires prior knowledge of the processes required to manufacture the parts.

Tolerance optimization is one of the most important tasks in product development as it directly affects the functionality and cost of production. Optimizing tolerances takes years of experience, expertise and practical knowledge. Several propositions have been made to optimize tolerances using math models. Speckhart [8] proposes an approach to automate tolerance synthesis using a mathematical model aimed at

minimizing production cost. It targets 3rd order tolerancing and heavily relies on an accurate cost model for each individual feature. Lee and Woo [10] proposed a tolerance synthesis algorithm based on a cost model and proposed an automation theory. It falls short when variability distribution is non-normal, and it is highly likely that distribution will lose normality if orientation and form tolerances are also included. It also fails to include the effect of sensitivities of the contributors in its algorithm. Bowman [11] proposes a gradient based optimization method aimed at reducing manufacturing cost. It estimates the yield gradient by Monte-Carlo Simulation, based on which tolerances are adjusted to achieve the required yield. The above methods target 3rd order tolerancing and relies heavily on cost models. They also fail to propose a framework to automate their methods with inclusion of all tolerance classes. Allocation/verification method proposed in this research utilizes allocation module to systematically assign tolerances. The analysis module evaluates the allocation and computes associated statistical parameters. The Analysis module coupled with allocation module, iteratively synthesizes satisficing tolerance on statistical fit basis.

Tolerance Analysis

Once initial tolerance values are established it's important to conduct tolerance analysis to ensure that the GD&T conforms to assemblability, functional and manufacturing requirements. Usually analysis and allocation are carried out in a loop and usually takes more than a single iteration to satisfy the above requirements. Some of the common methods used for tolerance analysis are as below.

Tolerance charting:

The most common method for tolerance analysis is the tolerance charting method which is a sure fit method. A local co-ordinate system is set up for each stack at the left most feature (horizontally oriented stack) or bottom most feature (vertically oriented stack) of the sum dimension. The direction from this origin to the other feature of the analyzed dimension is the positive direction and negative in the other direction. The final outcome of accumulation is the range of variation associated with the analyzed dimension. For dimensional tolerances when travelling in positive direction upper limit is entered in max column and lower limit is entered in min column. When travelling in negative direction the upper limit goes into the max columns and lower limit to the min column. For geometric tolerances half tolerance value is entered in max column with positive sign and the other half tolerance in the min column with negative sign regardless of direction of travel.

When material modifiers is assigned to the feature bonus tolerances have to be accounted for. When MMC condition is associated with position tolerance zero bonus tolerance is entered in the column with MMC radius and max bonus is entered in column with LMC radius. When LMC condition is associated with the feature then max bonus is entered in the column with MMC size and zero is entered in the column with LMC condition. And when only centerline or center plane of a feature is used in the stack, max bonus tolerances are added on both the columns. When material modifier is assigned to datums, shift tolerances should be accounted for. For MMC to a datum max shift tolerance is entered in the column with LMC size of datum and min shift tolerance is entered in column with MMC size. Conversely, when LMC is assigned to a datum the max shift is entered in the column with MMC size of datum and min shift is entered in column with LMC size of datum. When center line of the

FOS is used, max shift tolerance is entered in both columns. Figure 2- 1 shows a simple illustration of charting method on radio car support brackets and the stack is shown with black dotted arrows. It can be noticed that the size dimensions of both FOS are in negative direction. Therefore in Table 2- 1 maximum sizes are entered in min column with negative sign and minimum sizes are entered in max column with negative sign. Position tolerance for FOS2 is split to both columns. Max shift tolerance from datum *D* i.e. FOS1 is entered in column with min size of FOS1 and min shift is entered in column with max size of FOS1. Max bonus tolerances are entered in the column with LMC size and zero tolerance is entered in the column with MMC size.

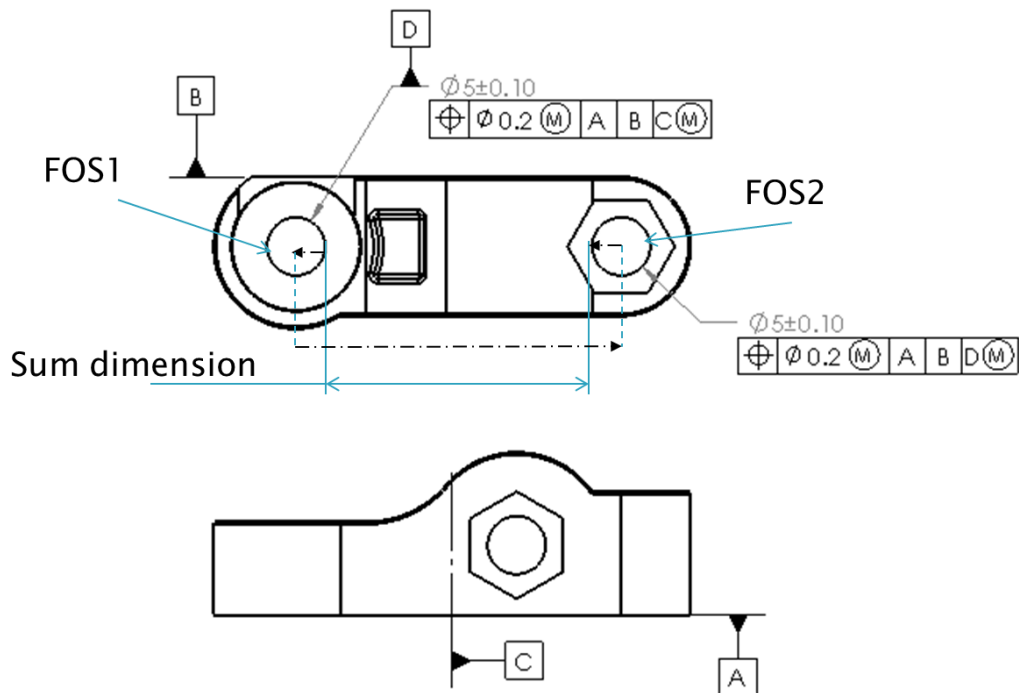


Figure 2- 1: Radio Car Support Brackets

Table 2- 1: Tolerance Chart

	" +/- "	Max	" +/- "	Min	Δ
Size(FOS1)	-	2.45	-	2.55	0.1
Basic dim	+	28	+	28	0
Position (FOS2)	+	0.1	-	0.1	0.2
Shift (datum FOS1)	+	0.15	-	0.05	0.2
Size(FOS2)	-	2.45	-	2.55	0.1
Bonus	+	0.1	-	0	0.1
Total		23.45		22.75	0.7

Linearized Tolerance Analysis :

Often in assemblies while detailing several multidimensional stacks may be encountered. It is just as important to analyze the multidimensional stacks as the 1D stack. In general the dimension to be analyzed can be expressed as function of all the contributing dimensions i.e.

$$A = f(d_1, d_2, d_3, \dots) \quad 2.1$$

This function is referred as the design function and may be non-linear w.r.t some or all dimensions. To simplify the tolerance analysis this function can be approximated by a linearizing the function by taking the first order terms of its Taylor series expansion as mentioned by Shah et al [15]

$$A \approx f(\bar{d}_1, \bar{d}_2, \bar{d}_3, \dots) + \sum_i^n \frac{\partial f}{\partial d_i} \Delta d_i \quad 2.2$$

Both worst case and statistical analysis can be performed once equation is linearized.

The worst can be conducted based on below equations

$$\bar{A} = f(\bar{d}_1, \bar{d}_2, \bar{d}_3, \dots) \quad 2.3$$

$$A \approx \sum_i^n \frac{\partial f}{\partial d_i} \Delta d_i \quad 2.4$$

For statistical analysis the mean can be computed using equation 2.3 and the deviation of analyzed dimension can be computed by below equation

$$\sigma_A = \sqrt{\sum_i^n \left(\frac{\partial f}{\partial d_i} \sigma_{d_i} \right)^2} \quad 2.5$$

The acceptance rates can be computed from the mean and standard deviation values of the analyzed dimension.

3D Parametric Variation Analysis: Linearized tolerance analysis discards the higher order terms of assembly equation's Taylor's expansion which can affect its accuracy. Another shortcoming in statistical linearized tolerance analysis methods is that it's not capable of capturing the refinement effect of the orientation and form tolerances. Linear statistical tolerance analysis captures variability of features only in 1 dimension whereas realistic variations occur in 3D space. This may result in less conservative outcomes thus showing fewer failures than actual. The 3D parametric tolerance variation analysis accounts for the above shortcomings. It is difficult to formulate the closed form equations for 3D variation. Therefore, Monte Carlo simulation is used to collect randomized samples of variation. It treats tolerances as random variables and collects random samples from its PDFs. It computes the transformation matrix for the feature based on these variations. Finally, it combines these transformations in a stack to compute the resulting variation of analyzed dimension. These samples of analyzed dimension variation are used to estimate its population's distribution parameters.

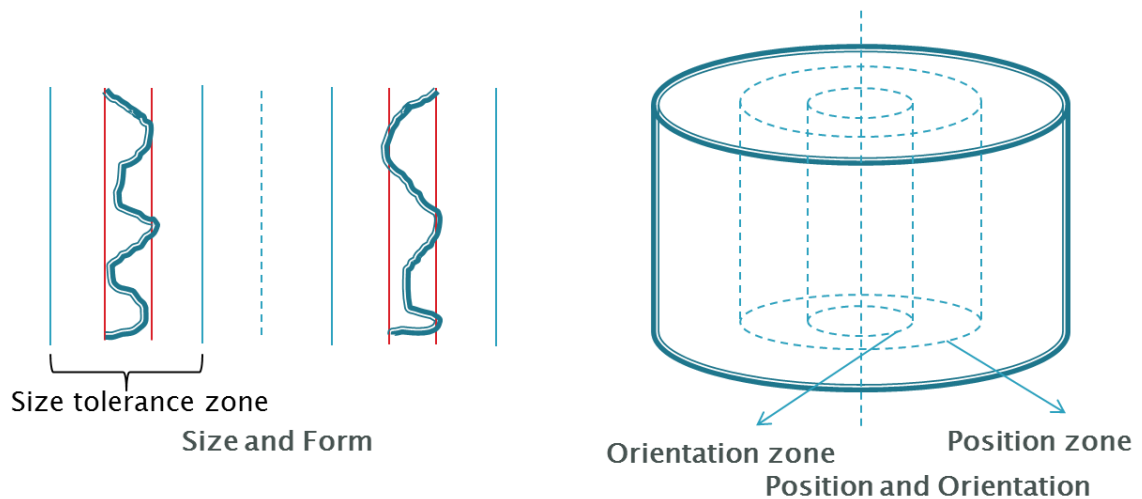


Figure 2- 2: Variation Algorithm for Cylindrical Feature

It uses variation algorithms for different features described by Shen et al [19], but sampling sequence is maintained. For example in a cylindrical feature (Figure 2- 2), the form zone is sampled first with zero mean and given standard deviation (some scaled value of tolerance). Following this the instantaneous form zone is sampled within the size zone. This instantaneous form zone is also sampled within the orientation zone. The resulting zone is then sampled within the position zone. The related actual mating envelope can be computed from the above variations and also the location variations. Based on the values of this variation a transformation matrix is developed for the feature. The algorithm takes care of shift and bonus tolerances by adding correction to tolerance zones.

T-Maps:

A T-Map® is a hypothetical Euclidean point space, the size and shape of which reflects all variational possibilities for a target feature. It is the range of points resulting from a one-to-one mapping from all the variational possibilities of a feature, within its tolerance zone, to the Euclidean point space. These variations are

determined by the tolerances that are specified for controlling size, position, orientation, etc., of the feature. It is currently is an evolving technology and being developed at ASU DAL.

Commercial CAT systems:

Current major CAT packages include *VisVSA* from UGS, *eTol-Mate* from Tecnomatix, *Mechanical Advantage* from Cognition Co., *3-DCS* from Dimensional Control Systems Inc., and *CETOL* from Sigmetrix LLC. Some other computer aided design (CAD) systems such as *IDEAS*® (*IDEAS* is also a registered trademark of UGS) from UGS also have tolerance analysis modules. Of all these packages, the first four (*VisVSA*, *eTol-Mate*, *3-DCS*, *Mechanical Advantage*) and the new version *CETOL* can be broadly classified as one category. Most CAT packages take advantage of the same parametric/variational approach used in CAD systems and apply the Monte Carlo simulation to tolerance analysis (22, 23, 24).

ASU Tolerance Analysis Testbed:

In the past tolerance analysis testbed was developed with C++ for windows platform using HOOPS for rendering. The testbed constituted three tolerance analysis methods Tolerance charting [14], 3D Parametric Tolerance Variation Analysis [18] and T-Maps [20, 21]. A fundamental issue for automatic geometric tolerance analysis is the representation model, which should, in conjunction with CAD models, accurately and completely represent the GD&T specification according to the GD&T standards. The ASU GD&T global model is used to communicate GD&T to the ASU tolerance analysis testbed.

GD&T Global Model :

To transmit geometric dimension and tolerance information completely, to CAT systems a rich data structure is needed. The data structure should be able to capture all types of geometric dimensions, mating constraints, tolerances and features. The comprehensive ASU GD&T Global Model [7] gives just that and can represent all tolerance types in conformity with ASME Y14.5 standards. Though STEP AP242 also allows transmitting the GD&T, but it is not suitable for conducting dimension variation analysis as varying the features directly on CAD model is neither computationally feasible nor is it desirable. This further strengthens the case for utilizing ASU GD&T Global Model represented as the Constraint Tolerance Feature Graph (CTF) as it gives all relevant data for tolerance analysis exempting the CAD model.

Figure 2- 4 is a representation of the global GD&T model and shows the connectedness between GD&T information. The data structure contains 5 main sections

Sec A: contains the name of the B-rep file.

Sec B: contains information about the features in a Part and their data. In case of an assembly this section will have multiple parts and their feature information.

Sec C: contains data about the constraints and metric relations, including the mating conditions.

Sec D: tolerance data and DoFs

Sec E: assembly hierarchy

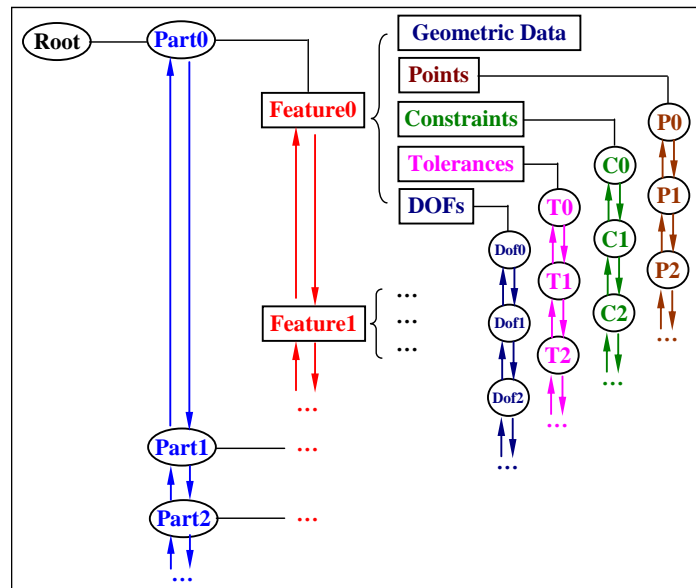


Figure 2- 3: Global GD&T Model

The entities are related in a doubly linked list structure as shown in Figure 2- 3. Figure 2- 4 shows Section A in CTF with the B-rep file name and directory information.

```
#0=FILE('C:\ASU_GDTtestbed\sat\LA04_CTP_internal_assembly.sat');
```

Figure 2- 4: Section A in CTF: File Directory

Figure 2- 5 shows Section B in CTF with parts and their feature information. Parts point to its member features which are trimmed feature representation of real features [7] and these features contain face IDs, location and geometric information.

```
#1=PART('part1', #2, #3);
#2=SLOT('CTP(FACE13&FACE15)_part1', (41.0224, -10, 3.69339), [0, -1.61554e-01
#3=TAB('CTP(FACE5&FACE7)_part1', (-0.993086, -10, 17.9209), [0, 1.61554e-015,
```

Figure 2- 5: Section B in CTF: Part and Member Features

Figure 2- 6 shows Section C with basic dimension pointing to member features and assembly constraints (or mating constraints) pointing to member features. This is particularly useful for stack detection and will be elaborated in section 4.2.

```
#19=CST_DISTANCE(69.463, #11, #12);
#20=METRIC_RELATIONSHIP(#19, CST_DISTANCE, (69.463, #11[PLANE(center plane of TAB)]
#21=CST_M_FLOAT(#2, #6);]
#22=METRIC_RELATIONSHIP(#21, CST_M_FLOAT, (#2[PLANE(center plane of SLOT)], #6[PLAN
```

Figure 2- 6: Section C in CTF: Assembly Constraints and Basic Dimensions

Figure 2- 7 shows Section D with tolerances along with modifiers, datums and material modifiers for both feature and datums. It also has the degrees of freedom DOF restricted by the datums of the target feature.

```
#39=T_SIZE(#8, (nFI, 0.12, RFS));
#40=DOF(#39, (SIZE_DOF, SHAPE_DOF));
#41=T_POSITION(#8, (FI, 0.2, MMC), PD(#9, MMC));
#42=DOF(#41, (#9, TDOF[1, 0, 0], RDOF[0, 1, -1.61554e-015], RDOF[0, 1.61554e-015, 1]));
```

Figure 2- 7: Section D in CTF: Tolerances and Associated Degree of Freedom

Figure 2- 8 shows Section E with the assembly hierarchy with pointers to the parts and sub-assemblies. Sub-assembly #51('assembly_1') points to parts #7 and #10 and #52 ('assembly_0') points to part #1 and #4 and sub-assembly #51

```
#51=ASSEMBLY('assembly_1', #7, #10)
#52=ASSEMBLY('assembly_0', #1, #4, #51)
```

Figure 2- 8: Section E in CTF; Assembly Hierarchy

This model will be used in the auto-tolerancing project described in this research.

Past work in Auto-tolerance at ASU DAL: Work on the auto-tolerancing project began as a part of DARPA Adaptive Vehicle Make (AVM) program. This program consisted of a series of crowd sourced design competitions based on computational tools, DBs, KBs provided by AVM contractors. The objective was to demonstrate and achieve DARPA's target of 10X reduction in development time and cost. Thus, heavy reliance on automated tools was required, both for aiding designers in their tasks (part catalogs, simulation tools) and for DARPA to evaluate those designs (testbenches). Competitors used different CAD systems, so GD&T could not be communicated digitally (STEP AP203 translators in current commercial CAD do not support GDT data). Also, we discovered that even when GD&T could be communicated (via pdf

files or native CAD), competitors often submitted incomplete & inconsistent GD&T that could not be analyzed for assemblability. Automating GD&T from nominal models of assemblies was proposed as a means for solving both the data communication problem and the lack of GD&T expertise amongst competitors.

GD&T on parts directly affects its functionality, assemblability and manufacturability. Encompassing all these factors in automation of GD&T is a paramount task. To simplify the problem, tolerancing is classified based on intent of GD&T as 1st order tolerancing, 2nd order tolerancing and 3rd order tolerancing. 1st order tolerancing is based on assemblability of parts in an assembly. 2nd order tolerancing is based on assemblability and functionality. 3rd order tolerancing is based on assemblability, functionality and manufacturing cost optimization. The initial proposition made by Mohan et al [4] for automating tolerance generation was aimed at 1st order tolerance allocation and tolerancing fits and fasteners. Figure 2- 9 shows the proposed auto-tolerancing framework under the AVM program.

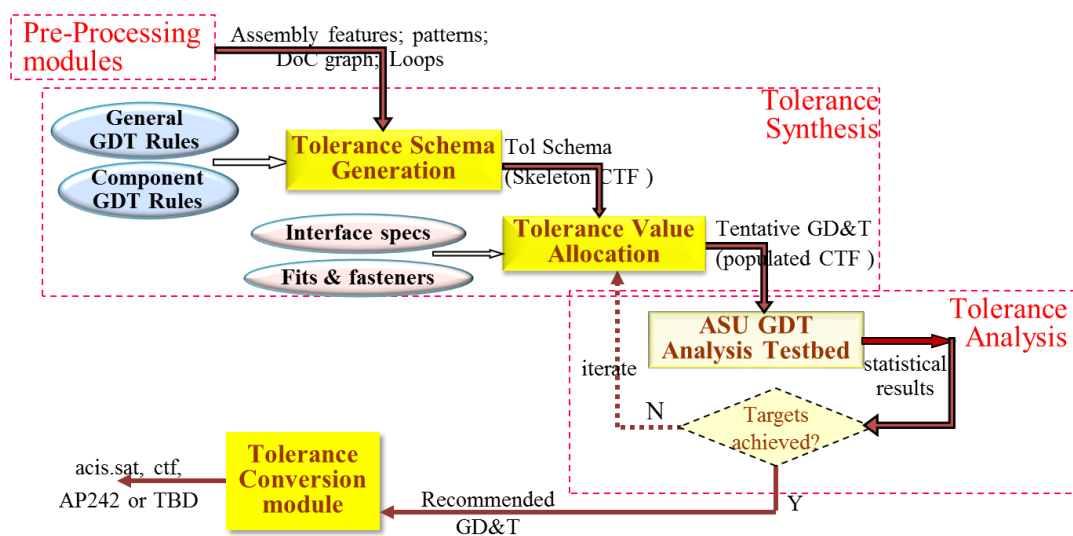


Figure 2- 9: Proposed Auto-Tolerancing Framework under AVM program

The framework can be divided into pre-processing tool-sets and primary auto-tolerancing tool-sets. Past work focused on building the pre-processing tool sets and their assembly is shown in Figure 2- 10. The pre-processing tools constitute the Assembly Feature Recognizer (AFR), Pattern Feature Recognizer and the Assembly Analysis modules. We are particularly interested in mating or assembly features because they dictate assemblability. The AFR detects all mating features in an assembly using the assembly liaison graph with details given in Mohan et al [4]. The Assembly analysis module extracts all possible direction of controls (DoC) in an assembly and detects all the mating pair loops. DCG is useful to determine possible directions of control for a feature. Assembly feature loops may help with tolerance analysis by identifying stacks. The primary tools constitute the schema (datums, modifiers and material modifiers) generator, the allocation/verification and the PMI translation module. Haghghi et al [5] established the M, D, P, L, X rules to generalize schema development for automation. M rules handle selection of primary datums, D and P rules handle selection of secondary and tertiary datums. L rules define the datum flow chains in a schema. X rules handles the refinement tolerances i.e. orientation and form tolerances. Under AVM, the auto-tolerancing project did not constitute development of any of the primary tools of auto-tolerancing. Development of primary tools is carried out under DMDII's initiative to complete tasks under AVM.

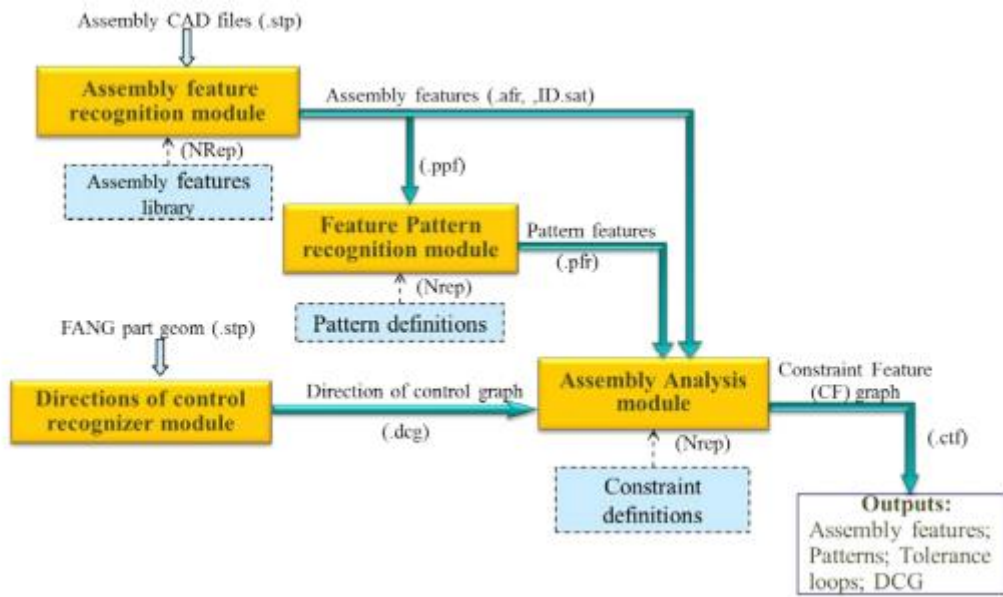


Figure 2- 10: Auto-tolerance Pre-processing Modules under AVM program

CHAPTER 3

AUTO-TOLERANCING FRAMEWORK

This section describes the modified auto-tolerancing framework constituting all the pre-processor and primary modules. It gives a schematic overview of each of these modules and data transmitted to downstream modules.

3.1 Auto-Tolerancing Framework

The auto-tolerancing framework was proposed by Haghghi et al [5]. Shown in Figure 3- 1 is the current standing framework. It starts with assembly STEP AP203 file and outputs the CAD in STEP AP242 standard with the generated PMI information.

The initial proposition of the auto-tolerancing framework was made in the past ASU DAL under the AVM program as discussed under literature review. Some modifications have been made to the current auto-tolerancing framework based on requirements. An additional tool was added to pre-processor to assign persistent IDs to faces in STEP AP203. This information added to AP203 is useful during translation of AP203 to AP242. It allows consistent association of tolerances with its respective features during data translation.

Initial auto-tolerancing proposed the loop detection to be used in the assembly analysis module to identify stacks. But these loops detected are not particularly tolerance stacks and cannot be directly useful for analysis. These loops constitute only the assembly features and no intermediate chains between them. A new loop detection tool has been implemented that identifies stacks once schema is generated.

It is integrated in the auto-tolerancing system between the schema generation and tolerance allocation module. In the initial auto-tolerance framework it was proposed to extend its capabilities to tolerancing fits and fasteners. This seemed to be a bigger

challenge than initially perceived. For example, when assigning tolerances to shaft and bearing the operating torque should be known. This poses challenges when trying to achieve complete automation.

This research describes the method and implementation of the second primary tool-set in the sequence that synthesizes systematic tolerance allocation and verification.

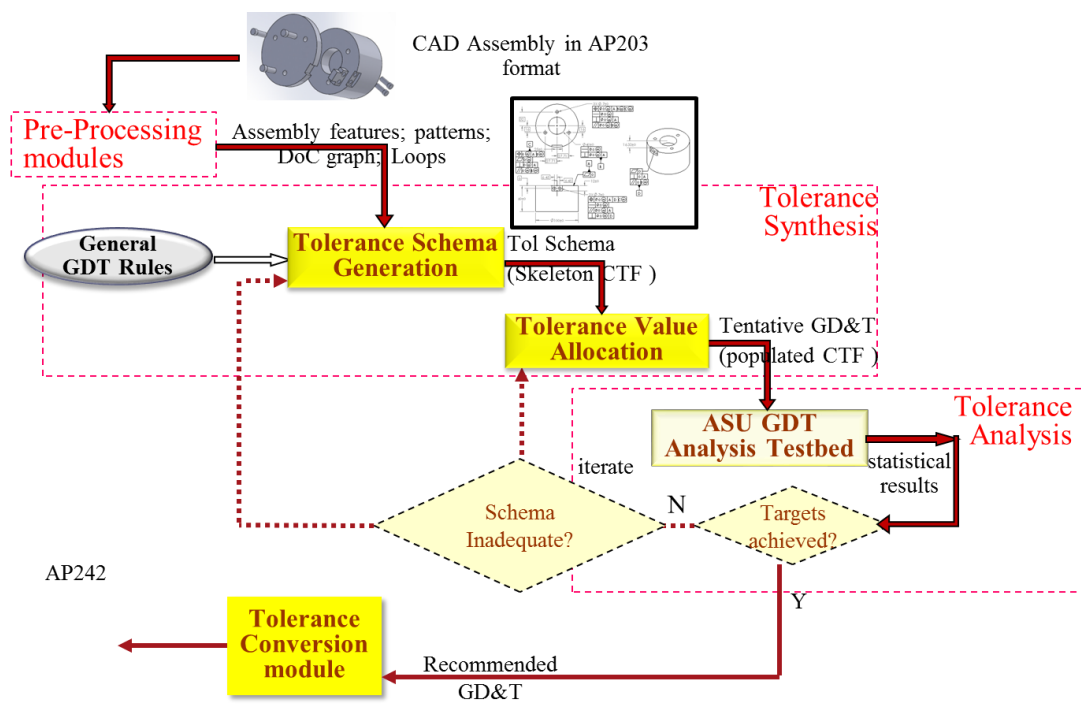


Figure 3- 1: Auto-Tolerancing Framework

3.2 Pre-processor Modules

Before defining GD&T on a part a detailer needs to evaluate the assembly i.e. determine the assembly constraints, determine clearances, kinematic constraints, etc. For automation of GD&T similar evaluations are needed. Past works at ASU DAL have successfully developed some tools for this purpose and assembled under the pre-processing block. Assembly features are the mating feature of sizes and planes. They are of particular interest as these dictate assemblability of parts in an assembly. 1st

order tolerancing should be based on these mating features. Assembly Feature Recognizer is used for automatic detection of these assembly features. Feature pattern recognizer identifies pattern features from AFR output. Identifying pattern features allows allocation of pattern specific tolerances to the pattern features.

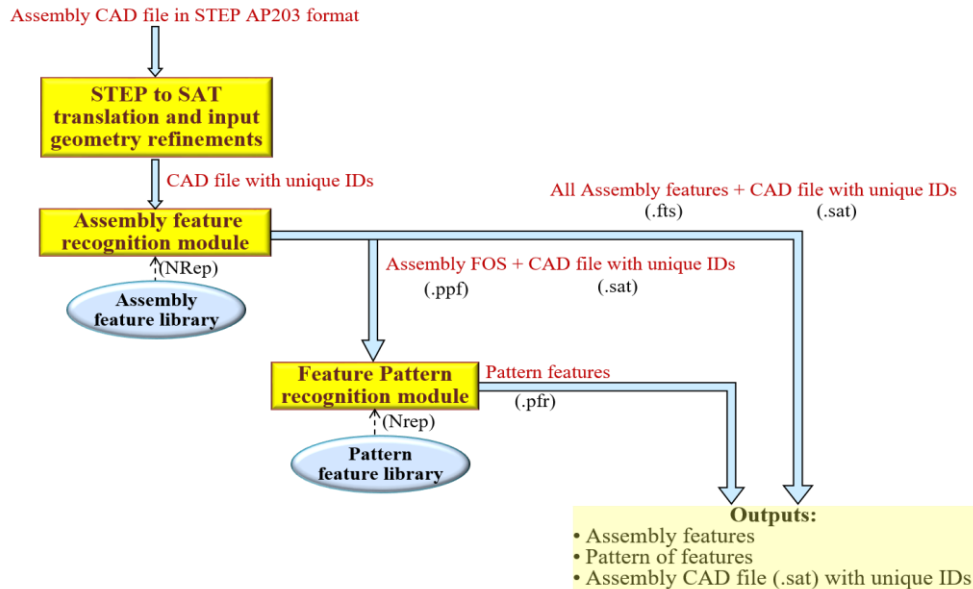


Figure 3- 2: Current Assembly of Auto-tolerance Pre-processing Modules

Figure 3- 2 shows a schematic of the re-designed implementation in the pre-processor block. The interop implementation embedded in the pre-processor translates it to ‘.sat’ ACIS native format. The assembly feature recognition (AFR) module reads in the ‘.sat’ file. Feature Pattern Recognition module uses output of AFR to identify patterned set of features.

3.2.1 STEP to SAT Translation and Input Geometry Refinements

The CAD data is input to the auto-tolerancing software in STEP AP203 file format. Since auto-tolerance software is built upon the ACIS geometric kernel, the STEP AP203 is translated to native SAT format.

During the translation process two additional operations are carried out: merging of half cylindrical faces for cylindrical features and attaching persistent IDs to the faces entities.

3.2.2 Assembly Feature Recognition (AFR)

Figure 3- 3 shows a schematic representation of the AFR. Efficient assembly feature recognition algorithm can be implemented by identifying contact regions. It can greatly reduce the graph size and therefore the search space as will be explained below. But the question stands that how close two faces must be, to be considered contacting each other. We refer to this distance as the proximity value. The ACON implementation in AFR identifies the contact regions based on the proximity value. AFR offers two ways of determining proximity values, adaptive value method and user defined method.

In earlier version of AFR, proximity value for adaptive method is computed as a fixed percentage of the radius of the largest cylinder feature in the assembly. One major shortcoming in this method is that it fails to consider the case when an assembly does not have any cylindrical features. To fix this and make better approximation in the current version an alternate method is used to compute proximity values. The current method takes the equivalent length of smallest part's volume and ratio of largest part to the smallest part into account. Equation 3. 1 shows the formulation of the current adaptive method for computing proximity value.

$$P = \frac{1}{\left(\frac{1}{1.5} \sqrt[3]{r} + 1\right) * 5} * L_{eq} \quad 3. 1$$

Where,

P = Proximity value

$$r = \frac{V_{max}}{V_{min}}$$

$$L_{eq} = \sqrt[3]{V_{min}}$$

V_{min} = Bounding box volume of the smallest part in the assembly

V_{max} = Bounding box volume of the biggest part in the assembly

Once contact pairs are identified by the ACON, the AFR builds the modified face adjacency graph (FAG) constituting all the contact pair faces and their neighboring faces. The nodes of the graph are face entities and the arcs are the edges or the contact links. The modified (FAG) is subdivided into more local level graphs to reduce the search space.

An existing feature library specifies geometric parameters, algebraic parameters, geometric constraints and algebraic constraints for primitive assembly features. The feature library is created through the assembly feature tutor tool [17]. The Assembly Feature Tutor tool was developed at ASU for users to interactively input feature definitions. These tool outputs the feature definitions in N-rep format. These primitive assembly features are read in by AFR and stored as a graph structure. The nodes are faces and the arcs are the edges and the contact links.

The primitive assembly feature graph is matched with the local level modified face adjacency graphs to identify isomorphic subgraphs. Graph matching is a NP hard problem with worst case exponential time complexity [16]. But the graph matching problem for AFR is of smaller size as the primitive feature graphs are relatively very small. Additionally, the modified FAG is divided into smaller local level graphs that further reduce the complexity of graph matching algorithm. Therefore the popular critique of graph matching algorithms being unfeasible is not warranted here.

Once isomorphic subgraphs are identified the geometric parameters, algebraic parameters, geometric constraints and algebraic constraints are verified. The features consistent with primitive feature definitions are the detected features. These detected features' feature IDs, feature types, and geometric parameters are output to a '.afr' file. Figure 3- 3 shows the high level overview of the AFR. Figure 3- 4 shows the current assembly features in the library.

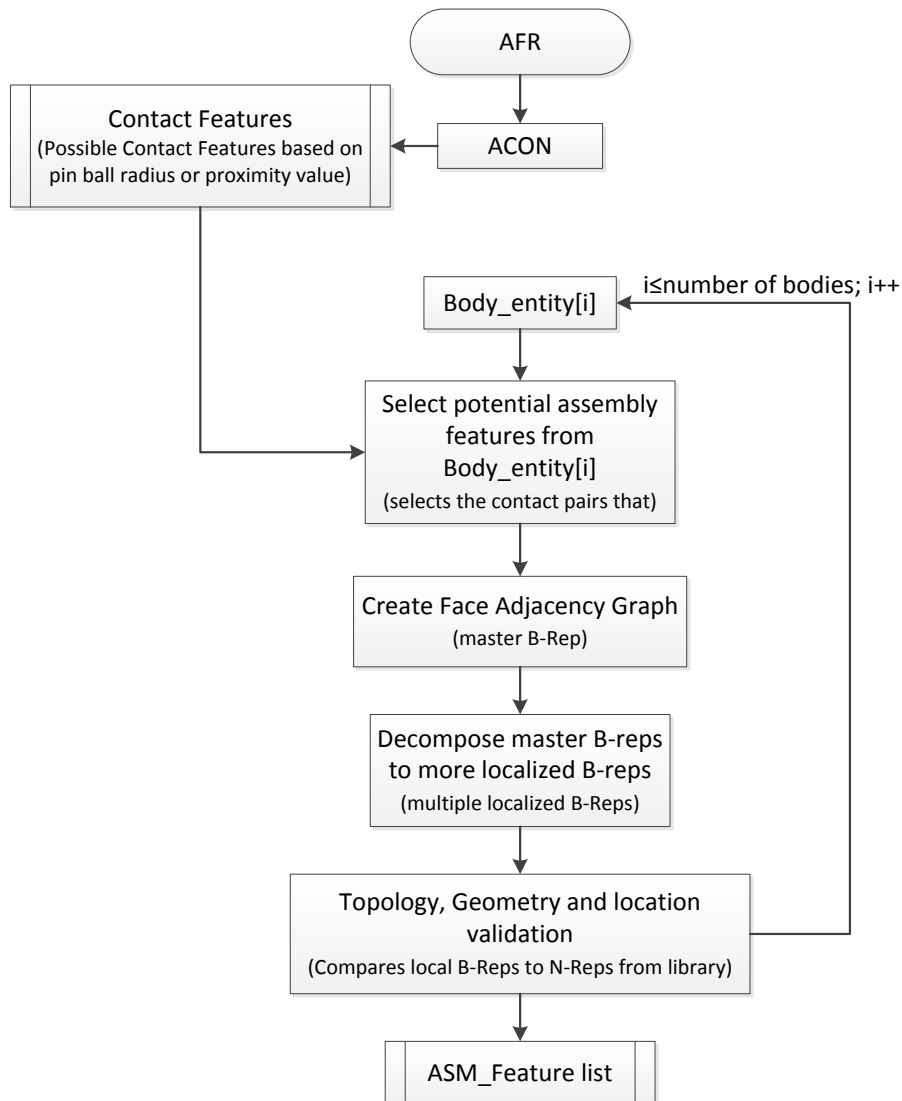


Figure 3- 3: Assembly Feature Recognition Schematic

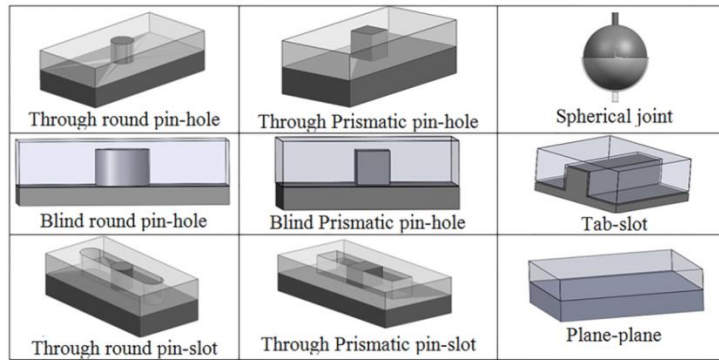


Figure 3- 4: ASU DAL Assembly Feature Library

3.2.3 Pattern Feature Recognition (PFR)

We define a feature pattern as a group of related features at part interfaces. Common arrangements are circular, linear, and rectangular. The mating pattern features also dictate assemblability just like other assembly features. These features are often toleranced as a group. ASME Y14.5 defines some rules for assigning pattern tolerances. These rules are incorporated in auto-tolerancing to account for pattern feature tolerancing.

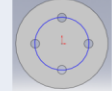
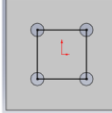


But first we should be able to automatically identify pattern features. Once assembly features have been recognized, we can search for patterns and pattern parameters. Input files for PFR is the ‘.afr’ file, ‘.ppf’ file output from the upstream AFR module and the ‘.sat’ file. There are two main tasks under pattern recognition to detect pattern features

1. Pattern feature matching which is checking conditions for pattern existence between features. Set of features that make potential pattern feature must have same feature type (pin tab, etc.) and same geometric parameters (diameter, width, etc.). They also must lie on the same plane and mate with the same counterpart.
2. Pattern type matching which is matching the pattern type with pre-defined pattern shapes. Figure 5 shows the library of pre-defined pattern types.

Some details of the available patterns in the library of ASU DAL Pattern Feature

Recognizer are given in Table 3- 1.

Table 3- 1: ASU DAL Pattern Feature Library

Patterns	Parameters	Extracted Constraints
 <p>Circular Pattern</p>	<p>-Center of pattern Defined as the center of the circle</p> <p>-R_{pin}: Radius of pin pattern circle</p> <p>-R_{hole}: Radius of hole pattern circle</p> <p>-θ: The angular distance of pins/holes</p>	<p>Global constraint: Center of pin-pattern concentric with center of hole-pattern</p> <p>Position of pins (in the local pattern Co. Sys.) position of hole: $R_{pin} = R_{hole}$, $\theta_{i,pin} = \theta_{i,hole}$</p>
 <p>Rectangular Pattern</p>	<p>-Center of pattern: Defined as the rectangle center</p> <p>- X_{pin} = length of hole pattern/2</p> <p>- X_{hole} = length of pattern rectangle/2</p> <p>- Y_{pin} = width of hole pattern/2</p> <p>- Y_{hole} = width of hole pattern/2</p>	<p>Global constraint: Center of pin-pattern concentric with center of hole-pattern</p> <p>Position of pins match with position of hole: $X_{pin} = X_{hole}$, $Y_{pin} = Y_{hole}$</p> <p>Length = the longer side of rectangle width = the longer side of rectangle</p>
 <p>Linear Pattern</p>	<p>-distance of the pins/holes from the first pin/hole (x) along the pattern line</p>	<p>Global constraint: Position of pins match with position of hole: $X_{i,pin} = X_{i,hole}$</p>
 <p>Collinear pattern</p>	<p>-Holes center line</p> <p>-Hinges length</p> <p>-Hinges distance</p>	<p>Global constraint: Hinges length be less than the other parts hinges distance</p> <p>Global constraint: Holes centerline to be concentric</p>

3.3 Schema Generator

It cannot be stressed enough the importance of developing a good tolerance schema as it directly decides the process design, inspection procedure and cost of manufacturing the assembly. Schema Generator module is designed to develops the tolerance frame for features in the part by evaluating the required tolerance types, datums, modifiers and material modifiers.

The rule sets developed to generate schema [5] is classified into five main rules (M, D, P, L, X). M rules is used for selection of primary datums. D and P rules are used to setup DRFs i.e. select secondary and tertiary datums. L rules help define the datum flow in schema i.e. maintain the right flow of connectedness between datums if multiple DRFs are used for tolerancing part. Finally X rules define the secondary tolerances or refinement tolerances i.e. orientation and form tolerances.

Input in the schema generator is the ‘.sat’, ‘.afr’, ‘.pfr’, ‘.dgc’ and ‘.ld’ files. Using the sat file and outputs from pre-processor tools it identifies the assembly

features, pattern features, direction of potential datum flows and assembly constraint loops as have been described in previous sections. At this stage all the tolerance frames have been setup, but no tolerance values have been assigned. It outputs the feature trimmed geometric representation with face IDs and the tolerance frames with zero tolerance values in a ‘.ctf’ file. Figure 3- 5 shows the schematic representation of current implementation of the schema generator.

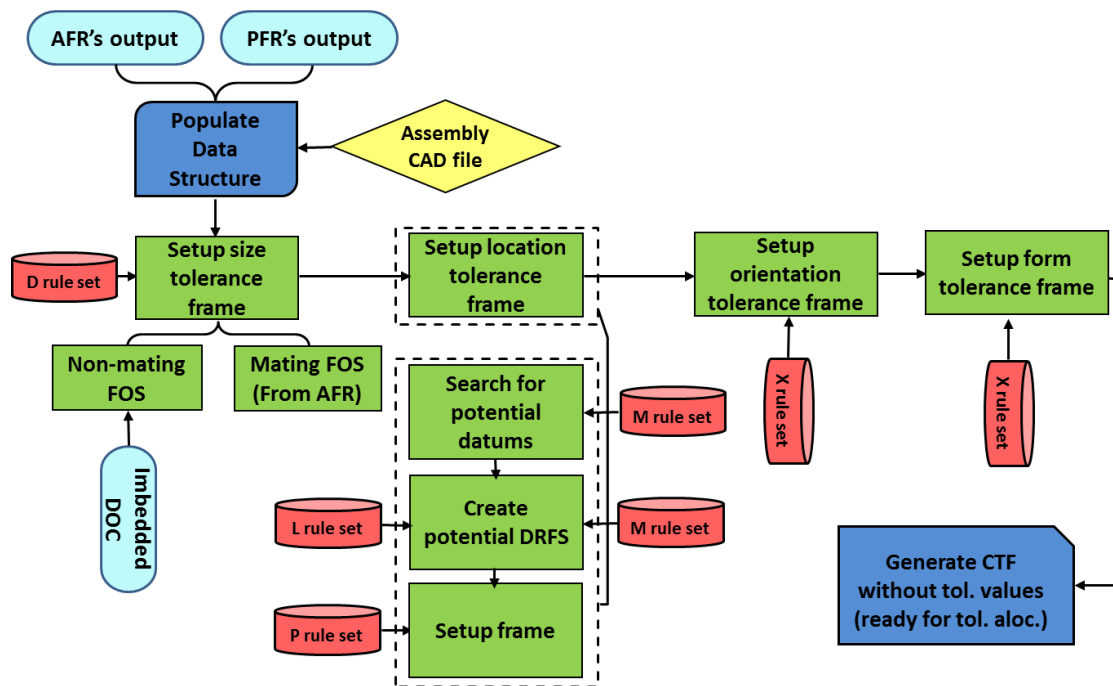


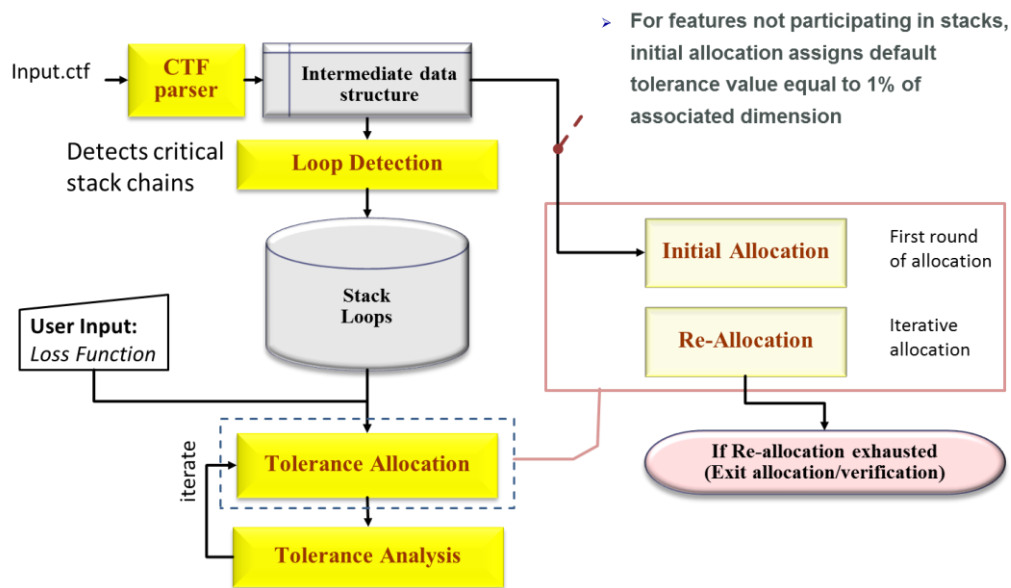
Figure 3- 5: General Overview of 1st Order GD&T Schema Generator Module

3.4 Tolerance Allocation and Analysis Module

The research presented here focuses on this module and will be discussed in more detail in the following sections. The input for this module is the ‘.ctf’ output from the schema generator. The loop detection uses the basic dimension and assembly constraint list from the CTF to detect multi-dimensional assembly level tolerance stacks and is discussed in detail in chapter 4.

These stacks are used to systematically allocate tolerances based on weights and sensitivities. The allocation module iteratively improves the allocated tolerance with

feedback from the analysis module. Allocation tool is subdivided into two separate modules for initial allocation and re-allocation. Initial allocation generates initial tolerance values purely entirely based on weights. Re-allocation module re-distributes and adjusts the tolerances in different stacks based on some hill climbing heuristics. Allocation module is discussed in detail in chapter 6. The tolerance analysis module uses variation analysis and computes acceptance rates for the tolerance stacks that are used by re-allocation module to compute heuristics. Though ASU DAL tolerance testbed has capabilities of automated tolerance charting, RSS tolerance analysis and 3D parametric dimension variation analysis, but in the current implementation of allocation/verification module 3D parametric dimensional variation analysis is used to compute acceptance rates. Chapter 6 discusses the reasoning behind selecting 3D parametric variation analysis over other methods.



11

Figure 3- 6: Tolerance Allocation and Analysis Module

Figure 3- 6 shows the different modules for tolerance allocation and analysis and their interaction. CTF parser reads in the CTF file produced by schema generation module and populates an intermediate data structure. The tolerance values are assigned zero at this stage in the CTF file. Loop detection uses the constraint list to extract tolerance stacks. These stacks are used by tolerance allocation and analysis module to adjust tolerance values. The stack acceptance rates are controlled based on a user defined loss function. The concept of loss function will be introduced under the discussion of tolerance allocation and analysis module.

3.5 Tolerance Conversion Module

GD&T generation is complete at this stage but it is important to translate the tolerance information into standard format so that it can be consumed by all commercial CAD systems. Tolerance conversion module is a translator that specifically translates the tolerance PMI information in CTF to standard STEP AP242 format. It uses STEP Tools dlls to achieve this. Currently several vendors are working on developing STEP AP242 parsers including the PMI information. Model based representation of tolerances in standard format may substitute 2D drawings in the future and will be extremely convenient and save enormous amount of resources that is put into detailing.

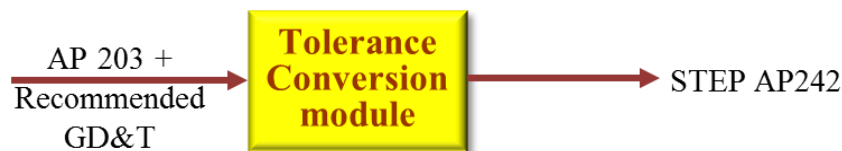


Figure 3- 7: Data Translation Module

CHAPTER 4

AUTOMATIC LOOP DETECTION

This chapter describes a new method to automatically extract all tolerance stacks from the geometric dimension and mating constraint graph that are important for assemblability.

4.1 Assembly Stacks

Before proceeding with estimation of assembly acceptance rates it is important to establish the concept of tolerance accumulation or stacks in a part. The geometric dimensions form chains of these dimensions. The uncontrolled dimensions to which the chain is associated to by the start point and end point is the analyzed dimension. For example, closed chains are usually associated to gaps (uncontrolled dimensions) and therefore gaps are the analyzed dimensions for closed chains.

As already discussed before, every dimension in a part is associated with some variation. In a chain these variations of the dimensions accumulate to augment the overall variability of the chain or stack. This augmented variation of uncontrolled dimensions may diminish the quality and reduce assembly acceptance rates. The mating face gaps dictate the assemblability and should maintain a positive clearance. Therefore, we are particularly interested in controlling the variations of closed chains. In reality, to conserve functionality of the assembly it may be useful to analyze some of the open chains but it is not in the scope of this project.

4.2 Automatic Loop Detection

For auto-tolerancing we are interested in automatic detection of all the closed basic dimension chains in an assembly. The loop detection module is designed to extract all

closed chains automatically. Loop detection is implemented as a stand-alone tool and is readily compatible with ASU DAL GD&T global model definitions.

4.2.1 Conceptual Design

Before describing the loop detection tool itself, it is important to interpret the theory behind it. Geometric dimensions and mating constraints in an assembly form a network and is an undirected (dis-regarding datum flow) weighted graph structure and may not be well connected. The nodes of the graph are features and pattern features as shown in Figure 4- 1. And the links are dimensions (including angular) or the mating constraints.

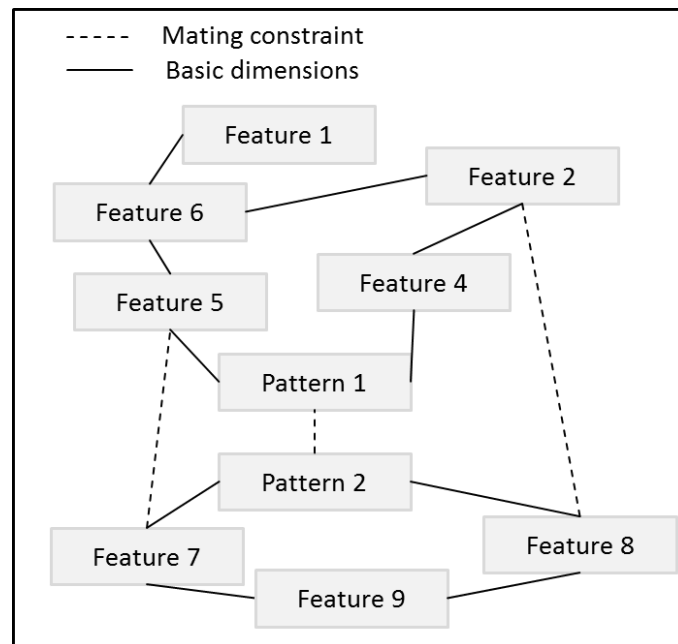


Figure 4- 1: Example of Geometric Dimensioning and Mating Constraint Graph

Fortunately there are standard graph traversal algorithms that can be implemented to detect loops from the constraint and dimension graph. These assembly level loops that can be extracted comprise of part level open chains. To be a valid stack it is to be ensured that these part level chains have a consistent datum flow. The datum flow is consistent if there is one master datum and all the other features stack-up as chain. A

feature cannot have 2 of its datums in the same stack. An assembly feature can be used as a start point to build an exhaustive search tree and find all the closed paths.

Things to avoid while building a search tree so that the loops are detected efficiently

1. When traversing the dimension and mating constraint graph, keep track of the datum flow and terminate the trail in the search tree that violates the datum flow
2. Closed chains comprise of part level chains from different parts in an assembly. It should be ensured that a part has only one set of continuous chain in the assembly level closed chain. This is because tolerance accumulation between two features from the same part happens only through the part level stacks. On the contrary two sets of separate stacks from a part in the same closed chain would mean that accumulation occurs between the set of features in the same part through variations of features in other parts. This would be considered an invalid stack.
3. Ensure same chain do not pass through same feature multiple times
4. Adopt strategies to avoid detecting duplicate loops

A loop detection algorithm designed based on the above listed considerations will not only make the search more efficient but also sidestep adding filters to remove invalid loops and duplicate loops.

4.2.2 Exhaustive Breadth First Search for loop detection

The loop detection tool is developed to extract the closed chains/loops. It traverses the weighted dimension and mating constraint graph to detect all the loops. The dimension and constraint graph is communicated to the loop detection system through constraint tolerance feature (CTF) graph as discussed before under ASU GD&T global model. CTF can communicate mating constrains, length dimensions and angular dimensions. Therefore, CTF is capable of entirely defining all the controlled

degrees of freedom of a feature. Loop detection sequentially picks one mating pair at a time from the CTF mating constraint list. The loop detection algorithm traverses the graph from one of the features of this mating pair. The traversal algorithm uses an exhaustive BFS (breadth first search) tree till all paths to the corresponding mating feature have been traversed. From leaf nodes where the corresponding mating feature is stored, it traverses the tree upwards to extract all the loops. Figure 4- 2 shows a small illustration for fully constrained assembly, corresponding constraint connection list, its graph representation and the loop detection search tree.

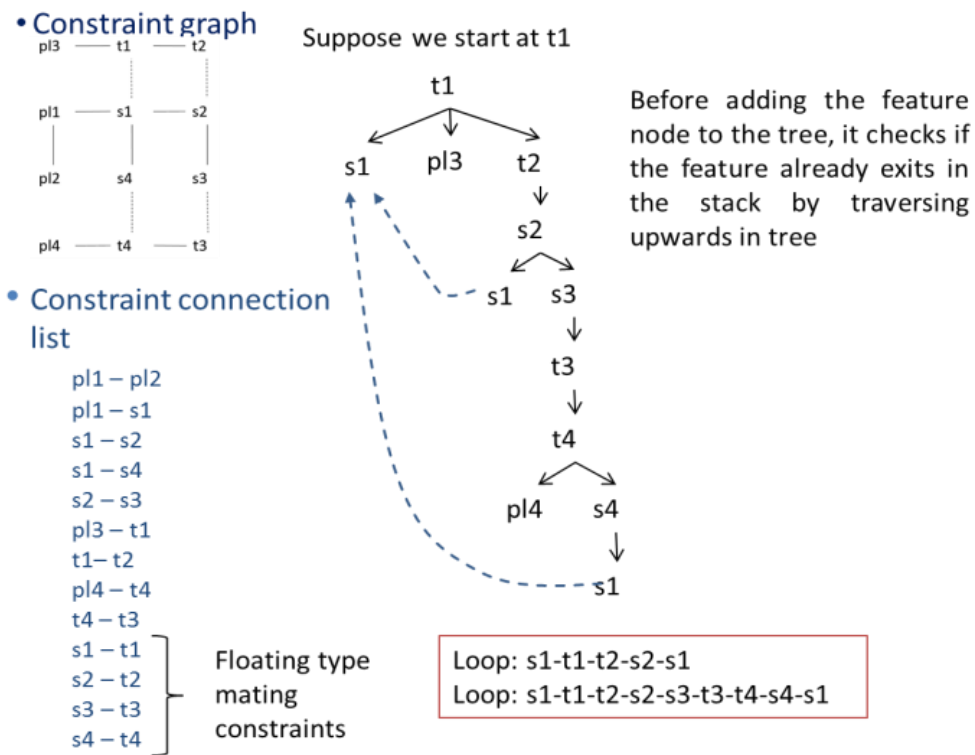


Figure 4- 2: Illustration of Loop Detection Search Tree

The algorithm selects s1-t1 mating pair to detect all closed loops related to the mating pair. The search tree starts building from either of the features of the selected mating pair say node t1. The algorithm traverses the graph by breadth first search till all paths to its corresponding mating feature s1 are traversed. The search tree can be

traversed upwards from the leaf nodes that store the s1 feature to extract the loops. Though the algorithm can be easily extended to detect even the open chains, but for the 1st order tolerancing these are discarded. The algorithm then moves to the next mating pair which is s2-t2 to extract all related loops. It traverses the graph starting node as t2, but this time it ensures to not include s1-t1 mating pair in search tree. As initially when search tree was built for s1-t1 mating pair, all loops that include s1-t1 were already detected in the search. To show this, if the loops for s1-t1 are noticed in figure 5, both cross s2-t2 mating pair. Therefore to avoid detecting these loops once again s1-t1 mating constraint is not included in search tree for s2-t2 mating pair. This not only reduces the size of search trees but also mitigates the need for duplicate loop filter. This may give considerable advantage when dealing with deeper search trees. Also, when adding features, it is important to ensure that it is not repeated in a loop and therefore needs repeated traversal up the trail to the root.

It also may happen that loop detection extracts a chain that contains two or more separate stacks from the same part as algorithm may exit a part from one assembly feature and enter it through another. To avoid entering the same part once exited as may happen when building the search tree, a log of parts visited is kept for each potential stack.

To ensure loop detection maintains consistent datum flow the algorithm ensures that the parent and child of a node are not both datums. Embedding this in the algorithm would reduce unnecessary adding of nodes to the tree and avoid implementation of filters later on in the code. Below is a pseudo code of the current implementation of this method.

Pseudo Code:

Read the constraint list from CTF

Iterate through mating feature list

Assign one of the mating features as root of search tree and add it to queue

Loop through the queue

Loop through all constraints till a connection found

If datum flow consistent and unique feature in the stack

Add features connected by constraint as child member

Store it in queue

Copy and update part logs

End looping through all connection constraint lists

End looping if end of queue

Move to next mating feature and purge the current mating feature from list

End

CHAPTER 5

TOELRANCE ANALYSIS

5.1 Selection of Tolerance Analysis Method for Auto-tolerancing

In GD&T specification process tolerance analysis is necessary to verify the allocated values to ensure specified design requirements are met. Tolerance analysis evaluates uncontrolled dimension variability that occurs because of the accumulation of tolerances. As already mentioned previously ASU DAL Tolerance Analysis Testbed constitutes 3 different kinds of tolerance analysis techniques worst case, 3D Parametric Variation and Tolerance Maps. Worst case is a sure fit approach to tolerance analysis. In other words worst case can determine if the assembly fails in the max or min stack up conditions. 3D parametric variation method imposes random variation to features w.r.t. characteristic probability distribution function (PDF) of assigned tolerances to the feature. After each instantaneous variation a sample of resulting stack up on analyzed dimension are stored. The resulting PDF parameters for the variation analyzed dimension are estimated from these variation samples. Tolerance Maps or T-Maps is a point space whose shape and size reflects the space in which the feature can be varied. Out of these three techniques T-Maps is the most accurate representation of tolerance zones but it is still an under development technology. Its application is being generalized to all types of features of sizes (FOS) like pins, holes and center plane type features.

For the purpose of auto-tolerancing primary interest is in statistical interpretation of dimension variability for reasons that are covered under Tolerance Allocation chapter. Root Sum Square is one of the methods that can be used for statistical tolerance analysis. It needs determination of assembly equations. Assembly equations

are the relation between dimensions in a stack and the analyzed dimension. At times there might be non-linear dependencies in assembly equations. These non-linear dependencies can be simplified through linearization. Linearization can be achieved by determining first order derivatives (sensitivities) of non-linear terms as shown in equation 5. 1.

One of the challenges in RSS tolerance analysis is incorporating secondary tolerances such as form and orientation. If the secondary tolerances are also participating then the subsystems (variables) of the equation may be dependent on other subsystems. For example, orientation affects allowed position variation therefore position is dependent on orientation. Similarly, if surface form is included, it affects the allowed size dimension variation and therefore size is dependent on form. In such cases equation 5. 2 will not hold true. Therefore RSS is incapable of capturing the refinement effect of form and orientation.

$$A = f(d_1, d_2, d_3, d_4, \dots) \quad 5. 1$$

Where,

$A = \text{sum dimension}$

$d_i = \text{contributors}$

$$\sigma_A = \sqrt{\left(\frac{\partial f}{\partial d_1}\right)^2 \sigma_{d_1}^2 + \left(\frac{\partial f}{\partial d_2}\right)^2 \sigma_{d_2}^2 + \dots + \left(\frac{\partial f}{\partial d_n}\right)^2 \sigma_{d_n}^2} \quad 5. 2$$

$$\bar{A} = \frac{\partial f}{\partial d_1} \bar{d}_1 + \frac{\partial f}{\partial d_2} \bar{d}_2 + \dots + \frac{\partial f}{\partial d_n} \bar{d}_n$$

3D parametric variation is another method that can be used for statistical analysis. It randomly varies a feature in its defined tolerance zone. Every tolerance variation is associated with a characteristic probability distribution function (PDF). The random

sampling occurs based on these PDF and these samples are combined to estimate the PDF of analyzed dimension's variation. This tolerance analysis method could be acceptably accurate if enough samples are collected. In most literature available, 30 samples are considered enough to estimate the distribution. This method could be computationally expensive if large numbers of samples are collected (for higher accuracy). But one important advantage of variation analysis is that it does not require exclusive computation of sensitivities and is therefore easier to implement. It also captures the effects of secondary tolerances (refinement effect) and therefore results are more meaningful. For auto-tolerancing purposes we use 3D parametric variation for these advantages. But, enough samples should be collected to maintain accuracy and reduce random perturbation of analysis results on repeating the analysis.

5.2 3D Parametric Variation Analysis

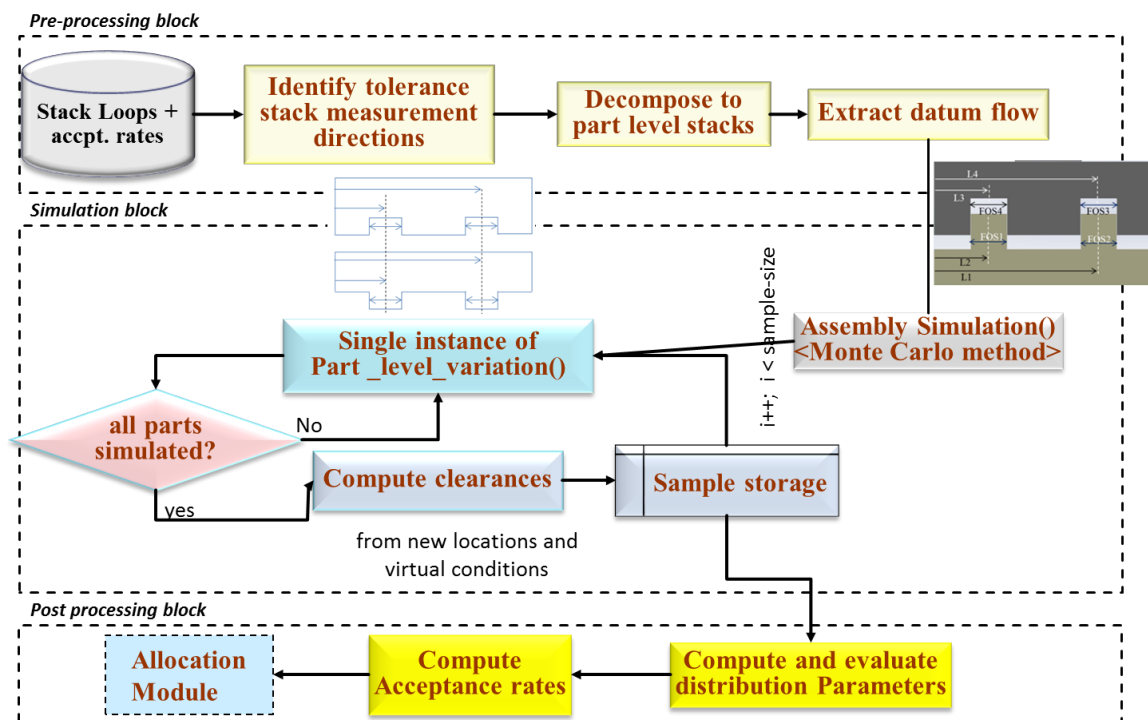


Figure 5- 1: 3D Parametric Variation Analysis

The 3D parametric variation for tolerance analysis is divided into three phases pre-processing, simulation and post-processing blocks as shown in figure. The inputs to the tolerance analysis system are the stacks extracted by the loop detection module.

The first process in the pre-processing module extracts the directions in which it is important to measure the analyzed dimension. For multidimensional loops clearances should be measured in multiple directions. As, in a multidimensional stack the transformation accumulation will have components in different vector directions. These measurement vector directions are simply the directions in which the dimensions of stack exist.

The second function in pre-processor decomposes an assembly level stack to part level stacks to simplify computation of total accumulation. Next, the datum flow is extracted as variations are processed with respect to datum flow pattern. It is also important to follow the datum flow in carrying out variations so that the instantaneous bonus from a datum can be computed beforehand and included in the target feature tolerance zone.

The information generated in the pre-processing module is passed downstream to the simulation block. As already mentioned the stacks are decomposed into part level stacks in pre-processing block, it is then passed by assembly simulation to part level simulation one by one. Part level simulation generates a single instance of variation on each feature in the stack. The resulting transformations of varied features are combined w.r.t. the datum flow to compute final accumulated transformations for the start and end feature. Each instance also generates the related mating envelop for all FOS in a stack and may be used for stack up if the size dimension participates in the stack. These instantaneous transformed locations and related mating envelopes of the

start and end features of the part level stacks are combined by assembly simulation to get overall transformation in assembly level stacks. Finally the clearances can be determined w.r.t the measurement directions. Once enough (acceptable accuracy and random perturbation) samples of clearances are collected they can be used for estimating the PDF for analyzed dimension.

5.3 Determination of Sensitivities

In multidimensional stacks the analyzed dimension may hold a non-linear dependency with some of the contributors. Knowledge of these dependencies of analyzed dimension could be useful particularly for tolerance analysis and to strategically allocate tolerance values. As we use variation tolerance analysis we don't need exclusive evaluation of the dependencies for tolerance analysis. But we are interested in evaluating the dependencies particularly to allocate tolerances based on these dependencies.

The dependencies can be estimated using linearization by determining the sensitivity of the tolerance on analyzed dimension variation. Chase et al [12] describes the vector loop method for computing sensitivities. Sensitivities are the first order terms of the Taylor's expansion of the assembly equation. This method does not address floating type mating features where the kinematic joints of the vector loop can have multiple degrees of freedom.

In the current implementation sensitivity is estimated by a simple two point method. Two point method is an efficient method to compute derivatives when accuracy is not critical. Also, since these partial derivatives are not affecting numerical solution convergence, a simple method can be used to approximate them. It induces small

change in the variable and computes the change in the dependent variable. Therefore, small amount of change is induced in the tolerance value. Using Variation tolerance analysis the change in analyzed dimension variation is estimated. The sensitivity can be determined by taking the ratio of induced change in tolerance value to the resulting change in tolerance value of the analyzed dimension as shown in equation 5. 3.

$$S_i = \frac{\partial T_i}{\partial T_{stack}} \cong \frac{\Delta T_i}{\Delta T_{stack}} = \frac{\Delta \sqrt{V_i}}{\sqrt{\Delta V_{stack}}} \quad 5.3$$

Where,

S_i = *sensitivity of contributor*

V_i = *variance of contributor*

V_{stack} = *variance of stack*

CHAPTER 6

TOLERANCE ALLOCATION

This chapter starts of by discussing a general procedure that designers can use for tolerance allocation. The chapter then continues to discuss the adaption of a heuristic iterative re-design system for tolerance allocation. The chapter then moves to discuss different components of this tolerance allocation module. It uses ideas from the allocation procedure suggested at the beginning of this chapter and of the iterative re-design system.

6.1 Allocation Procedure for Designers

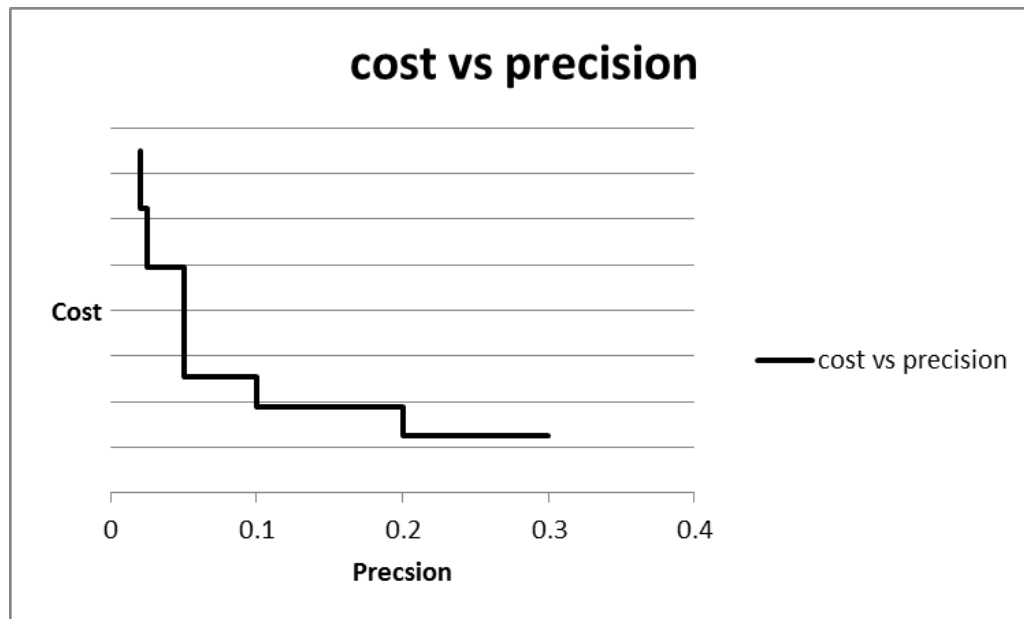


Figure 6- 1: Cost vs Precision

The objective of tolerance allocation is to meet different requirements e.g. assembly, functional, cost etc. Assembly and functional objectives can be met by simply allocating narrow tolerances to the features and dimensions. But narrow tolerances i.e. higher precision can increase the production cost of parts exponentially. Figure 6- 1 from Bjork [6] represents a typical exponentially increasing stepped

function of precision vs cost. Therefore, it is greatly desirable to widen tolerances but without compromising the minimum functional and the minimum assemblability requirements.

Furthermore, a wider tolerance can reduce the cost of manufacturing significantly when in the region of higher slope. For example looking at Figure 6- 1 widening the tolerance from 0.025 to 0.1 will result in large cost reduction but widening tolerance from 0.1 to 0.3 has relatively lesser cost reduction. Therefore tolerances should not be impracticably tight in places and needlessly wide (insignificant reduction in cost and may aggravate the quality of part) in others. Therefore strategies to allocate tolerances should be tailored to reduce the cost of a product simultaneously keeping the quality within acceptable limits. This usually cannot be achieved at one shot as several tolerance stacks are to be satisfied simultaneously. Therefore tolerance allocation is usually an iterative process.

The evaluation of tolerance allocated after a single iteration is done using tolerance analysis. Worst case or statistical tolerance analysis methods could be used to achieve this. Worst case tolerance analysis ensures 100% fit. Though, in a mass production scenario some assemblies fail to meet assembly or functional dimension requirements because of random variation of machining processes. This may not make it possible to control dimensions within given bounds if they are too narrow. In general these failures are permitted. As trying to control the dimension within the given tolerance zone 100% of the times could be very expensive (exponential increase with precision). Adopting statistical (maybe simulation based) tolerance analysis would allow the designer to estimate the failure rates and directly control the allocation based on acceptable failure rates of the assembly.

A general approach to allocate tolerance to parts based on statistical fit can be easily established for designers. Equation 6. 1 equates variability of analyzed dimension to the variability of the contributor which is the RSS approach to tolerance analysis. This equation assumes that probability distribution of the contributors is normal resulting in a normally distributed analyzed dimension. The sensitivities may be computed using vector loop method [12] or some other perturbation techniques.

$$\sigma_{\Sigma}^2 = \sum_{i=1}^n (s_i \sigma_i)^2 \quad 6. 1$$

Where,

n = number of contributor

σ_{Σ} = Standard dev. of analyzed dimension w.r.t. to the stack

σ_i = Standard dev. of i^{th} contributor

s_i = Sensitivity of i^{th} contributor

As the acceptable range of variation for analyzed dimensions are pre-determined (assemblability or functional dimension limit) by designers, the required total variance of a stack can be easily computed if the target acceptance rate of the stack is pre-determined. So a strategy could be to set several intermediate targets in short intervals (as all interrelated stack should be satisfied simultaneously) for the stacks and then try to adjust the tolerances to achieve these acceptance rates for each target. For each iteration, select the worst conditioned (either failing or has very tight tolerances) loop. Then satisfy this stack for the next intermediate target acceptance rate. The general procedure that can be adopted by designers for allocation assuming variability is normal, is as follows

1. Identify the important uncontrolled dimensions to be analyzed
2. Determine all the contributors in a stack
3. Determine the geometrical dependency between contributors and sum dimensions
4. Determine the sensitivities (coefficient of linear term in Taylor expansion) of contributors if non-linearly related by taking derivative or Euler's method
5. Define the confidence interval for the sum dimension value i.e. the acceptance rate for the stack. If dealing with inter-related stacks allocation should be done in iterations i.e. set several intermediate targets. This ensures that tolerances equally distributed over all the interrelated stacks
6. Say a gap dimension is being analyzed for assemblability then the allowed standard deviation of the sum dimension can be assessed as follows

$$\sigma = \text{Gap}/n \qquad 6.2$$

Where,

$\sigma =$ *Sum dimension standard deviation*

$\text{Gap} =$ *sum of clearance of all assembly features in a stack*

$n =$ *range on unit normal for required acceptance rate*

e.g. assuming 99.7% which is the 3-sigma value acceptance rate

$$n = 6$$

$$\sigma = \text{Gap}/6$$

Tolerances are assumed to be a scaled value of standard deviation of feature variation and are usually 6 times the standard deviation

$$\Delta = 6 \text{ Gap}/n \quad 6.3$$

Where,

$$\Delta = \text{Tolerance budget}$$

The above equation to compute tolerance budget assumes normality of stochastic effect of the tolerances in the stack. Another assumption is made that the designer always leaves a positive clearance in the mating features such that positive tolerance budget is computed. This assumption may fail when zero clearance or negative clearance mating features exist in a stack. This issue will be addressed and solutions will be proposed under initial allocation in this chapter.

7. Distribute the tolerance on the primary contributors (position and size tolerances) in such a way that following equation is satisfied

$$\Delta^2 = \sum_{i=1}^n (S_i T_i)^2 \quad 6.4$$

Where,

$$T_i = \text{Tolerance of } i^{\text{th}} \text{ contributor}$$

This equation is equivalent to equation 6. 1.

The above suggested process can be implemented manually by a designer. But it could be a tedious process even when dealing with moderately sized assemblies because of reasons like interrelated stacks, multi-dimensional stacks, lack of experience with GD&T. etc. A convenient alternative solution would be to automate the above suggested approach.

6.2 Tolerance Allocation for Auto-tolerancing

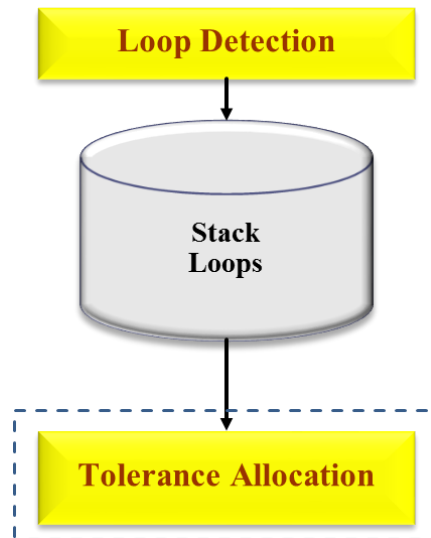


Figure 6- 2: Inception of Allocation after Loop Detection

From the detected stacks (output of loop detection tool), assembly (system) equations can be formed. These equations have to be satisfied for functionality and assemblability. Figure 6- 2 shows the flow of allocation process after the loop detection. Assuming all potentially critical stacks have been identified by loop detections, the auto-tolerancing system process can proceed with tolerance value allocation

For assemblability, evaluation of assembly equations for closed chains should be greater than zero i.e. clearances at mating faces greater than zero. In current implementation we use statistical evaluation to meet overall acceptance rates requirements for the assembly.

The main aim of this research is not to suggest a robust and precise tolerance optimization model, but to describe a flexible auto-tolerancing architecture and tolerance satisficing allocation scheme. Nonetheless, the flexibility added to the software design allows optimization schemes to be incorporated in the future. The re-

allocation method presented here is primarily based on intuitive scoring system and heuristics. Analysis module gives feedback about the acceptance rates and distribution parameters of stacks, which is utilized by allocation module to re-allocate or re-adjust tolerances. Therefore analysis and allocation is iterated till all or maximum number of stacks is in satisfactory bracket and distributed based on a user defined loss function. This will be described under re-allocation. First, we describe how domain independent heuristics driven iterative re-design methods can be used for allocation.

6.2.1 Domain Independent Iterative Re-design

Dixon et al [3] developed a system “Dominic” an adaptive and domain independent iterative redesign method. This system encodes a heuristic hill climbing method that supports achieving various levels of satisfaction for multiple objectives. The Dominic system is useful for design problems when relations between variables and the performance parameters are complex, hard to define and acceptable zone design suffices i.e. optimum design is needless. The ultimate objective of the Dominic system is to attain an acceptable solution (not optimal) by iterative redesign. ASU Design Shell developed at ASU DAL is based on similar principle of iterative redesign with the aim to reach an acceptable design. But there are several fundamental differences between both these systems. ASU Design shell uses mathematical model unlike Dominic which uses Heuristic knowledge. ASU design shell evaluates the design based on a utility function unlike Dominic which uses levels of satisfaction given on ad-hoc basis. Moreover, the ASU Design Shell is user driven iteration whereas Dominic is automatic. Some of the features of the Dominic and the ASU Design Shell can be useful if integrated for tolerance allocation problem. Like for tolerance allocation the ASU Design Shell system would allow changing set of

tolerances (multiple independent variables). This will be useful for faster convergence and uniform allocation of tolerance values. Dominic allows changing one variable per iteration and is not suitable for uniform distribution of tolerance or fast convergence. One very important feature of Dominic is that it uses pre-specified solution procedure which is rule based and therefore iterations can be easily automated.

An adaptation of ideas from Dominic method and ASU Design Shell for the tolerance allocation problem is shown below

1. **Problem specification parameters** - Desired assembly acceptance rates and the critical loops
2. **Variables**- Contributors of stacks
3. **Performance parameters** – Stack acceptance rates
4. **Specified initial design**- Initial allocation
5. **Analysis** -Using MC simulation
6. **Dependencies**- Assembly equations
7. **Dependency order list**- Contributors sorted based on weights and sensitivity
8. **Perf satisfaction** - Based on User defined utility function

Both Dominic and ASU Design Shell need an initial design before the iterative redesign process. This may be made with some rationale behind it but rarely this would give the best solution. The initial allocation assigns initial tolerance values and is described below.

6.2.2 Initial Tolerance Allocation

This section first describes some methods that can be used by designers to distribute the tolerance budget at an early stage assuming 1D stacks. Then the section proceeds to the adaption of one of these methods in auto-tolerancing.

Equal tolerance distribution:

Once the tolerance budget is computed using equation 6. 3 the tolerance budget can be distributed with different methods. One simple method is to equally distribute tolerances between primary contributors. Therefore,

$$T_i = \frac{\Delta}{\sqrt{n}} \quad 6. 5$$

$T_i = \text{Tolerance of } i^{\text{th}} \text{ contributor}$

$\Delta = \text{Tolerance budget}$

The above expression satisfies the equation 6. 4 assuming sensitivities are unity.

Where,

Tolerances proportional to dimensions:

The size of variation in dimension is generally also associated to larger dimension values. Usually the larger the dimension the larger is the variation. Tolerance allocation based on proportionality with the dimension can better capture this behavior and is more suitable than the previous method.

$$w_i = \frac{d_i}{\sqrt{\sum_j^n d_j^2}} \quad 6.6$$

$$T_i = w_i \Delta$$

$d_i =$ dimension associated to contributing tolerances

The dimension is divided by root sum square to compute weights of primary tolerances. The assumption made here is that primary tolerance variance should sum up to variance of sum dimension i.e. satisfies equation 6.4 assuming sensitivities are unity.

Tolerances proportional to Process Deviation:

Tolerances can be also allocated proportional to the standard deviation associated with the machining process of a feature.

$$w_i = \frac{\sigma_i^2}{\sqrt{\sum_j^n \sigma_j^2}} \quad 6.7$$

$$T_i = w_i \Delta$$

Where,

$\sigma_i^2 =$ is the process variation for i^{th} feature in stack

Initial Allocation method for auto-tolerancing:

Evaluating the methods discussed above a suitable method for initial allocation is selected. Estimating the process deviation needs pre-determined machining process for each feature and may not be ideal for auto-tolerancing. This goes away if there is some reliable feature recognition technology available that specifies suitable

machining processes for features. Equal tolerance budget distribution dis-regards the fact that bigger dimensions are prone to higher variability. The tolerance allocation proportional to dimension method seems to be a more meaningful method as it tends to associate higher variability with bigger dimensions. Moreover, it can be easily evaluated from geometry unlike tolerances proportional to process deviation. This method is adopted for generating the initial allocation in the auto-tolerancing tool and is described below.

Each contributor is associated with a weight factor and is a heuristic attribute computed using equation 6. 6. Tolerance budget is computed from the total nominal clearance at mating faces in the stack. If total clearance in stack is zero or negative the lower limits of tolerance values (adaptive or user defined) can be allocated to the contributors.

Initial allocation uses only the weights for distributing tolerance irrespective of the contributor being in plane or out of plane (multidimensional stack). As the sensitivities (derivatives) are unknown at this stage of allocation, 1D stack assumption is maintained. Once initial tolerance values have been established, sensitivities can be computed by perturbing the values. The method can be better explained by a simple illustration and can be directly extended to more complex assemblies. Figure 6- 3 shows an assembly of a part with two slot features mating with a part with two tab features. Equation 6. 8 shows tolerance budget evaluation for the target acceptance rate (AR). Equation 6. 9 shows root sum square of dimensions.

Table 6- 1 shows the allocated tolerances for the dimensions. This assumes variation of feature is linear which may not be accurate. But nonetheless it should give a good approximation and should drive the solution to faster convergence. Also,

since the nominal dimensions don't change, the weights are an invariant heuristic attribute of a contributor.

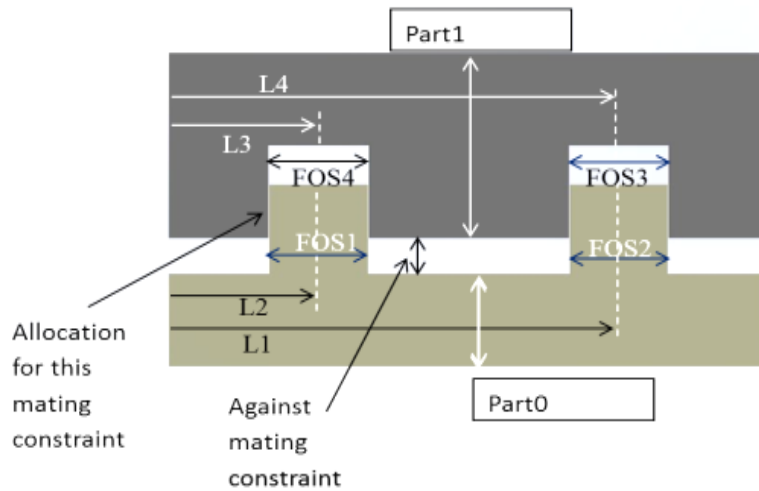


Figure 6- 3: Tab and Slot Assembly

6. 8

$$\Delta = \frac{6 \left(\frac{\text{FOS4} - \text{FOS1}}{2} + \frac{\text{FOS3} - \text{FOS2}}{2} \right)}{n}$$

Where,

Δ = tolerance budget

Unit normal probability between $\pm n$ outcome = AR%

Tolerance budget is computed from total gap in the stack. For tolerance analysis we adjust the parts in such a way that the analyzed dimension adds up to be the total gap in the stack. Even during variation analysis the instant total gaps are measured (in different measurement directions for multidimensional stacks) for each stack. We now evaluate dimension root sum square

$$L = \sqrt{\left(\left(\frac{FOS_1}{2} \right)^2 + (L_2)^2 + (L_1)^2 + \left(\frac{FOS_2}{2} \right)^2 \right) + \left(\frac{FOS_3}{2} \right)^2 + (L_4)^2 + (L_3)^2 + \left(\frac{FOS_4}{2} \right)^2}$$

Where,

L = Root sum square of dimensions

Table 6- 1: Initial Tolerance Distribution Tab-Slot Feature from Figure 6- 3

Dimensions	Weights	Tolerances
FOS_1 (half size)	$\frac{FOS_1}{2L}$	$\frac{FOS_1}{2L} \Delta$
L_2	$\frac{L_2}{L}$	$\frac{L_2}{L} \Delta$
L_1	$\frac{L_1}{L}$	$\frac{L_1}{L} \Delta$
FOS_2 (half size)	$\frac{FOS_2}{2L}$	$\frac{FOS_2}{2L} \Delta$
FOS_3 (half size)	$\frac{FOS_3}{2L}$	$\frac{FOS_3}{2L} \Delta$
L_4	$\frac{L_4}{L}$	$\frac{L_4}{L} \Delta$
L_3	$\frac{L_3}{L}$	$\frac{L_3}{L} \Delta$
FOS_4 (half size)	$\frac{FOS_4}{2L}$	$\frac{FOS_4}{2L} \Delta$

Figure 6- 4, shows the initial allocation module flowchart

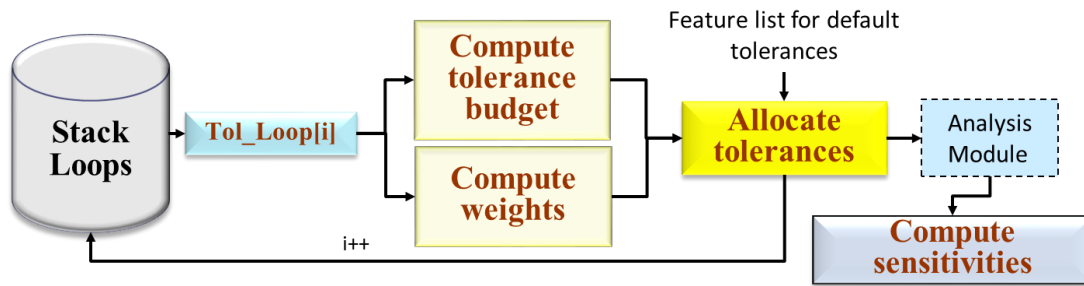


Figure 6- 4: Initial Allocation

Ambiguity arises when common contributors exist between different chains i.e. inter-related stacks, as different stacks might produce different values because of different weights. For such situation Bjørke [6] suggests to use the smallest value of the computed values for a contributor from different stacks. This is a conservative approach to avoid over-prediction of tolerance change.

Another ambiguity is when assigning weight for secondary tolerances, orientation and form. These tolerances are the refinement tolerances i.e. they reduce process deviation for a feature and as a result reduce total stack variability. In current implementation an estimate is used where the assigned weight is fraction of primary tolerance weights and depends on feature's length-diameter ratio ($\frac{l}{d}$ for cylinders) or height-width ($\frac{h}{w}$ for center-planes) ratios. Allocation module ensures the following GD&T conditions are met.

$$\textit{Orientation Tolerance} < \textit{Position Tolerance}$$

$$\textit{Form Tolerance} < \textit{Size Tolerance}$$

$$\textit{Form Tolerance} < \textit{Orientation Tolerance}$$

Non participating features (features not part of any closed stack) are assigned adaptive or user defined tolerance values.

Zero or Negative tolerance budget:

In cases when stacks have interference or no-clearance fit type mating features the tolerance budget computed from previously described method may compute negative tolerance budget. This would tend to allocate negative or zero tolerance values to contributors which has practically no realizable meaning. Therefore, when doing the initial allocation, designer may adopt another method to ensure that positive tolerance values are allocated. One way is to allocate tolerances based on some percentage of the associated dimension, therefore giving no regard to assembly equations and assembly feature clearances. Following this, designer can satisfy these values iteratively based on the tolerance analysis results, so the eventual outcome will be based on the assembly equations and assembly feature clearances. Another way to ensure positive non-zero tolerances are assigned is that, whenever a stack has negative tolerance budget (i.e. interference) or zero tolerance budget, all its contributors may be set to their lowest limit allowed for a dimension. This may be fixed values decided based on industry standards.

6.2.3 Tolerance Re-Allocation

Tolerance allocation is not a one shot deal. They have to be analyzed for assemblability, functionality and manufacturability. If these requirements are not met, tolerances are re-allocated and this process iterates till all targets are satisfied. Typically tolerance allocation should also include an accurate cost model to minimize the machining cost. But it is difficult to automatically predetermine machining processes used for features. Therefore, for now we limit ourselves to satisficing tolerances for the required assemblability rate based on some heuristics and rule based procedure. This heuristic hill climbing algorithm adjusts the tolerances systematically

to achieve assemblability for certain acceptance rate of stack. Shown in Figure 6- 5 is the tolerance reallocation framework.

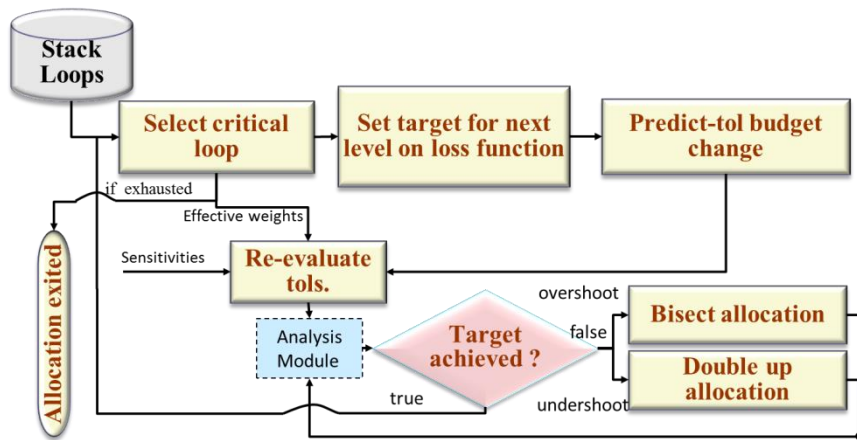


Figure 6- 5: Tolerance Re-allocation Framework

Application of heuristics based iterative re-design concepts for tolerance re-allocation, tolerance distribution method for re-allocation and convergence strategies are discussed below in detail.

Loss Function:

Maintaining tight control over the dimensions of a product may reduce the number of faulty assemblies as the dimensions will tend to be very close to nominal. On the other hand, with tighter dimension control manufacturing processes tend to become exponentially expensive as discussed under section 6.1. Also more liberal dimension control result in lower machining cost, but it may lead to increased rates of the faulty assemblies due to failure to meet required dimensional specifications. Therefore, for an assembly in a production environment assembly scrap rates are usually inversely related to machining cost and both contribute to the total cost of production. Figure 6- 6 shows the usual trend of total cost with tolerances and scrap rates.

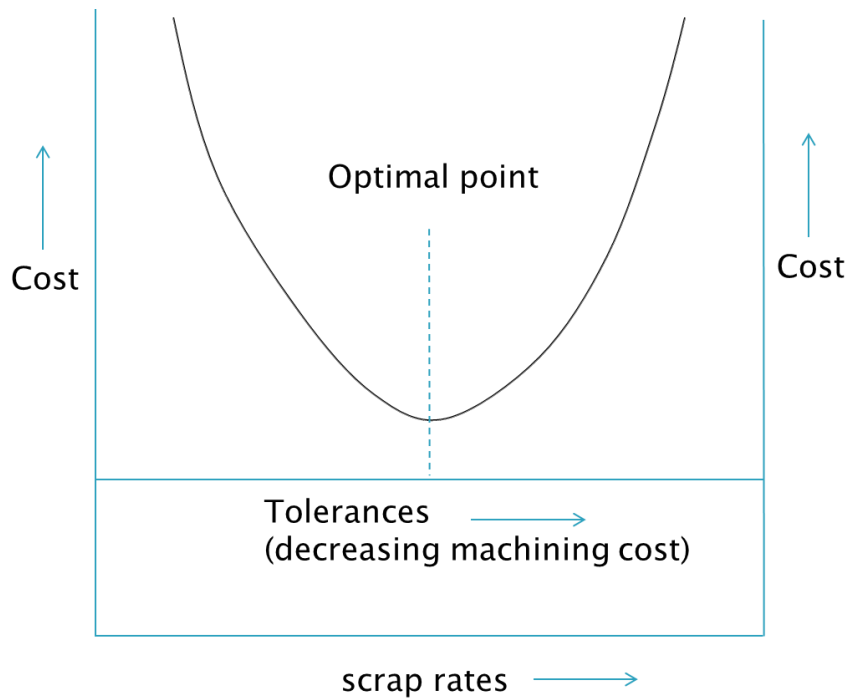


Figure 6- 6: Cost vs Machining Cost and Scrap rates

Thus, tolerances should be adjusted in such a way that the total cost due to machining process and scrapped assemblies are reduced (close to optimal point in Figure 6- 6) while maintaining quality. This is achieved by controlling the acceptance rates of stacks so that some allow higher scrap rates (wider tolerances) and others allow lower scrap rates (narrower tolerances). This can be accomplished by on a loss function basis as shown in Figure 6- 7. Negative side of peak signifies higher scrap rates and wider tolerances (lower machining costs) and the other side signifies narrower tolerances (higher machining cost) consequently reducing scrap rates. Acceptance rates below lower bound are unacceptable as it indicates unacceptably high scrap rates. Acceptance rates above upper bound indicate insignificantly low contribution to overall scrap rate of assembly. Defining an accurate loss function could take some in-depth analysis of manufacturing data. The research here does not

focus on the process of defining a loss function. Instead this research focuses on utilizing a loss function for allocation once defined.

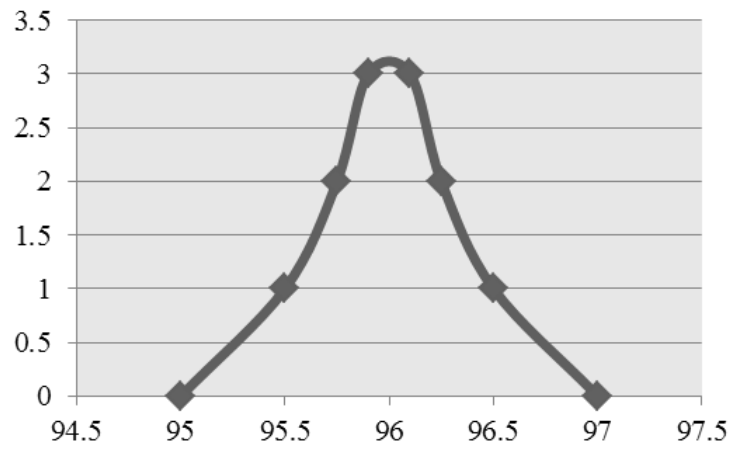


Figure 6- 7: Loss Function

Re-Allocation Method:

Once the loss function is established the acceptance rates of stacks should be equally distributed about the peak dropping off on both sides.

The method proposed here to achieve the distribution of stacks, can be better visualized by simple illustrations. Figure 6- 8 shows the input loss function. Stack acceptance rates are represented by blue points and are distributed over the loss function after initial allocation. Initial allocation though done systematically is based on approximations discussed under 6.2.2. Therefore, it is unlikely that initially allocated values would satisfy the assemblability requirements. In this illustration the figure shows that most stacks after initial allocation have acceptance rates higher than peak. This indicates that there is room to widen tolerances (lowers production cost).

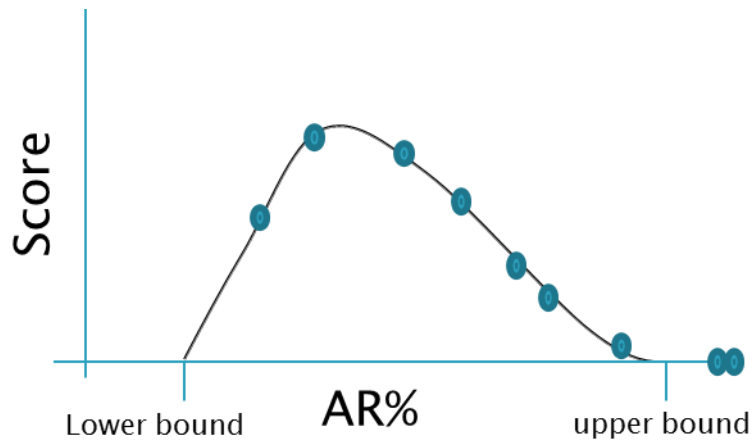


Figure 6- 8: Distribution of Stacks after Initial Allocation

To simplify and speed up the allocation process the function is first discretized into discrete levels as shown in Figure 6- 9. Each step will be treated as a target bracket with same score throughout its range. The convergence method targets these brackets (range of values) instead of one single value resulting in fewer numbers of convergence iterations. This is further discussed under bisection and double up convergence.

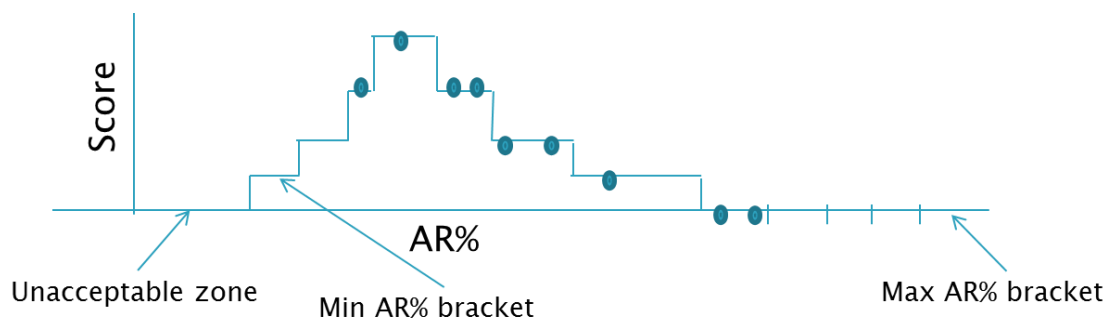


Figure 6- 9: Loss Function Discretized

The re-allocation happens in two phases, 1) identifying the critical loops 2) satisficing these critical loops.

1) Identifying the critical loops-

The first phase pushes the acceptance-rates of stacks towards the lower bound of the loss function by widening the tolerances. If a stack lies in the unacceptable zone, the

algorithm would first bring their acceptance rates above the lower bound of loss function by narrowing tolerance values. Then the algorithm pushes acceptance rates of stacks towards lowest bound. This process exhausts when no further tolerance values could be widened without sending a stack in the unacceptable zone. The objective is to identify the stacks that lie within the loss function range when tolerances are widened while ensuring acceptance rates of all stacks are above lower bound. The acceptance rates of stacks above the upper bound are not critical as they do not have significant contribution to scrap rates. Therefore the stacks lying within the loss function range after first phase are considered critical stacks. In the following phase of allocation these critical stacks can be used to drive the tolerance satisficing process.

Some terms need to be defined before proceeding with the description of the process for first phase of allocation.

Acceptance Rate Brackets – The loss functions is subdivided into discrete set of levels (ranges of acceptance rates). For a single iteration these brackets will be set as target (range of values instead of one value). This will allow faster convergence of solution and will be explained under convergence method in this chapter. The score throughout the range of a bracket is constant.

Effective weights (EW) – In re-allocation process some contributors in the selected stack may not be changed in which case the effective weights of other contributors are computed. Using effective weights for re-allocation ensures that the entire tolerance budget change is utilized for tolerance distribution in every iteration.

$$ew_i = \frac{C_i w_i}{\sqrt{\sum C_i w_i^2}} \quad 6.10$$

Where,

ew_i = Effective weight of i^{th} contributor

w_i = Weight of i^{th} contributor

C_i = Change label of i^{th} contributor

$C_i = 1$, if Tolerance value is to be changed

$C_i = 0$, if Tolerance value is NOT to be changed

Assigning some heuristic scores to the stacks can be useful to rank them. Once the initial tolerance values are assigned, the tolerance analysis tool evaluates the stacks. Based on the acceptance rates and other heuristics, the re-allocation module selects a candidate loop for tolerance re-adjustment. The re-allocation loop selection process is listed below

1. Selects loops from the lowest bracket and puts them through a selection process. It is similar to dependency lists from Dominic instead here a set of tolerances (variables) are getting ranked instead of just one variable. If loops are in the unacceptable zone then select loops from the unacceptable zone first to bring them in the acceptable zone. This stage selects a set of worst conditioned loops to make them as close as possible to the lower bound. This stage ensures that all the stacks are collectively close to the lower bound.
2. Stacks with acceptance rates in the bracket and closest to lower bound are not altered. This holds true unless there is a contributor shared with a stack in

unacceptable zone. Conversely, stack not in the unacceptable zone happen to share contributors with a stack with acceptance rate in the bracket closest to lower bound, then these shared contributors are not altered. Therefore, these shared contributors are designated 0 to their change labels (C_i) as described under effective weights definition. If all the contributors in the selected set of stacks have change labels of 0 then process reselects set of stacks from the next worst conditioned acceptance rate bracket

3. Assign effective weights using equation 6. 10. Weight w_i is computed during initial allocation
4. The target bracket for next iteration will be the neighboring bracket towards the lowest bound
5. Once target acceptance rate bracket is determined the required budget change can be computed for every loop from the selected set of loops. The required budget change can be computed by determining variance required to attain the target (closest bound of target bracket) acceptance rate (AR%). Using standard root finding algorithms like secant method to compute the new required variance (for the AR%). Taking the difference with this required variance with the current variance gives the required variance change. Shown below in equation 6. 11 and equation 6. 12 are tolerance budget change formulations. Since primary tolerances are assumed to be scaled 6 times the standard deviation, the new standard deviation change should be scaled by 6 to compute the tolerance budget change. This tolerance budget change is an approximation and is subject to change for convergence to the target as is discussed below.

if $variance_{new} > variance_{current}$

$$\Delta_{change} = 6\sqrt{variance_{new} - variance_{current}} \quad 6.11$$

if $variance_{current} > variance_{new}$

$$\Delta_{change} = -6\sqrt{variance_{new} - variance_{current}} \quad 6.12$$

Where,

$variance_{current}$ = is the current stack variance

$variance_{new}$ = is the required sum dimension variance for targetAR

Δ_{change} = Required tolerance budget change

Note: The value is scaled by 6 because tolerances are considered to be 6 times the standard deviation values.

6. From equation 6. 13 compute minimum mean tolerance change per contributor (R_{min}) for the set of loops qualified till this stage. Total tolerance value change made to primary contributors divided by number of the primary contributors gives a heuristic measure on effective tolerance budget distributed per contributor. This value is used to rank loops with stacks having smallest non-zero values on top. Computing (R_{min})

$$R_{min} = \min \left\{ \frac{\left(\sum \frac{ew_i}{s_i} \right)_j |\Delta_{change}_j|}{(\text{Number of changeable contributors})_j} \right\} \quad 6.13$$

Where,

j = is stack index of qualified stacks from step 3

i = is primary contributor index in the j^{th} stack

$$\Delta_{change_j} = \text{Required tolerance budget change } j^{th} \text{ stack}$$

Stacks satisfying equation 6. 14 are qualified in this step

$$R_{min} \leq R_i \leq R_{min} + 20\%R_{min} \quad 6. 14$$

Where,

$$R_i = \left(\frac{\left(\frac{\sum e w_i}{s_i} \right)_j |\Delta_{change_i}|}{(\text{Number of changeable contributors})_i} \right), i^{th} \text{ stack}$$

Where,

$j =$ is stack index of qualified stacks from step 3

$i =$ is primary contributor index in the j^{th} stack

$\Delta_{change_j} =$ Required tolerance budget change for j^{th} stack

Here 20% buffer is a default value and can be defined by end user

7. If multiple loops passed till this stage, select the stack with acceptance rate closest to the target acceptance rate bracket.

This heuristic based selection process identifies the stack which requires smallest mean tolerance change. The identified stack performance parameters are sent to the next bracket in this 1st phase of re-allocation. This loop selection strategy aims to achieve evenly distributed tolerance allocation.

The tolerance budget change computed from equations 6. 11 or 6. 12 for selected loop are distributed over the contributors using equation 6. 15, 6. 16 and 6. 17

If, Lower Limit $< T_{i_{current}} + \left(\frac{e w_i}{s_i} \Delta_{change_i} \right) <$ upper limit, then

$$T_{i_{new}} = T_{i_{current}} + \left(\frac{ew_i}{s_i} \Delta_{change_i} \right) \quad 6.15$$

If, $T_{i_{current}} + \left(\frac{ew_i}{s_i} \Delta_{change_i} \right) > \text{upper limit}$, then

$$T_{i_{new}} = \text{upper limit} \quad 6.16$$

If, $T_{i_{current}} + \left(\frac{ew_i}{s_i} \Delta_{change_i} \right) < \text{Lower limit}$, then

$$T_{i_{new}} = \text{Lower limit} \quad 6.17$$

Here,

$T_{i_{current}} = \text{current tolerance value}$

$T_{i_{new}} = \text{new tolerance value}$

$ew_i = \text{effective weight from equation 6.10}$

$s_i = \text{sensitivity (computed by tolerance analysis module)}$

$\text{upper_limit} = \text{max allowable tolerance value}$

$\text{lower_limit} = \text{min allowable tolerance value}$

The above loop selection process and tolerance allocation is continued till: a) all stacks are in the lowest bracket or b) acceptance rates cannot be brought any closer to the lower bound of loss function without sending a loop in the unacceptable zone. After initial allocation the new distribution may look like Figure 6-10 (b).

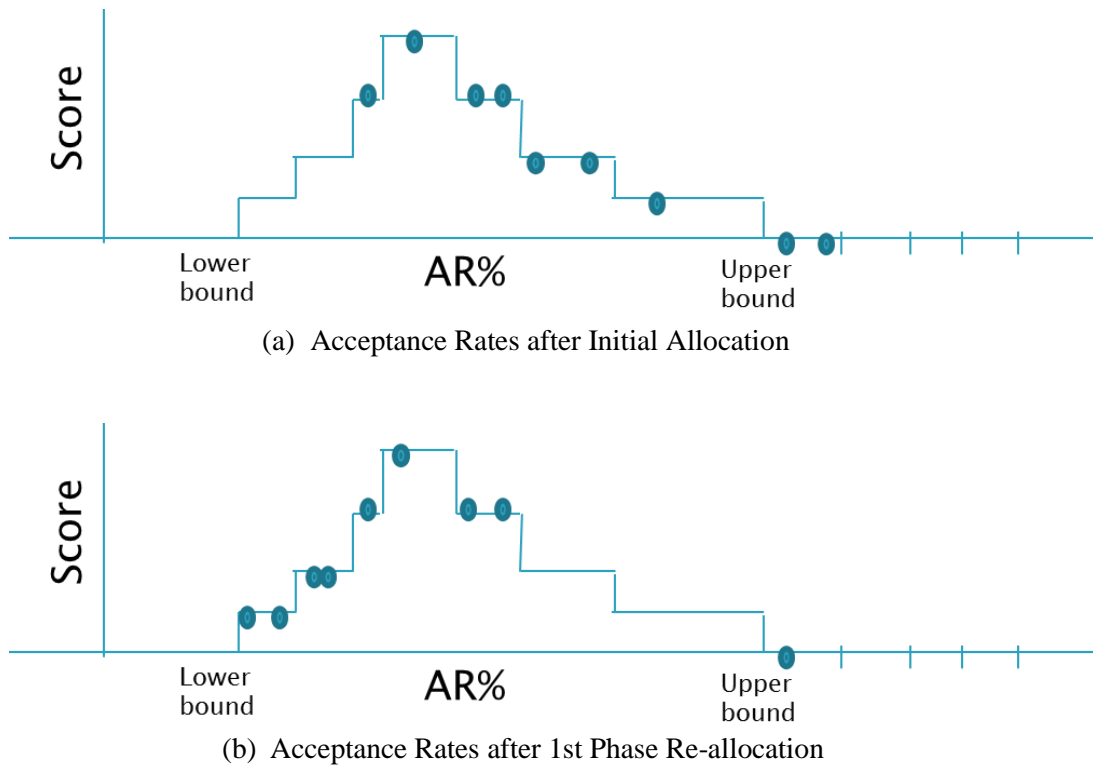


Figure 6- 10: Acceptance Rates Distribution After 1st Phase of Re-Allocation

The stacks lying within the loss function bound are considered critical stacks. After 1st phase these critical stacks are used to drive the tolerance satisficing based on loss function, though all of the stacks are evaluated by analysis module after every iteration.

2) Satisfice Tolerances Using Critical Loops-

This phase of allocation aims at achieving tolerance values to reduce the overall cost of production based on the definition of loss function. The acceptance rates of critical stacks should be equally distributed about the peak on both sides. As already established negative side of the peak signifies wider tolerances and higher scrap rates while the other side signifies tighter tolerances and lower scrap rates. Overall the stacks should be as close to the peak as possible, as liberalizing narrower tolerances (tolerances stacks on right side) of peak gives larger gains in cost (section 6.1). Wider

tolerances on left side would give higher gains from reduced losses on assemblies. To measure the balance about the loss function the total score should be equal on both sides. The balance of score can be computed from the equation 6. 18. Value closer to 0 indicates equal distribution on both sides of the peak.

$$\sum_i^n \{(-1)^a \cdot Score_i\} \quad 6. 18$$

Where,

a = 1, for stacks with acceptance rates on right side of peak

a = 0, for stacks with acceptance rates on left side of peak

The loop selection process for tolerance satisficing phase is explained below

1. Select set of critical stacks with acceptance rates in the lowest bracket on the high scrap rate side of the peak to ensure that all stacks are collectively balanced
2. Target acceptance rate bracket is the neighboring bracket towards higher acceptance rate
3. Compute required tolerance budget change for each loop using equation 6. 11 or 6. 12 for the selected set of loops till this stage.
4. From the set of loop the one with smallest mean tolerance change per contributor R_{\min} (equation 6. 13) is selected for re-allocation.

Once a loop has been selected tolerances can be allocated using the equations below

If, Lower Limit $< T_{i_{current}} + \left(\frac{w_i}{s_i} \Delta_{change_i}\right) <$ Upper limit, then

$$T_{i_{new}} = T_{i_{current}} + \left(\frac{w_i}{s_i} \Delta_{change_i} \right) \quad 6.19$$

If, $T_{i_{current}} + \left(\frac{w_i}{s_i} \Delta_{change_i} \right) > \text{Upper limit}$, then

$$T_{i_{new}} = \text{upper limit} \quad 6.20$$

If, $T_{i_{current}} + \left(\frac{w_i}{s_i} \Delta_{change_i} \right) < \text{Lower limit}$, then

$$T_{i_{new}} = \text{Lower limit} \quad 6.21$$

Here,

$T_{i_{current}} = \text{current tolerance value}$

$T_{i_{new}} = \text{new tolerance value}$

$w_i = \text{weight}$

$s_i = \text{sensitivity}(\text{computed by tolerance analysis module})$

$\text{upper_limit} = \text{max allowable tolerance value}$

$\text{lower_limit} = \text{min allowable tolerance value}$

The iterations continue till 0 or minimum balance (equation 6. 18) has been achieved.

Finally the distribution of acceptance rates may look like Figure 6- 11 with scores

balanced on both sides of the peak.

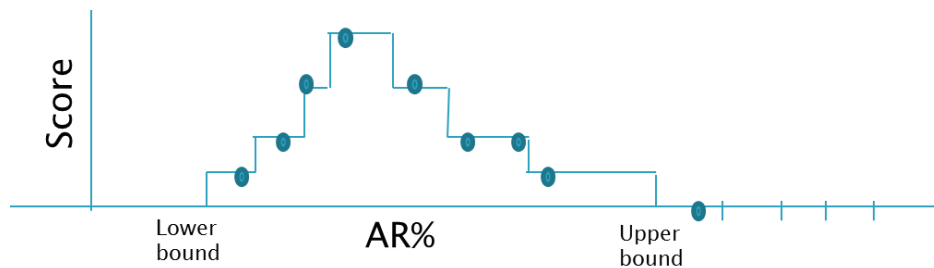


Figure 6- 11: Acceptance Rate Distribution after Satisficing of Tolerances

Bisection and Double up for Convergence:

The tolerance allocation method described above is based on approximations and may give deviated results than expected and therefore we need some convergence method. For auto-tolerancing the bisection and double up allocation methods (Figure 6- 5) help achieve convergence. After allocation if the acceptance rate of a stack overshoots from the target bracket, the allocation is bisected i.e. the change added to the tolerances from previous iteration are halved. The bisection continues till the acceptance rate of stack reaches the undershoot zone or neighborhood of target. On the other hand if stack acceptance rate undershoots the required acceptance rate, the allocation is doubled up i.e. the tolerance change from previous iteration is doubled up. The tolerance is doubled up till the allocation is in overshoot zone or converges to the target bracket. The bisection and double-up methods may be called alternately depending on undershoot or overshoot. In this situation if the change from last bisection was say Δt for a contributor i.e Δt was subtracted from a contributor then for double up this value is halved and added to the contributor. Similarly if bisection is invoked after double up that sent the stack in overshoot zone, the tolerance change Δt of a contributor from the previous double up method is halved and subtracted. This sidesteps the oscillation of solution indefinitely about a target. This happens simultaneously to all contributors that undergo change in the selected stack. It is also important to put a lower and upper limit on tolerance values based on feature size or some fixed values defined by user. This would avoid assigning impractical values i.e. close to zero values or unnecessarily high values. To avoid going into long convergence iterations the discretization of loss function should be of reasonably low

resolution (subdivided into wide discrete levels). Then during convergence cycle if the value reaches the target bracket, the solution will be considered converged. Higher resolution discretization will take more number of cycles to converge to the target bracket and therefore should be avoided.


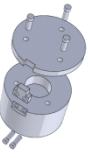
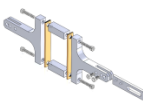
CHAPTER 7

TEST CASES

The allocation/verification method described in this research is the 2nd primary tool in the auto-tolerancing system. But this tool is not limited to this system, it may be used as a separate tool where user can input schema and allocation /verification can generate the values. It can be easily customized to give more user control to modify tolerance values. This has a potential to reduce the product dimension specification time significantly and the overall product development time.

It has been successfully tested with different assembly types simulated for at least 2000 samples per loop for tolerance analysis. Table 7- 1 shows the performance chart of the tolerance allocation and analysis module for 3 test cases. All the three test cases were allocated based on a user defined loss function defined within the bounds of 95%-99% acceptance rates.

Table 7- 1: Tolerance Analysis Verification Test Case Time Statistics

	Loop detection		Allocation/Verification		
	Loops Detected	Time (Seconds)	No. of tolerances	Iterations	Time (Seconds)
	182	1.203	62	46	312.988
	152	0.826	57	30	161.599
	27	0.1	97	60	17.693

The time consumed by re-allocation and initial allocation was relatively negligible. Most of the time was consumed by tolerance analysis module. The final

output of the tolerance allocation and analysis module for the three assemblies from Table 7- 1 are presented below.

7.1 Cam Follower

Figure 7- 1 shows the Cam Follower assembly (3rd assembly in Table 7- 1) with 13 parts and was provided by our industry partner RECON SERVICES. User input loss function upper and lower bound for this test case is given as 95%-99%.

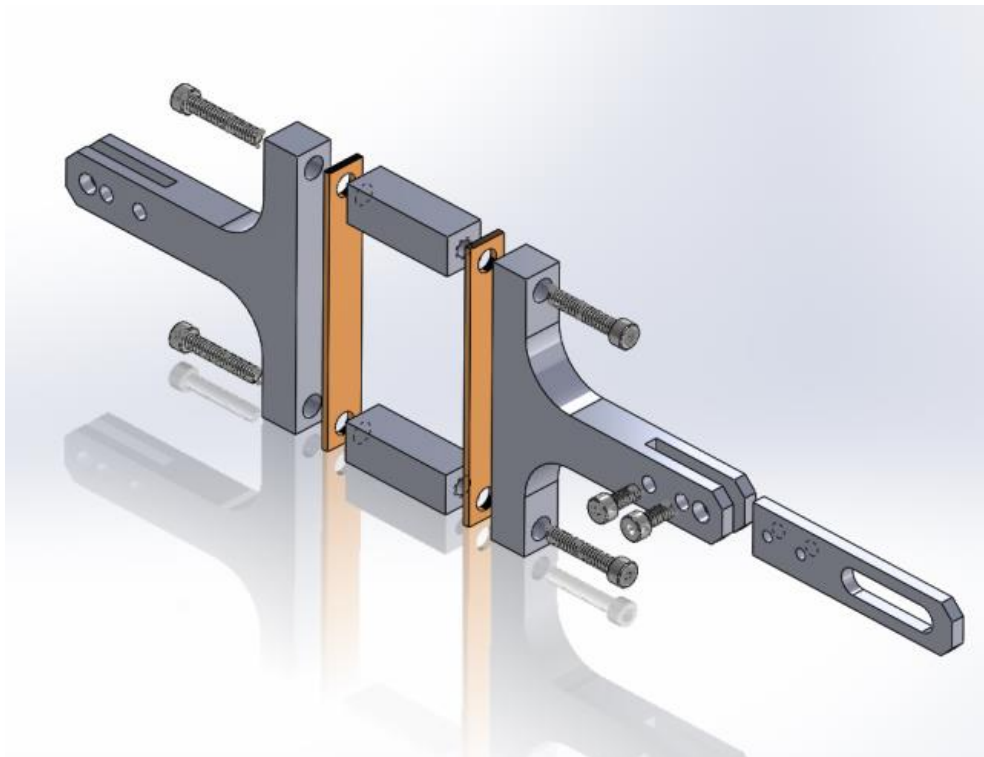


Figure 7- 1: Cam Follower Assembly in Exploded View

Figure 7- 2 shows the auto-tolerancing STEP AP242 output for the right support of cam follower and the washer plate on MBDVidia CAD viewer. Total of 9 loops identified critical out of 27 during the first phase of re-allocation (identifys critical stacks). Figure 7- 3, shows the final acceptance rates of the critical loops distributed based on the input loss function between 95%-99%. After the 2nd phase of re-

allocation the balance score achieved was 0.12 which is close to zero. Therefore it is balanced on both sides about the optimum value of the loss function.

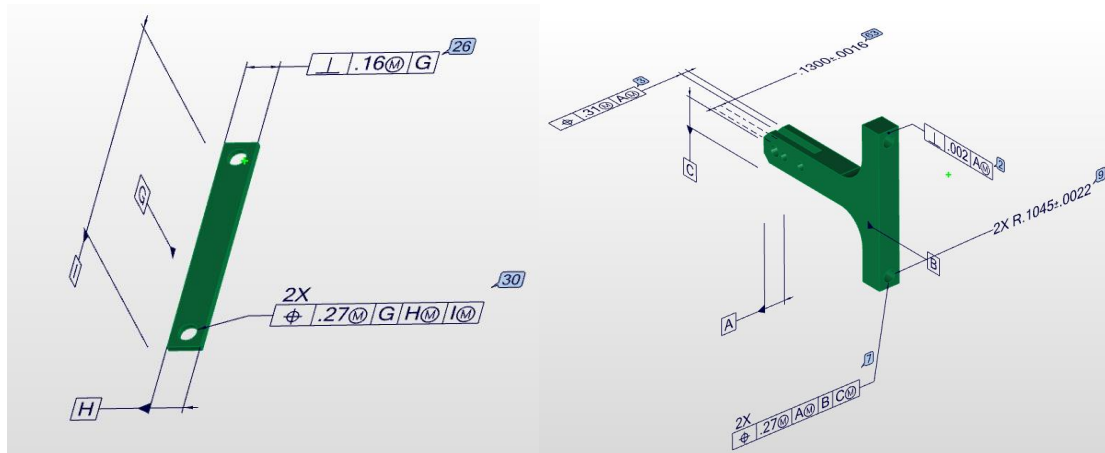


Figure 7- 2: Automated GD&T Output for Right Support of Cam Follower

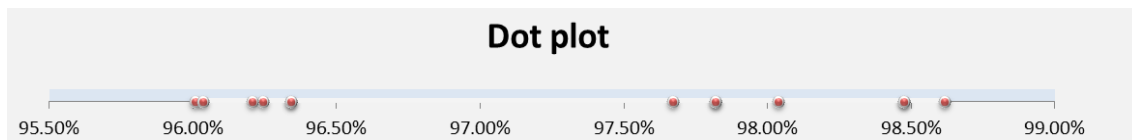


Figure 7- 3: Final distribution for Critical Stacks of Cam Follower

7.2 Radio Car

Figure 7- 4 shows the Radio Car assembly (1st assembly in Table 7- 1) with 9 parts and was provided by DARPA under the AVM program (figure 4). User input loss function upper and lower bound for this test case is given as 95%-99%.



Figure 7- 4: Radio Car Assembly Exploded View

Figure 7- 5 shows the auto-tolerancing STEP AP242 output for the radio car chassis and the cross beam on MBDVidia CAD viewer. Total of 39 loops identified critical out of 182 during the first phase of re-allocation (identifying critical stacks). All tolerances Figure 7- 6, shows the final acceptance rates of the critical loops distributed based on the input loss function between 95%-99%. After the 2nd phase of re-allocation (distribute about the loss function mean) balance score achieved was 0.08 which is close to zero. Therefore it is evenly distributed.

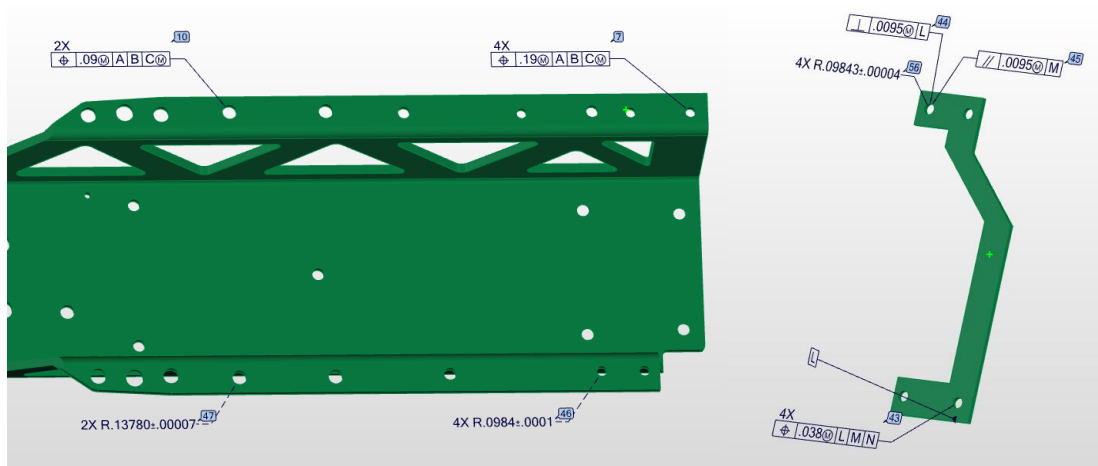


Figure 7- 5: Automated GD&T Output for Radio Car Chassis and Rear Cross Beam

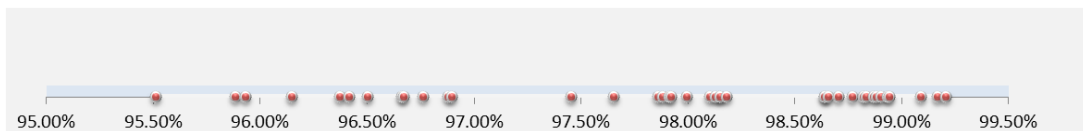


Figure 7- 6: Final distribution for Critical Stacks of Radio Car Assembly

7.3 Cylinder Body Cap

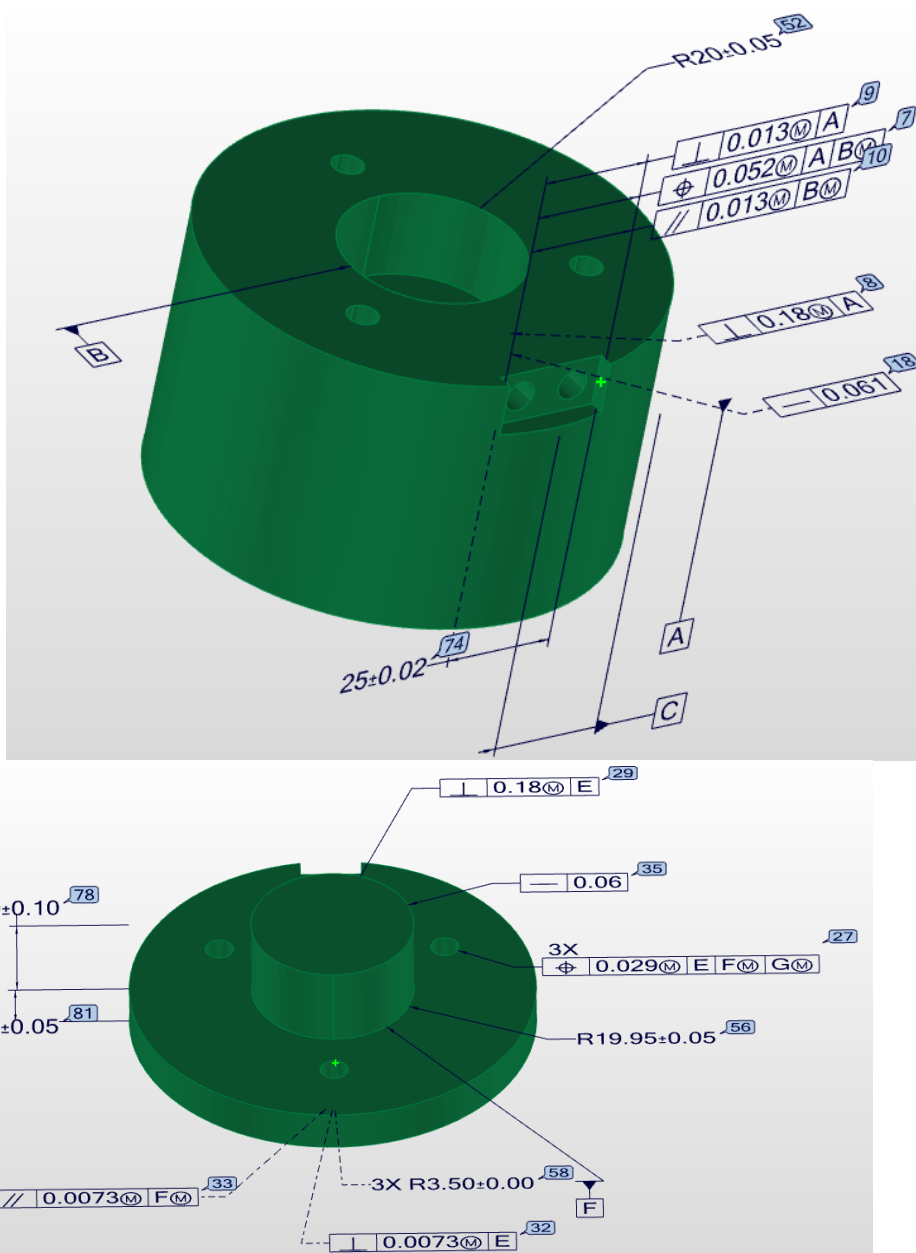


Figure 7- 8 shows the Cylinder Body Cap assembly (2nd assembly in Table 7- 1) with 8 parts and was provided by DARPA under the AVM program. User input loss function upper and lower bound for this test case is given as 95%-99%.

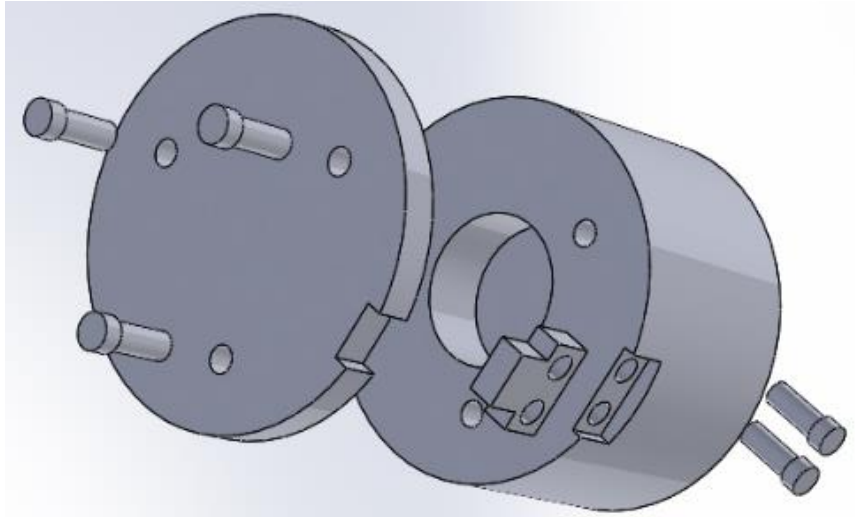


Figure 7- 7: Body Cap Assembly Exploded View

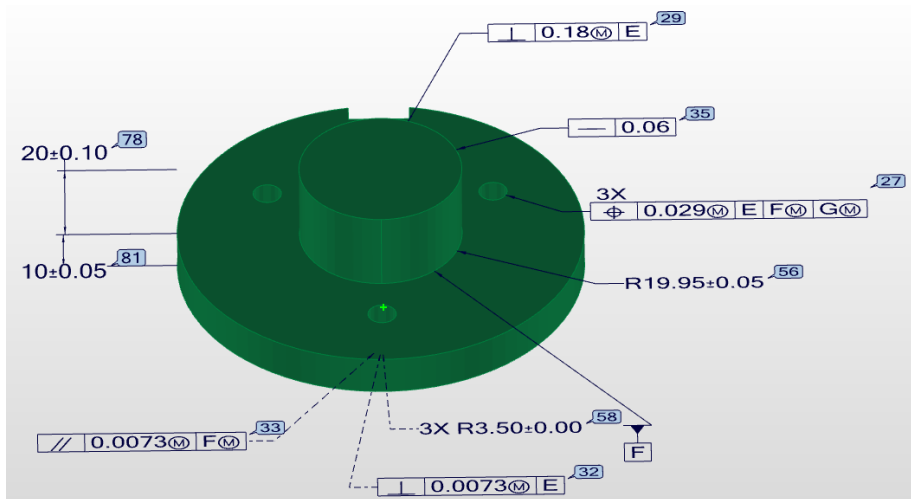
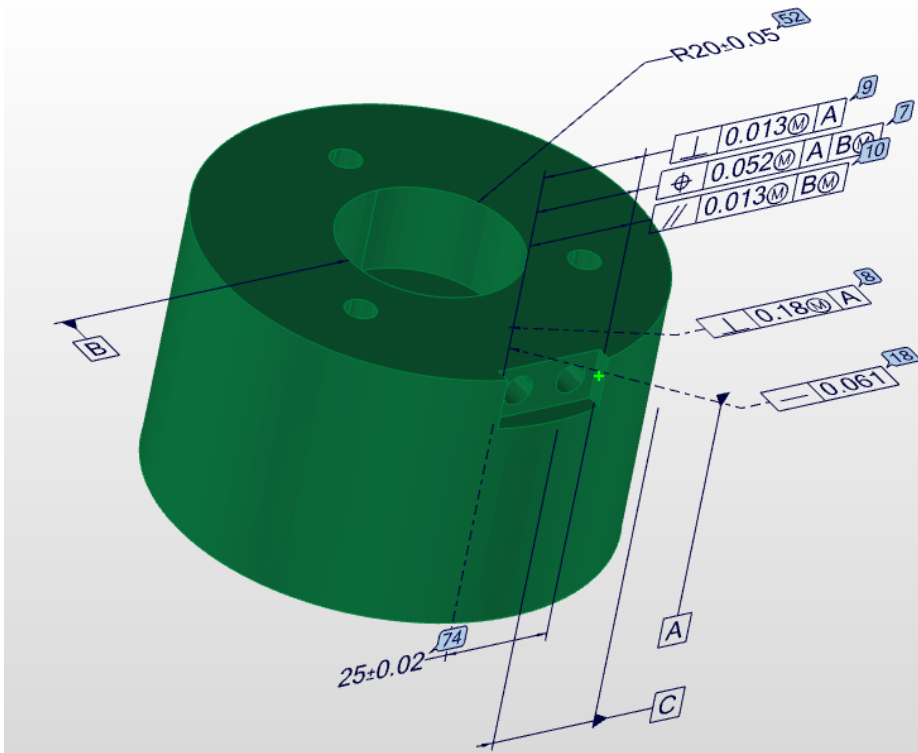


Figure 7- 8 shows the auto-tolerancing STEP AP242 output for the radio car chassis and the cross beam on MBDVidia CAD viewer. Total of 26 loops identified critical out of 152 during the first phase of re-allocation (identifying critical stacks). Figure 7- 9, shows the final acceptance rates of the critical loops distributed based on the input loss function between 95%-99%. This is a special case where most of the

acceptance rates lay on right side of optimum value of the loss function after first phase of re-allocation. Therefore, the re-allocation terminates after the first phase.

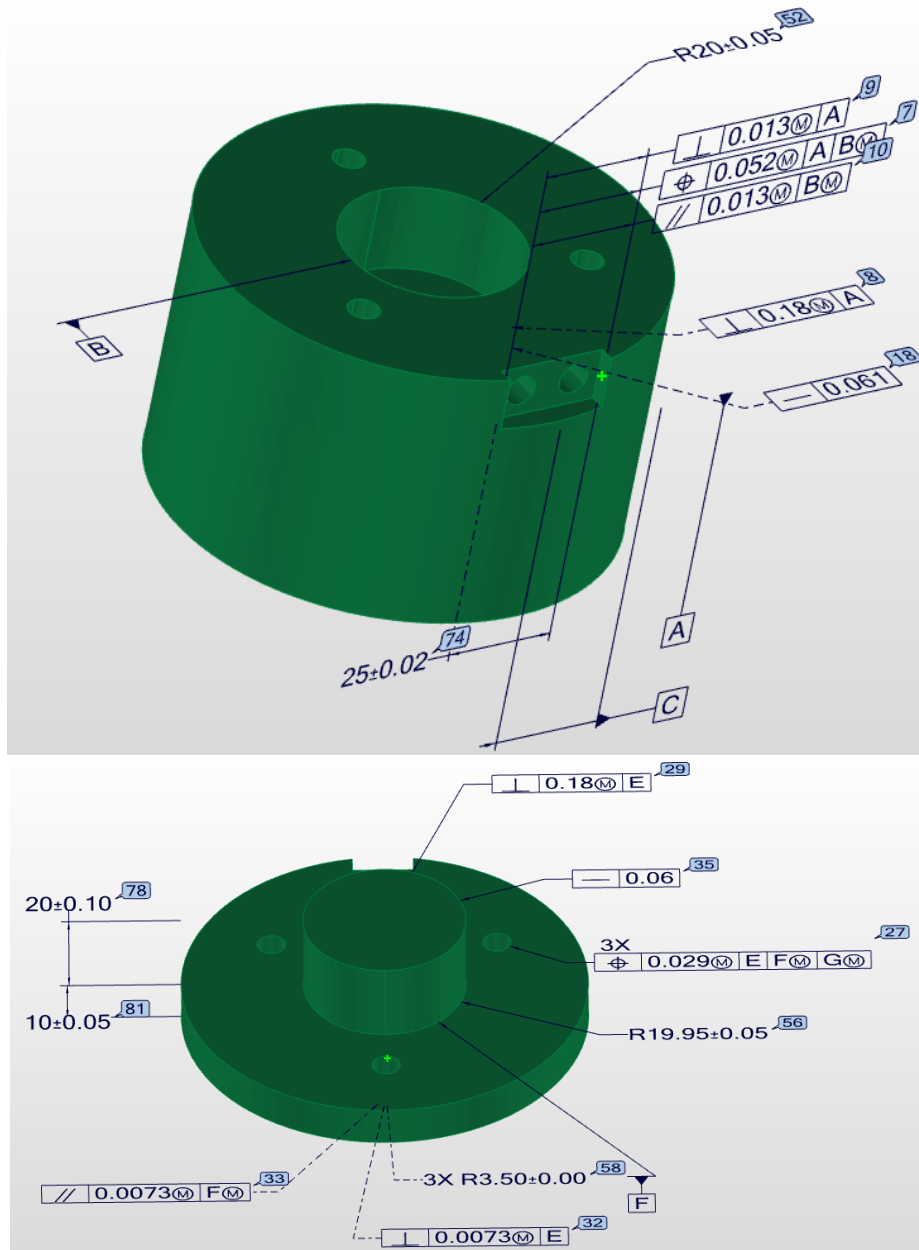


Figure 7- 8: Automated GD&T Output for Cylinder Body and Cap

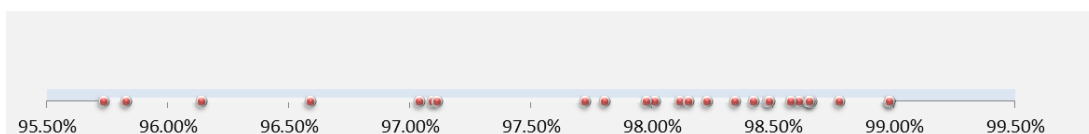


Figure 7- 9: Final Distribution for Critical Stacks of Cylinder Body Cap Assembly

CHAPTER 8

CLOSURE

This chapter summarizes the research work presented in this thesis. It summarizes the contributions, lists out the limitations and discusses possible future works.

8.1 Contributions

The research work presented here has made following contributions:

1. It suggests a scalable and a flexible tolerance allocation and analysis framework for auto-tolerancing. It incorporates the loop detection module, tolerance allocation module and tolerance analysis modules. All the three modules are designed as standalone modules. Modifications can be incorporated easily without affecting the other modules. It will allow different loop detection, tolerance analysis method or tolerance allocation method to be easily adopted without modifying other modules. This design will also allow the modules to be used independently to develop other tools in the future.
2. Describes in detail the loop detection module developed to detect closed chains of tolerance stacks. Identified automatic stack extraction as a graph problem. The graph nodes represent tolerance features or datums and edges represent the geometric dimension or mating constraints. The loop detection uses Breadth First (level order) search to detect all assembly level stack between two mating features.
3. Evaluates the different tolerance analysis methods and discusses their suitability for the auto-tolerancing system. 3D Parametric variation analysis

seemed to be suitable choice and can be easily adapted for the system. 3D Parametric variation analysis does not require exclusive computation of sensitivities required for the purpose of tolerance analysis. It's a more consistent method to conduct tolerance analysis as it varies features in 3 dimensional tolerance zones. It also captures the effect of higher order terms from Taylor's series unlike linearized root sum square. Under tolerance analysis we also discuss the Euler's approach to estimate the sensitivities (first order derivatives).

4. Suggests a method for iterative tolerance allocation for a target acceptance rate that can be used by designers. Introduces the concept of using loss function for tolerance satisficing. One side of loss function about optimum mean signifies tighter tolerance and higher acceptance rates and the other side signifies wider tolerance lower acceptance rates. Thesis proposes a method and software architecture to automate an iterative tolerance allocation method based on the loss function. The allocation happens in two phases. Phase 1 identifies the critical loops. These loops lie within the loss function bounds after the phase 1. Phase 2 satisfices the tolerance allocation by evenly distributing the acceptance rates about the optimum mean.

8.2 Limitations

The limitations of the research presented here are as follows:

1. Toleranced features are limited to cylindrical features, Center plane type features and planar features. Currently the auto-tolerancing system cannot handle free form surfaces and other complex machining surfaces.

2. Limited to tolerancing based on assemblability. Current auto-tolerancing system does not account for functionality and manufacturing cost.

8.3 Future Work

1. Future work in allocation/verification would explore scalability of the tool. Explore multithreading for the tolerance analysis process.
2. Integrate different tolerance analysis methods to improve the accuracy or reduce the computational cost
3. Attempt tolerancing based on functional requirements and manufacturing cost of assemblies which may require more user involvement.
4. Extend the feature library for assembly feature recognizer. Extend the feature variation algorithm library for tolerance analysis module to other features.
5. Addressing the issue of computing overall acceptance rate of an assembly from acceptance rates of stacks. This is not a trivial problem as often tolerance stacks are inter-related and have complex dependence with shared section of stack. Assuming 2 inter-related stacks assemble as event A and event B. The probability that they assemble together can be evaluated by finding the intersection. Here computing the intersection of the events is not a trivial problem. One approach could be to design a parametric variation method that directly incorporates the effect of inter-relation of stacks and computes the overall probability.

REFERENCES

1. ASME, 1994, "Dimensioning and Tolerancing, ASME Y14.5M-1994", American Society of Mechanical Engineers, New York.
2. ISO 1101, 2012, "Geometrical product specifications (Gps) - Geometrical tolerancing - Tolerances of form, orientation, location and run-out", ISO, 2004.
3. Dixon JR, Howe A, Cohen PR, Simmons MK. Dominic 1: Progress toward domain independence in design by iterative redesign. *Engineering with Computers*, 1987, Volume 2, Number 3, Page 137
4. Mohan P, Haghghi P, Vemulapalli P, Kalish N, Shah JJ, Davidson JK. Toward Automatic Tolerancing of Mechanical Assemblies: Assembly Analyses. *ASME. J. Comput. Inf. Sci. Eng.* 2014;14(4):041009-041009-14. doi:10.1115/1.4028592.
5. Haghghi P, Mohan P, Kalish N, Vemulapalli P, Shah JJ, Davidson JK. Toward Automatic Tolerancing of Mechanical Assemblies: First-Order GD&T Schema Development and Tolerance Allocation. *ASME. J. Comput. Inf. Sci. Eng.* 2015;15(4):041003-041003-9. doi:10.1115/1.4030939.
6. Bjørke Ø, *Computer-Aided Tolerancing*, 2nd Edition, ASME Press 1989.
7. Shen Z, Shah JJ, Davidson JK. Analysis neutral data structure for GD&T. *Journal of Intelligent Manufacturing*, 2008, Volume 19, Number 4, Page 455
8. Speckhart FH. Calculation of Tolerance Based on a Minimum Cost Approach. *ASME. J. Eng. Ind.* 1972;94(2):447-453. doi:10.1115/1.3428175.
9. Lai D., Yuen M., 2011, "Vector based Datum Transformation scheme for computer aided measurement", *Computer-Aided Design & Applications*, Vol. 8.
10. Lee, W.J. and Woo, T.C., "Tolerancing: Its Distribution, Analysis and Synthesis," Department of Industrial and Operations Engineering, The University of Michigan, Aug., 1987b
11. Bowman R. Efficient Gradient-Based Tolerance Optimization Using Monte Carlo Simulation. *ASME. J. Manuf. Sci. Eng.* 2009;131(3):031005-031005-8. doi:10.1115/1.3123328.
12. Chase K., 1999, "Multi-Dimensional Tolerance analysis", Ch. 13, *Dimensioning and Tolerancing Handbook*
13. Vemulapalli, Prabhat, Prashant Mohan, Jami J. Shah and Joseph K. Davidson, "User Defined Assembly Features and Pattern Recognition from STEP

- AP203”, In ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. V01AT02A067-V01AT02A067. American Society of Mechanical Engineers, 2014
14. Shen Z, Shah JJ, Davidson JK. Automatic generation of min/max tolerance charts for tolerance analysis from CAD models. *Computer Integrated Manufacturing*, 2008, Volume 28, Issue 8, Page 869
 15. Davidson JK, Mujezinović AA, Shah JJ. A New Mathematical Model for Geometric Tolerances as Applied to Round Faces. *ASME. J. Mech. Des.* 2002;124(4):609-622. doi:10.1115/1.1497362.
 16. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NPCompleteness*. W. H. Freeman and Company, New York, 1979.
 17. Vemulapalli, P., Mohan, P., Shah, J. J., and Davidson, J. K., 2014, “User Defined Assembly Features and Pattern Recognition From STEP AP203,” ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. V01AT02A067–V01AT02A067.
 18. Shen, Z., Shah, J. J., and Davidson, J. K., 2005A, “Simulation-Based Tolerance and Assemblability Analyses of Assemblies with Multiple Pin-Hole Floating Mating Conditions,” accepted by DETC’05, ASME 2005 DETC/CIE Conf. , Long Beach, CA, 25–28 Sept., 2005.
 19. Shen, Z., Shah, J. J., and Davidson, J. K., 2005B, “A Complete Variation Algorithm for Slot and Tab Features for 3D Simulation-Based Tolerance Analysis,” accepted by DETC’05, ASME 2005 DETC/DAC Conf. , Long Beach, CA, 25–28 Sept., 2005.
 20. Davidson, J. K., Mujezinović, A., and Shah, J. J., 2002, “A New Mathematical Model for Geometric Tolerances as Applied to Round Faces,” *ASME J. Mech. Des.* [CrossRef], 124 , pp. 609–621.
 21. Mujezinović, A., Davidson, J. K., and Shah, J. J., 2004, “A New Mathematical Model for Geometric Tolerances as Applied to Polygonal Faces,” *ASME J. Mech. Des.* [CrossRef], 126 (3), pp. 504–518.

22. Prisco, U., and Giorleo, G., 2002, "Overview of Current CAT Systems," *Comput.-Aided Eng. J.*, 9, pp. 373–387.
23. Chiesi, F., and Governi, L., 2003, "Software Review-Tolerance analysis with eTol-Mate," *ASME J. Comput. Inf. Sci. Eng.*, special issue on GD&T, 31, pp. 100–105.
24. Shen, Z., 2003, "Software Review-Tolerance analysis with EDS/VisVSA," *ASME J. Comput. Inf. Sci. Eng.*, special issue on GD&T, 31, pp. 95–99.