J Intell Manuf (2008) 19:455–472
DOI 10.1007/s10845-008-0096-2

Analysis neutral data structure for GD&T

Zhengshu Shen · Jami J. Shah · Joseph K. Davidson

Received: 18 October 2007 / Accepted: 11 February 2008 / Published online: 19 March 2008

Abstract The 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. Furthermore, such a representation model should facilitate GD&T validation and tolerance analysis. Most GD&T representation models proposed so far are specific to the tolerance analysis method. Common tolerance analysis methods are min/max chart, Monte Carlo simulation and multivariate regions. This paper will propose a semantic GD&T model, which can be used for any of these methods. The model is a super constraint-tolerance-feature-graph (SCTF-Graph). This paper will demonstrate how the SCTF-Graph model can represent all the tolerance types in the standards, and can contain all the information that is needed for tolerance analysis: nominal geometry (i.e. trimmed features in this research), constraints, tolerances, degrees of freedom (DoFs) to be controlled, assembly hierarchy, and their respective inter-relationships. This paper will discuss the content of the model, how it can be automatically created from the CAD model containing GD&T information (e.g. attributed B-Rep model), and the implementation of such a model, along with some case studies.

Z. Shen
NX Part Modeling, UGS PLM Software, Siemens Automation and Drives, 10824 Hope Street, Cypress, CA 90630, USA
e-mail: Zhengshu.Shen@siemens.com

J. J. Shah (✉) · J. K. Davidson
Design Automation Lab, Department of Mechanical and Aerospace Engineering, Ira A. Fulton School of Engineering, Arizona State University, Tempe, AZ, 85287, USA
e-mail: Jami.Shah@asu.edu

J. K. Davidson
e-mail: J.Davidson@asu.edu

Keywords GD&T · Tolerance modeling · Data modeling

Introduction

Most CAD systems support GD&T specification via attributes attached to the CAD entities. It is easy and convenient to implement. One such attributed model is *the ASU GD&T Global Model* (Wu 2002; Wu et al. 2003), where the different types of dimensional and geometric information are attached to the corresponding geometric entities like face, edge, etc. Such attributed CAD models are not suitable for direct use in developing automated tolerance analysis tools because it is very inefficient to directly manipulate the corresponding CAD model or database for tolerance analysis. They are especially not appropriate for simulation based tolerance analysis (Shen et al. 2005a,b,c), because direct variation of the CAD model is very computationally expensive. Furthermore, they do not facilitate tolerance chain detection, because the GD&T data are saved as attributes that are attached to the geometric entities and traversing the CAD model and its entity attributes to extract the tolerance chain is not efficient either. Therefore, it is necessary to create a separate model for tolerance analysis. Another advantage is that driving all tolerance analysis tools with the same independent (or neutral) model provides a uniform basis for comparison of all the approaches to tolerance analysis and optimization. In our lab we have derived tolerance analysis tools based on all three methods, i.e. min/max chart, Monte Carlo simulation and multivariate regions; the motivation for a semantic GD&T model was to make the tolerance specification independent of the tolerance analysis method.

To summarize the scope, this paper is to propose a neutral data structure for GD&T, and the model should be accurate and complete in terms of GD&T definitions, and independent

of any specific tolerance analysis method. In the following, we will briefly review current modeling state of GD&T, and describe the basic contents of the proposed the model, and then discuss the data structure, and implementation of the model. Some case study is provided at the end of the paper.

Review

This work assumes that the model to be developed is in conjunction with mainstream commercial CAD systems, and GD&T specifications are in accordance with ASME and ISO Y14.5 standards (ASME 1994). Therefore, we first give a quick overview of CAD data models and the tolerance standard.

CAD models

3D CAD systems today represent parts as BRep (Boundary Representation) with history and parametrics. Two levels of parametrics are supported: (1) 2D parameters, associated with sketches (geometric constraints between lines, arcs, etc.; algebraic relations between dimensional parameters); (2) 3D parameters, associated with 3D construction operations (sweeps, lofts, shells, fillets, blends, etc.). The history tree records the procedure used for creating an object. Editing a model involves rolling back to the point of change in the history, making the change and rolling forward. Changes to the sketch composition of parameters require a constraint solver to recalculate all the dimensions. Changes to 3D parameters are usually directly assignment or sequential parameter calculation without the need for a solver. Thus, 3D CAD systems of today are hybrid in the sense that the representation is partly explicit (BRep and parametrics) and partly procedural (history). The BRep data contains topology and geometry; parametrics contains constraints and explicit dimensions; history contains the sequence of construction operations in the form of a binary tree.

CAD also supports assemblies—collection of functionally related parts that need to be put together in particular way to realize the function of the device. An assembly model contains: hierarchical relationships (components, sub-assemblies, assemblies), assembly relations (mating conditions, assembly level constraints), and components/sub-assembly positions (global or relative).

GD&T standard classes

The standards have classified dimensional variations (size) and geometric variations (form, orientation, profile, position, and runout) into six separate classes (ASME 1994). This is because of the types of variation that need to be controlled depends on functional and assembly requirements. Size tolerances are specified by a $\pm$ variation while geometric tolerance classes imply an allowable region (zone). The shape of the zone depends on the tolerance type and the feature being toleranced; the size of the zone depends on the tolerance value, material condition modifier, and certain rules; the position/orientation of the zone depends on the tolerance type and datums. Datums are references for measurements; they are neither on the part nor on the gage, but simulated by the contact between the two; all D&T relations (except size) are one-dimensional, i.e. datum-to-target. When multiple datums are used, the order in which they are specified creates a precedence order used to determine the co-ordinates and the directions of control. Certain tolerances can be applied to revolved entities (e.g. axes, midplanes), while others only to boundary elements (e.g. faces, edges), and some to either. Material conditions modifiers (MMC, LMC, and RFS) are used to indicate what the geometric tolerance is when the size is at its largest or smallest value. When the feature size deviates from that value, a "bonus tolerance" is added to the geometric tolerance; when a modifier is applied to a datum feature of size, the geometric tolerance zones "shift" which is equivalent to a larger zone.

The standard was not based on any mathematical model but it contains valuable experiential knowledge collected from decades of engineering practice. This lack of a mathematical basis for Y14.5 causes ambiguities in interpretation of the standard, making it challenging to create semantic data models that can be manipulated for tolerance analysis.

GD&T models

We have classified GD&T models into six major categories discussed below.

Attribute models: The basic characteristic of attribute models is that a tolerance is directly stored as an attribute of either geometric entities or metric relations in CAD systems (Johnson 1985; Ranyak et al. 1988; Shah et al. 1990; Roy et al. 1993, 1996; Maeda et al. 1995; Tsai et al. 1997). The common deficiency of this approach is that they cannot do validation since GD&T semantics is not built into the model structure.

Offset models: In this approach, the maximal and minimal object volumes are obtained by offsetting the object by corresponding amounts on either side of the nominal boundary (Requicha 1983; Jayaraman et al. 1989). Offset models can only represent a composite tolerance zone; they cannot distinguish between effects of different tolerance types, nor interrelations among tolerance specifications.

Parametric models: Tolerances are modeled as plus/minus variations of dimensional or shape parameters. Parameter values can be found by a set of simultaneous equations representing the constraints (Hillyard et al. 1978; Krishnan et al.

1997; Turner 1993). The parametric equations can be used for point-to-point tolerance analysis rather than zone based analysis. This is not consistent with GD&T standards.

Kinematic models: Entities are modeled in terms of "virtual" links and joints. A "kinematic link" is used between a tolerance zone and its datum features (Rivest et al. 1993; Desrochers et al. 1997; Chase et al. 2000). Tolerance analysis is based on vector additions. The first order partial derivative of analyzed dimension with respect to its component dimensions in terms of a transformation matrix was employed for tolerance analysis. Both the parametric model and kinematic model can represent all the tolerance classes, but not all the information involved in GD&T can be stored. Datum systems cannot be validated and the analysis is point based rather than zone based.

DoF models treat geometric entities (points, lines, planes) as if they were rigid bodies with degrees of freedom (DoFs) (Zhang 1992; Wu 2002; Kandikjan and Shah 1998; Kandikjan et al. 2001). Geometric relations (angular and linear) are treated as constraints on DoFs. Y14.5 tolerance classes are characterized by how each DoF of each entity is controlled. Technologically and Topologically Related Surfaces (TTRS) models bear many similarities to DoF models (Clément et al. 1995; Desrochers et al. 1995). Later researchers have tried to express Y14.5 tolerance classes in terms of TTRS but this is not fully achieved. Although mathematically elegant, TTRS models are indifferent to Y14.5 Rule #1, floating zones, effects of bonus and shift, form tolerance, or datum precedence. DoF models facilitate the validation of DRF and tolerance types. The model presented here is therefore based on DoF models.

Some hybrid models have been proposed to combine the good aspects of the different models. One of the best hybrid models is *the ASU GD&T Global Model* from Wu et al. (2003), which is mainly a hybrid model of DoF model and Attribute model.

Morphology of GD&T

In this section we consider the types or information, entities and relations needed to express GD&T in accordance with the standards. This will lay the groundwork for the development of the GD&T data model.

Size tolerances as applied to linear, radial or angular dimensions corresponding to parameters related to features of size (i.e. holes, pins, slots, tabs, pockets, bosses, etc.). Therefore, a definition of FOS (feature of size) is needed along with its parametrization (radius, diameter, depth, etc.). Size tolerance specifies max/min values and can be expressed in a variety of ways: max/min parameter limits, nominal value and ± variation which may be equal on both sides (i.e. equal bilateral size tolerance), unequal or unilateral. Any of these can be calculated for any of the other representations. Size tolerances are directly related to the corresponding size parameter, which are defined by the distance/angle between 2 lines and 2 planes or between the center and the boundary of a radial feature.

Geometric tolerances are applied to given entities (edge, surface) or to features of size. There is a tolerance type, value and up to three datums if applicable. Modifiers may be applied to the tolerance value or tolerance zone shape. If datums are FOS, material condition modifiers may be applied. The geometric tolerances will control the orientation, location, shape (form), and profile of the toleranced entity, relative to datum reference(s) of frame.

This leads to the following requirements for entities that need to be supported: face (planar or freeform surface), line, point, and features of size (cylindrical, spherical, tab/slot, etc.) and relations that must be supported: size, orientation, locations, and shape, where a shape relation control the intrinsic form of the feature, and it could be one or several linear, and orientation relations.

SCTF basic concepts

This section will present the basic concepts used in the neutral model and the next section will develop the data structure. The model content includes nominal geometry (features), constraints (including dimensions, mating conditions, assembly constraints), tolerances (including datum reference frames), degrees of freedom (DoFs), and assembly hierarchy. The model is named *Super-Constraint-Tolerance-Feature-Graph-Based Model* (or *the SCTF Model* for short).

Nominal geometric information

The nominal geometric information of the model is composed of the geometric primitives, and their combinations. Each geometric entity has certain inherent DoFs to be controlled.

Geometric primitives and their DoFs

A rigid object in 3D space has six DoFs: three translational (x, y, z) DoFs (TDOFs) along the X-, Y-, and Z-axes, and three rotational (α, β, γ) DoFs (RDOFs) around the X-, Y-, and Z-axes. A DoF is a defined as quantity which defines either the shape, orientation or location of an object. However, for a specific type of primitive, there exist some directions of translation and/or rotation in which the transformation will not change the entity's shape, location or orientation. Such directions, if they exist, are called the invariant DoFs of the primitive. Controlling of a geometric primitive's invariant DoFs is not necessary in the variation based tolerance

analysis. Instead, the active (i.e. non-invariant) DoFs should be fully controlled. By looking at the active DoFs and invariant DoFs an geometric entity may have, all the geometric entities in 3D space can boil down to three primitive entities (i.e. point, infinite line and infinite plane) and their combinations (Kandikjan and Shah 1998; Kandikjan et al. 2001). These abstracted primitives are also referred to as point feature, line feature, and plane feature. Figure 1 shows the active and invariant DoFs of the primitives and their combinations. Because of the existence of the invariant DoF(s), the total number of DoFs required to fully constrain a geometric entity is less than six. For example, a point has three TDOFs along the X-, Y-, Z-axes of the coordinate system. A line has two TDOFs along two directions, orthogonal to each other and to the line direction, and two corresponding RDOFs along the same directions as those for the two TDOFs. A plane has one TDOF along the direction parallel to the plane's normal, and two RDOFs along the two directions, which are orthogonal to each other and are perpendicular to the plane's normal.

Other primitives, such as a sphere, cylinder, circle slot/tab, can be regarded as special cases of point, line and plane. A sphere is a special point that has a size parameter, i.e. spherical radius. A cylinder is a special line with a radius parameter and a height parameter. A circle is a special plane with a radius parameter. A slot/tab is a special plane with three parameters of thickness (size), length, and depth. Sphere, cylinder, circle and slot/tab ate all features of size. If their size control is regarded as a special DoF (Wu et al. 2003), they will have a size DoF, in addition to appropriate kinematic DoFs. In this situation, they should be treated as different primitives other than point, line and plane. In simulation based tolerance analysis, when a geometric entity (corresponding to one of cases listed in Fig. 1 is varied, its active DoFs need to be controlled within the constraints of the geometric tolerance(s) specified on this entity. This point will be revisited later on.

Note that primitives and their combinations are assumed infinite features. They represent the DoFs of the real or trimmed features (see section "Real features and trimmed features"), from which they are abstracted. In this sense, a real freeform feature is a combination of a point, a line and a plane, thus all the six DoFs are active and need to be controlled (Kandikjan and Shah 1998; Kandikjan et al. 2001). Furthermore, the DoF here refers to the kinematic directions

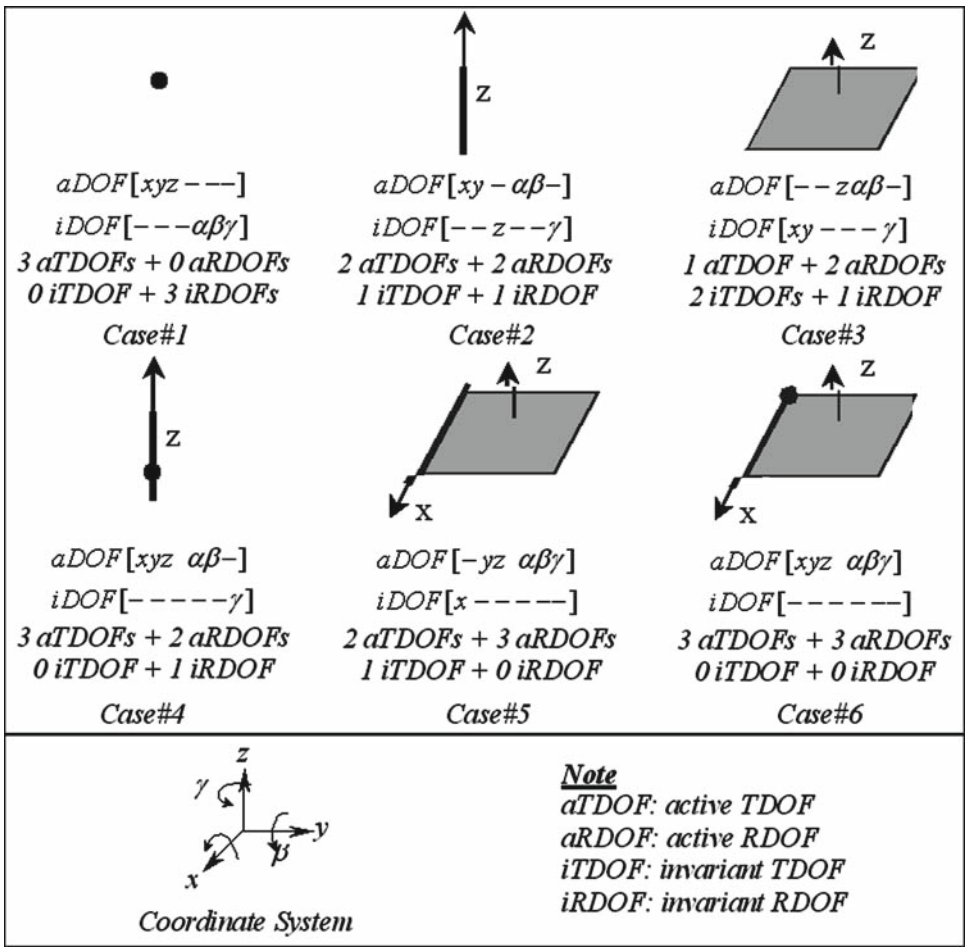


Fig. 1 Active and invariant DoFs of the primitives and their combinations (Kandikjan and Shah 1998)

that a rigid entity can translate or rotate. In CAD area, DoFs are also used to represent the number of independent parameters needed to fully define a geometric entity, such as the shape parameter of a sphere. This will lead to more than just 3TDoFs and 3RDoF. For instance, a vector has three DoFs, i.e. three independent parameters to define its direction and length (x, y, z, L), where $L = \sqrt{x^2 + y^2 + z^2}$ and therefore not all the four parameters are independent.

Real features and trimmed features

In reality, real features (i.e. toleranced surfaces on a part) can be of any type and shape (or profile). There are no abstracted primitives like a pure point feature, an infinite line feature, or an infinite plane feature. Instead, what we see are trimmed features, or approximated features that are abstracted from real ones. In tolerance analysis, it is necessary, as well as, possible to approximate the real features by trimmed features, which are defined as the features simplified or abstracted from the real ones with minor cutouts and protrusions suppressed. For example, all the planar features (assuming the toleranced planar surfaces are involved in a tolerance analysis, rather than otherwise like the tolerance slot, or hole, or hole pattern) in Fig. 2, can be approximated by an ideal rectangular planar feature. Indeed, these real surfaces, with their cutouts and/or protrusions, would most likely to be manufactured (e.g. milled) at one setup. The presence of these minor cutouts and/or protrusions would not affect, in most cases, the choice of the manufacturing process. Therefore, this abstraction from the real feature to the ideal ones is not only necessary but also reasonable, because its effect on the simulation result is negligible.

To view the SCTF-Graph Based Model, the user would like to see the real features or at least the trimmed features, but not the primitives or their combinations. However, behind the scenes, these real features are *simplified* (or *abstracted*) to trimmed features. The latter can be further abstracted to the primitives. Therefore, there are two levels of abstraction involved: from the real features to the trimmed features, from the trimmed features to the abstracted primitives. The CTF-Graph Based Model will save the trimmed feature information, which is a bridge between the real features and the primitives. It is a trivial thing to go from the trimmed features to the primitive features, but not the other way around.

Besides the basic parameters that define a primitive feature, additional parameters are used to differentiate one trimmed feature from the other within the same group corresponding to one primitive type. For instance, a pin feature and a conical feature are both of a line feature, which has a base point parameter and a direction parameter. To differentiate a pin from a conical feature, a pin has one diameter, a height, and a type attribute, while a conical has two heights or diameters, one cone angle, and a type attribute. Figure 3

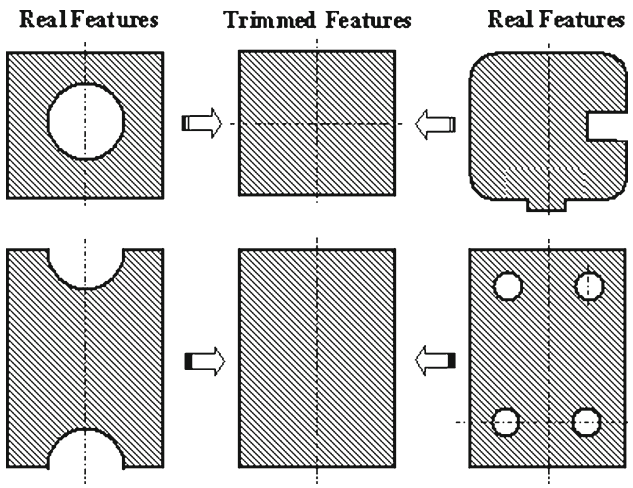


Fig. 2 Shape approximation for planar features

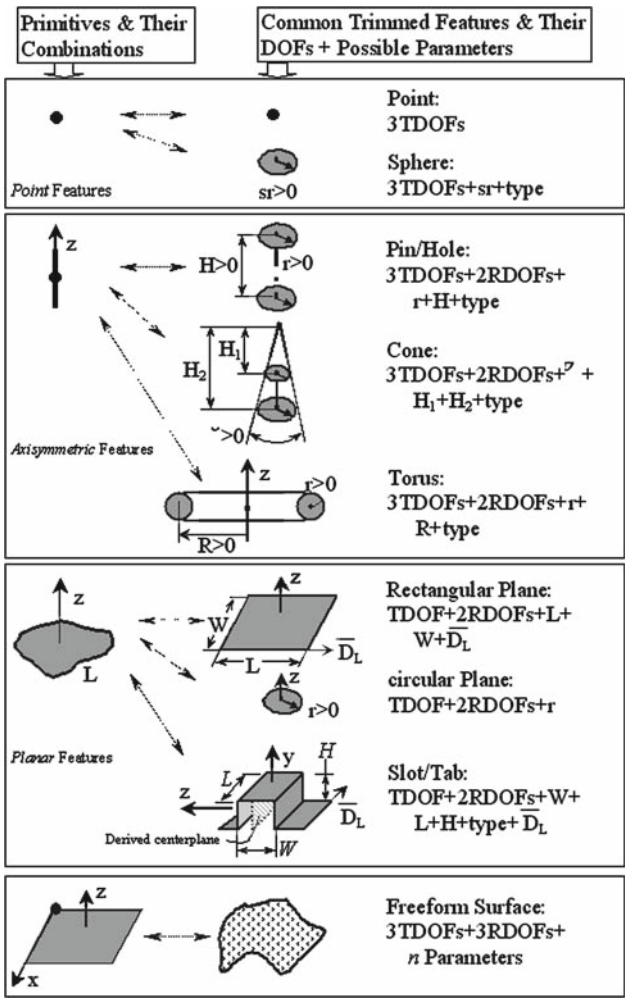


Fig. 3 Common trimmed features corresponding to the primitives and their combinations

Table 1 Partial list of the metric relationships in 3D space

Entity 1	Entity 2	Relationship	Constraints
Line 1	Plane P	Coincident	$pA + qB + rC = 0$ and $AX_1 + BY_1 + CZ_1 + D \neq 0$
		Perpendicular	$\frac{P_1}{A_p} = \frac{q_1}{B_p} = \frac{r_1}{C_p}$
Plane 1	Plane2	Angle θ	$\cos(\theta) = \frac{\lvert A_1A_2+B_1B_2+C_1C_2\rvert}{\sqrt{A_1^2+B_1^2+C_1^2}\sqrt{A_2^2+B_2^2+C_2^2}}$
		Parallel	$\frac{A_1}{A_2} = \frac{B_1}{B_2} = \frac{C_1}{C_2}$
		Distance d	$d = \frac{\lvert A_2X_1+B_2Y_1+C_2Z_1+D_2\rvert}{\sqrt{(A_2)^2+(B_2)^2+(C_2)^2}}$ and $\frac{A_1}{A_2} = \frac{B_1}{B_2} = \frac{C_1}{C_2}$
		Coincident	$A_2X_1 + B_2Y_1 + C_2Z_1 + D_2 = 0$ and $\frac{A_1}{A_2} = \frac{B_1}{B_2} = \frac{C_1}{C_2}$
		Perpendicular	$A_1A_2 + B_1B_2 + C_1C_2 = 0$
Point 1	Point 2	Distance d	$(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2 = d^2$
Point P	Line 1	Distance d	$d = \frac{\begin{vmatrix} i & j & k \\ p_1 & q_1 & r_1 \\ X_p - X_1 & Y_p - Y_1 & Z_p - Z_1 \end{vmatrix}}{\sqrt{(p_1)^2+(q_1)^2+(r_1)^2}}$
		Coincident	$\frac{X_p-X_1}{p_1} = \frac{Y_p-Y_1}{q_1} = \frac{Z_p-Z_1}{r_1}$
Point p	Plane 1	Distance d	$d = \frac{\lvert A_1X_p+B_1Y_p+C_1Z_p+D_1\rvert}{\sqrt{(A_1)^2+(B_1)^2+(C_1)^2}}$
		Coincident	$A_1X_p + B_1Y_p + C_1Z_p + D_1 = 0$
Line 1	Line 2	Angle θ	$\cos(\theta) = \frac{p_1p_2+q_1q_2+r_1r_2}{\sqrt{p_1^2+q_1^2+r_1^2}\sqrt{p_2^2+q_2^2+r_2^2}}$
		Parallel	$\frac{p_1}{p_2} = \frac{q_1}{q_2} = \frac{r_1}{r_2}$
		Distance d	$d = \sqrt{\left([(X_2 - X_1)^2 + (Y_2 - Y_1)^2 + (Z_2 - Z_1)^2] + \frac{p_1[X_2-X_1]+q_1[Y_2-Y_1]+r_1[Z_2-Z_1]}{p_1^2+q_1^2+r_1^2}\right)}$ where $i = 1$ or 2
Line 1	Line 2	Coincident	$\frac{X_1-X_2}{p_2} = \frac{Y_1-Y_2}{q_2} = \frac{Z_1-Z_2}{r_2}$
		Perpendicular	$p_1p_2 + q_1q_2 + r_1r_2 = 0$
Line 1	Plane p	Angle θ	$\sin(\theta) = \frac{\lvert p_1A_p+q_1B_p+r_1C_p\rvert}{\sqrt{p_1^2+q_1^2+r_1^2}\sqrt{A_p^2+B_p^2+C_p^2}}$
		Parallel	$pA + qB + rC = 0$
		Distance d	$d = \frac{\lvert A_pX_1+B_pY_1+C_pZ_1+D_p\rvert}{\sqrt{A_p^2+B_p^2+C_p^2}}$ and $pA + qB + rC = 0$

lists some of the common trimmed features and their corresponding primitives or combinations of the primitives. The trimmed feature's active DoFs and possible parameters are also listed in Fig. 3. Note that the *type* parameter in Fig. 3 specifies the type of the feature, because the same surface has two sides to which the material of the part can reside. For instance, a cylindrical surface can correspond to a pin feature if the material is inside the surface, or to a hole feature if the material is outside the surface.

Note that there are different ways to define a trimmed feature. The representation of the trimmed features is discussed in section "SCTF model structure".

Constraints and metric relationships

A geometric constraint in GD&T corresponds to a basic metric relationship between the primitives. Each metric relationship may be expressed in one or more analytical equations from the analytical geometry.

Different metric relationships exist between the primitives, and constrain the DoFs of geometric entities w.r.t. each other (Wu et al. 2003). We use the same representation, i.e. (X_i, Y_i, Z_i) for points, $A_iX + B_iY + C_iZ + D_i = 0$ for planes, and $\frac{X-X_i}{p_i} = \frac{Y-Y_i}{q_i} = \frac{Z-Z_i}{r_i}$ for lines, then the metric relationships are as those listed in Table 1. When a feature of size is involved, the metric relationships in Table 1 can still be used, but the feature's size must be taken into account. As shown in Fig. 4, in order to compute the distance between a point and a circle (i.e. a special plane), the coincident relationship between a point and a plane can be used to check if the point is coincident with the circle defined plane: (1) If yes (say point P_d), the distance can be defined as the point-to-point distance L_C between point P_d and the circle center P_c. The distances between point P_d and the boundary of the circle can be easily computed from L_C and the circle's radius r. (2) If no (say point P_h), the distance can be defined as the

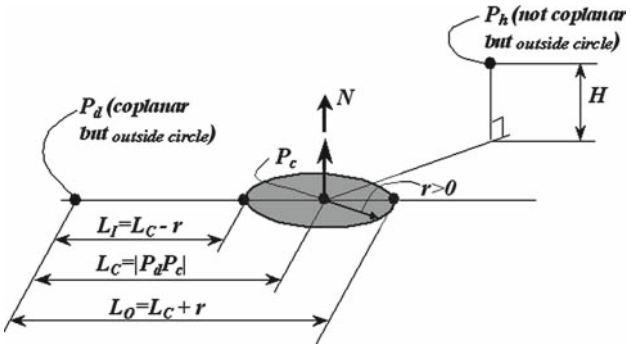


Fig. 4 Computation of distances between points and a circle

perpendicular distance H from point P_h to the circle-defined plane. In a word, the metric relationships between the primitives and the size primitives (sphere, cylinder, circle, and slot/tab) are either directly available or derivable from analytical geometry.

Geometric constraints may be specified dimensions, mating conditions, or geometric relations, such as perpendicularity, parallelism. For size features, size constraints can be directly attached to the features themselves. A geometric constraint may have a measurement direction associated with it, but it is not always the case. For instance, if a constraint involves a *plane*, its measurement direction is the plane normal. If it involves two parallel but non-coincident *lines*, its measurement direction is the direction that passes through and perpendicular to the lines. If a constraint involves two coincident *lines* only, its measurement direction is the *line direction*. A size constraint of a sphere (i.e. a *point* feature) will not have a fixed measurement direction. Other cases are not enumerated here. The key point is that a constraint, especially a dimension, requires a measurement direction, and that direction will depend on how this constraint is actually measured in manufacturing and inspection. The user can specify the measurement direction for a geometric constraint by specifying the datum of this measurement.

Entity degrees of freedom

The geometric primitives or the trimmed features have their respective active DoFs and invariant DoFs. To limit the variation of a certain feature, its active DoFs should be fully controlled within certain ranges, i.e. tolerances. In words, the tolerance specification should control the feature's variations along its active DoFs with respect to its datum reference frame. Indeed, a GD&T specification will control the corresponding active DoFs of the toleranced feature, and each datums, if any, will control some of them.

We do not have enough space to fully list all possible tolerance classes and how they control the active DoFs of the primitive features, i.e. point, line, and plane features in most scenarios. Instead, we will use the position tolerance of three datums on a line for demonstration. But interested readers can refer to (Shen 2005) for a complete coverage. Let V stand for a *Point* feature (point, sphere, part of a spherical surface), L for a *Line* feature (e.g. line, pin, hole, revolved, prismatic, cone, torus, etc.), P for a *Plane* feature (e.g. plane, tab, slot, etc.), T for the target feature, A for the primary datum feature, B for the secondary datum feature, C for the tertiary datum feature, // for parallelism relationship, $\perp$ for perpendicularity relationship, $\rightarrow$ for "control" (certain DOFs), = for "is" or "equal to" , D_E for direction of E (i.e., line direction or plane normal direction), + for "and", *RDOF* for rotational DOF, *TDOF* for translational DOF, *RDOF(D)* for rotational DOF around direction D, *TDOF(D)* for translational DOF along direction D, and $\times$ for "cross product" operation.

A line feature has one invariant RDOF, one invariant TDOF, two active RDOFs, and two active TDOFs, as shown in case#2 in Fig. 1. In trimmed features, a line segment does not have the invariant TDOF along the line direction, and this TDOF may needs to be controlled by the GD&T specification as well. A line feature can represent any revolved surface features and any linear edges in the SCTF-Graph. For position tolerance of three-datum (ABC) on a line feature:

(i) A = P, B = P, C = P, A$\perp$B, A$\perp$C, B$\perp$C, A$\perp$T (i.e. D_A// D_T): then A $\rightarrow$ 2RDOF + 1TDOF corresponding to A(=P); B $\rightarrow$ 1TDOF(D_B); C $\rightarrow$ 1TDOF(D_C). Note that A controls the TDOF along the line direction, because a real line feature is not infinite, and its invariant TDOF becomes active.

(ii) A = P, B = P, C = L, A$\perp$B, A$\perp$C, B//C but B is not coincident with C, A$\perp$T (i.e. D_A//D_T): then A $\rightarrow$ 2RDOF + 1TDOF corresponding to A (=P); B $\rightarrow$ 1TDOF(D_B); C $\rightarrow$ 1TDOF($D_A \times D_B$).

(iii) A = P, B = V, C = V, B is not coincident with A or C, C is not coincident with A, A$\perp$T (i.e. D_A//D_T): then A $\rightarrow$ 2RDOF + 1TDOF corresponding to A (=P); B $\rightarrow$ 1TDOF; C $\rightarrow$ 1TDOF.

Note: Reordering of the above ABC is not allowed, because the primary datum should control more DOF than the secondary datum, which should control no less DOF than the tertiary.

Very similarly, how other different tolerance specifications control the different active DoFs of a certain abstracted primitive can be inferred, see (Shen 2005). The DoF information is implicit from the target feature type and its tolerance specification, including datum reference frame and datum feature type, and geometric relationships between them and the target feature itself. This DoF information can be output to a data file and it is used for developing variation algorithms of the features.

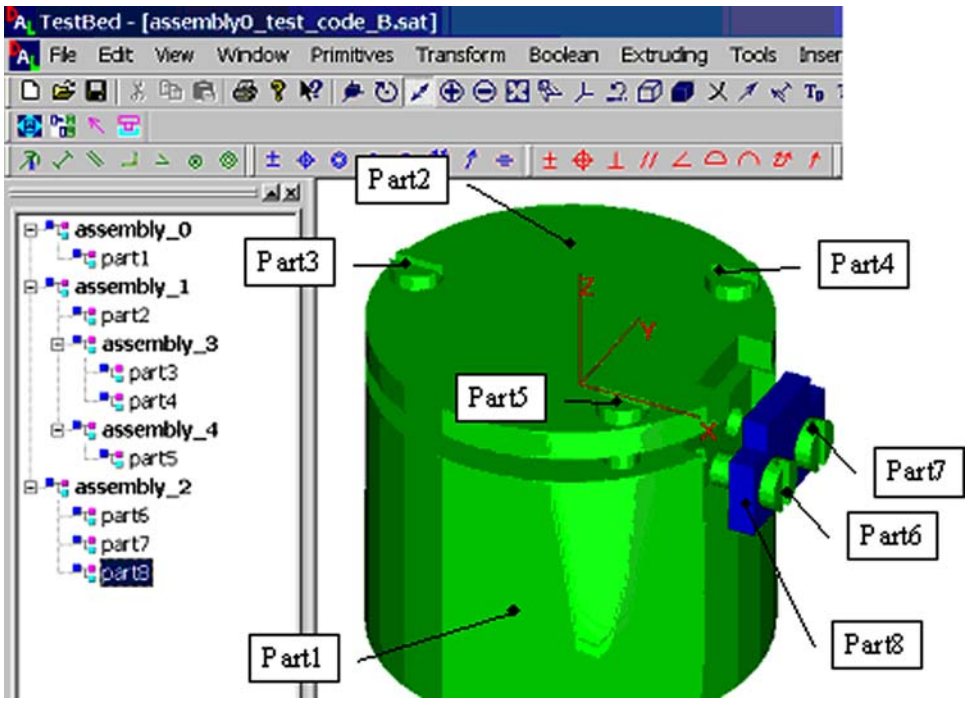


Fig. 5 The B-Rep model with GD&T and assembly information

Assembly hierarchy information

An assembly can be composed of parts and sub-assemblies. Each sub-assembly can be again composed of parts and/or sub-assemblies, and so on. Figure 5 shows an assembly composed of multiple parts and subassemblies. Note that in Fig. 5 Part 5 is intentionally separate from Part 3 and Part 4 to form a subassembly of "assembly_1", in order to test the data structures and implementation. In reality, a simpler hierarchy would be used. We will revisit Fig. 5 later. Mating constraint data is included as a subset of constraints in the SCTF-Graph. Assembly hierarchy is useful when solving the mating constraints between a subassembly and other components in the assembly. All parts in a subassembly are treated as a "set" when the constraint model is created using DCM 3D.

SCTF model structure

This section will discuss how the SCTF-Graph Based Model is represented and implemented in this research. The higher-level structures are discussed before the lower-level structures.

SCTF-Graph representation

The SCTF-Graph, at the highest level, is a general tree. The data structures for this *general tree* are the *tree node* class and the *general tree* class. A general tree node has a data member variable *data* and three pointers to link current node to its *parent*, its *child* and its *sibling* nodes. It is also a template class, since the tree node is a template class. To traverse, modify and retrieve data from the general tree, various *access*, *utility* and *modifier* functions are defined as well.

Using the template general tree representation, general trees of different data types can be created, depending on the user-defined data type. For the SCTF-Graph general tree, the data type T is defined in a *CTF struct*, shown in Fig. 6. The variable *node_name* holds the name of current node, e.g. "assembly_1". The variable *part_list_ptr* is a nested *doubly-linked list* of data type *CGeometry*; in this research, it points to a CTF-Graph at the current node in the assembly general tree structure (i.e. the SCTF-Graph structure). Representations of *CTF-Graph*, *doubly-linked list*, *CGeometry* will be discussed in a moment. For now, let's concentrate on the high level structure, where the SCTF-Graph expands like the one shown in Fig. 7.

```
//Data type for the general tree node in Super CTF-graph
struct CTF_TreeNodeData
{
        CString node_name;
        LinkedList<LinkedList< CGeometry*>*> *part_list_ptr;

        CTF_TreeNodeData (CString n = "", ENTITY *e = NULL);
        ~CTF_TreeNodeData() {}
        ...
};
```

Fig. 6 The tree node data type for the SCTF-Graph

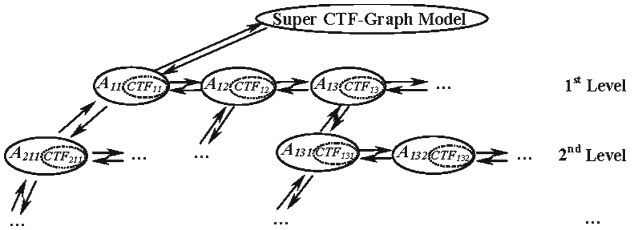


Fig. 7 The SCTF-Graph structure

The general the tree node in Fig. 7 contains the node's pointers, name and a CTF-Graph. The CTF-Graph can further expand like the one shown in Fig. 8. It is a nested *doubly-linked list* data structures. At the very fist layer, it is a *list* of "parts" contained in the current subassembly node. At the second layer, each "part" is composed of a *list* of geometric "features" (of *CGeometry* type). At the third layer, each geometric "feature" contains its basic geometric data, a *list* of geometric constraints, a *list* of tolerances, a *list* of DoFs, and a *list* of associated points (i.e. a special *CGeometry*).

The whole model is created from top down, and lower level data is gradually populated when the higher-level data is available. The order is "*The general tree* → *parts* → *features* → *constraints* → *tolerances* → *DoFs*".

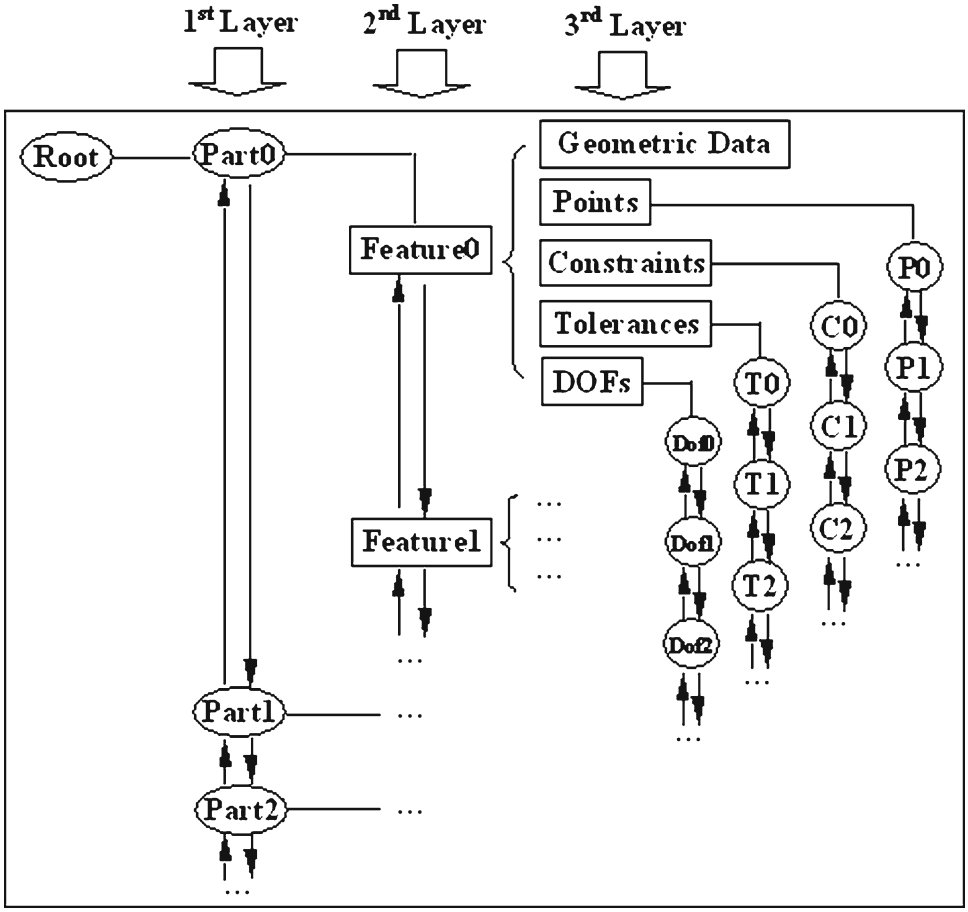
The lower-level data representations, such as those for *CGeometry* (or feature), constraint (C-graph), tolerance (T-graph), and DoF are discussed in the following sub-sections.

Constraint structure

The constraints are represented by a C-Graph, an undirected graph with the involved trimmed features at the nodes and the geometric constraints as the arcs. This measurement direction is useful for traversing the C-Graph to detect the tolerance chain (Shen 2005; Shen et al. 2008). Figure 9 shows the GD&T specification for a part with two through slot features, where the measurement direction is along the normal to datum B. Note that X, Y and C in Fig. 9 are different dimensions which we might be interested in analyzing. Figure 10 shows the C-Graph for this part. Note that the face ID numbers are automatically assigned when the model is created, and uniqueness of the face ID is guaranteed.

The geometric constraints in the Model are represented in different classes derived from the constraint base class *DAL_Geom_Cst*. The constraint class hierarchy is shown Fig. 11. Note that constraint type *eType* is an *enum* type data. Involved geometric entities are saved in the pointers

Fig. 8 The structure of the CTF-Graph



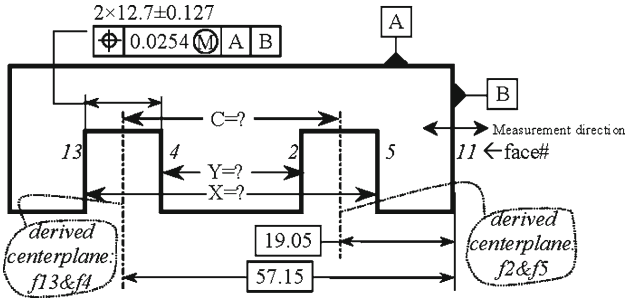


Fig. 9 GD&T specification for a part with two through slots features

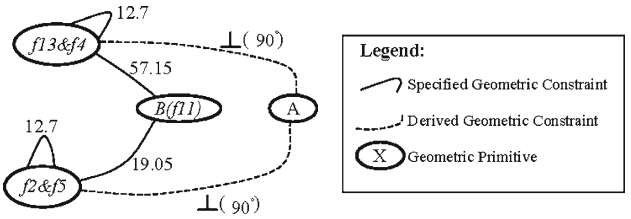


Fig. 10 The undirected C-Graph for the part shown in Fig. 9

(i.e. *gTarget_A*, *gTarget_B*) of the type *CGeometry*. A geometric constraint can have direction vector, defined as a *CCoordinate3D* struct.

Feature data structure

Geometric information of the trimmed features needs to be encoded along with the GD&T information. Representation of the trimmed features needs to be designed in such a way that it can link to the GD&T data and be supported by the geometric constraint solver, and can accommodate the requirements from the analysis processes. As pointed out in section "Real features and trimmed features", all geometric entities boil down to *point*, *line* and *plane* from the DoF point of view. A *point*, *line* or *plane* can correspond to many different trimmed features. During the tolerance analysis, it

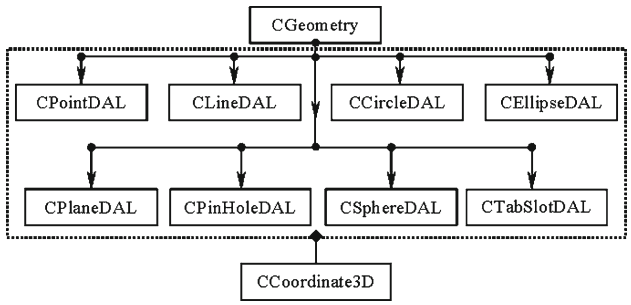


Fig. 12 Hierarchy of the classes for the features in the Model

is necessary to distinguish one from the other within the group of trimmed features corresponding to the same primitive entity. For instance, a *line* can represent a *pin*, a *hole*, a *cone*, and other revolved surfaces; but they each have their special attributes that a pure *line* does not have. In tolerance analysis involving a *line*, it is necessary to distinguish a pin or hole feature from a cone or helix feature. Therefore, the feature representation will recognize this difference, instead of just three types of features, i.e. *point*, *line*, and *plane*.

Figure 12 shows the hierarchy of the feature classes, where the *CGeometry* is the base class, and other geometric types are all derived from it. Any new features (e.g. pockets) can be easily extended from the *CGeometry* base class. The *struct CCoordinate3D* is extensively used in defining these derived classes.

The definition of the base class *CGeometry* is shown in Fig. 13. The virtual functions are to be overridden in the derived classes. The geometric constraints are saved in a *doubly-linked list* variable (i.e. *CstList*) and the tolerances are also saved in a *doubly-linked list* variable (i.e. *TolList*). A non-point feature can have multiple points on it, and these associated points (i.e. *CPointDAL*) are also saved in a *doubly-linked list* variable (i.e. *PointList*). The *ENTITY* type variable *acis_pointer* will associate this *CGeometry* object to the

Fig. 11 Constraint class hierarchy

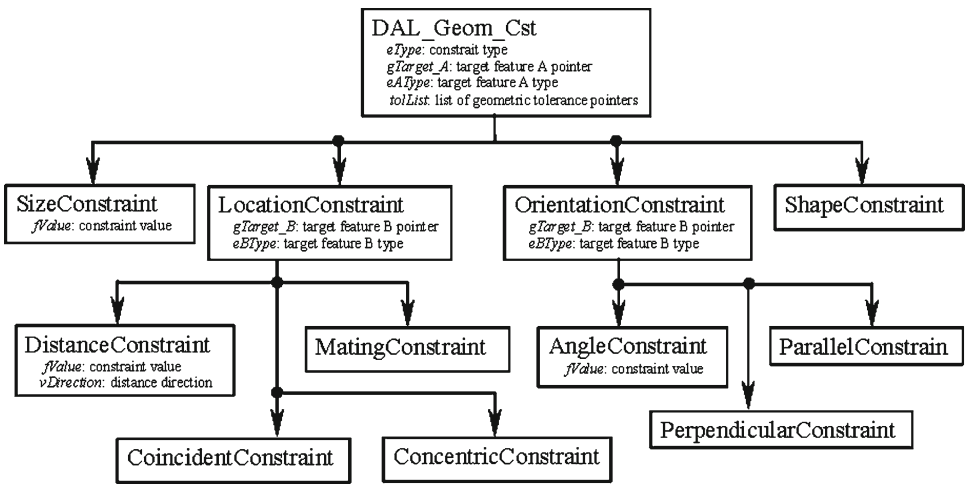


Fig. 13 Definition of the base class CGeometry

```
//This is the base CGeometry class.  All other geometric (CPointDAL, CLineDAL,
//CCircleDAL, CEllipseDAL, CPlaneDAL, CPinHoleDAL, CTabSlotDAL,
//CSphereDAL etc.) classes are derived from this base class.
class CGeometry
{
 public:
        //some virtual functions==================================
        virtual void transform(double [4][4]) {}
        virtual void getString(CString &s) {}
        virtual int EFeatureTypeDAL() const = 0;
        virtual ENTITY* getACISpointer() { return acis_pointer; }
        virtual void setACISpointer(ENTITY* pt) { acis_pointer = pt; }
        virtual void set_radius(double) {}; // Only required for circles/spheres/cylinders
        //geometric constraints, including mating conditions================
        LinkedList<DAL_Geom_Cst*>CstList;
        //geometric tolerances, including size/dimensional tolerance===========
        LinkedList<DAL_Geom_Tol*>TolList;
        //DOF list
        LinkedList<DAL_Geom_DOF*>DoFList;
        //other variable
        LinkedList<CPointDAL*> PointList;
        ENTITY* acis_pointer;
        ...
};
```

```
// Plane
class CPlaneDAL : public CGeometry
{       CCoordinate3D fbase_point;
        CCoordinate3D fnormal;
        double flength;
        double fwidth;
        CCoordinate3D fLdirection;
        int type; //rectangular, disk, tab, slot, etc.
        // functions for metric relationships
        double angle_to_line(const CLineDAL& g);
        double angle_to_plane(const CPlaneDAL& g);
        double angle_to_circle(const CCircleDAL& g);
        double angle_to_cylinder(const CPinHoleDAL& g);
        double distance_to_point(const CPointDAL& g);
        double distance_to_line(const CLineDAL& g);
        double distance_to_plane(const CPlaneDAL& g);
        double distance_to_circle(const CCircleDAL& g, int type);
        double distance_to_cylinder(const CPinHoleDAL& g, int type);
        ...
        //other functions
        void transform(double mat[4][4]);
        void getString(CString&s);
        int EFeatureTypeDAL() const;
        ENTITY* getACISpointer() {return acis_pointer; }
        void setACISpointer(ENTITY* pt) {acis_pointer = pt; }
        ...
};
```

Fig. 14 Definition of class for the geometric entity plane

corresponding geometric kernel *ENTITY* for the visualization purpose. Note that ENTITY is a class type of the geometry kernel ACIS.

Derived from the base *CGeometry* class, different types of trimmed features are defined, such as point, line, plane, tab/slot, etc. As an example, we show the definitions of only plane feature, i.e. *CPlaneDAL* in Fig. 14. The common metric relationships (as discussed in section "Constraints and metric relationships") of a *plane* feature with respect to other geometric features (e.g. *line*, *plane*, *cylinder*, etc.) are defined as member functions of this class. The virtual functions in *CGeometry* are to be overridden in *CPlaneDAL*.

Definitions for other types of features are very similar to *CPlaneDAL* presented above. Therefore, they are omitted as well.

Tolerance data structure

The tolerance information in a CTF-Graph Based Model forms a tolerance-graph, called T-Graph. Unlike C-Graph, T-Graph is a directed graph with the toleranced geometric features at the nodes and tolerance specification as the arcs. For those tolerances that have no datum reference frame (DRF, i.e. the coordinate systems used to locate and orient a part feature (ASME 1994)), the tolerance is attached to the geometric features itself. Figure 15 shows the T-Graph for the part shown in Fig. 9. Note that the tolerances in Fig. 15 are actually pointers to such tolerance objects.

Since a tolerance is used to control the variation of a certain geometric constraint, it depends on the corresponding geometric constraint. In other words, a tolerance cannot exist without its corresponding geometric constraint. For instance, a dimensional plus/minus tolerance has no meaning if the

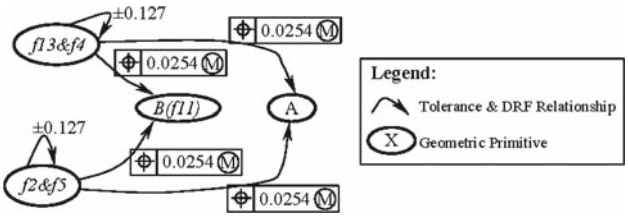


Fig. 15 The directed T-Graph for the part shown in Fig. 9

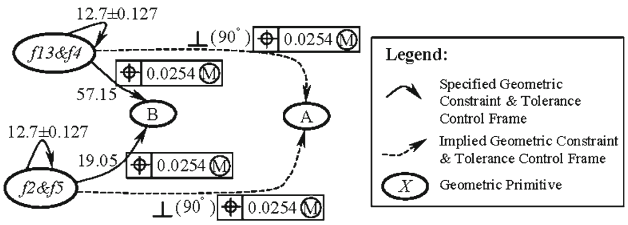


Fig. 16 The CT-Graph for the part shown in Fig. 9

corresponding dimension does not exist. Therefore, C-Graph and T-Graph can be combined together to form a directed constraint-tolerance-graph, i.e. CT-Graph. Figure 16 shows the CT-Graph for the part shown in Fig. 9. In the actual model, the tolerance specifications associated with the arcs in Fig. 16 are replaced by corresponding pointers to the same tolerance instance to avoid duplication. The tolerance instance is duplicated in Fig. 16 for illustration purpose only.

A CT-Graph is also referred to as constraint-tolerance-feature-graph i.e. CTF-Graph, since it has the geometric features (or trimmed features) at its nodes. Indeed, the geometric features are indispensable components for a C-Graph and a T-Graph. The CTF-Graph does not include the assembly hierarchy information, which is encoded in the SCTF-Graph.

The tolerance information in the Model is represented in different classes derived from the tolerance base class *DAL_Geom_Tol*. The tolerance class hierarchy is shown Fig. 17. Note that tolerance type *eType* is an *enum* type data. Involved geometric entities are saved in the pointers (i.e. *gTarget*, *datum_A*, *datum_B*, *datum_C*) of the type *CGeometry*. Material condition modifiers (e.g. MMC, LMC, and RFS) are also saved as *enum* type.

DoF representation

A DoF (in the kinematic sense) of a feature is represented in a *struct* using *CCoordinate3D* definition. All active DoFs of a feature is saved in a *doubly-linked list* in the feature itself, as shown previously in Fig. 13. These active DoFs are controlled by the corresponding tolerance(s) specified on this feature; therefore, for each tolerance object, there is *doubly-linked list* that contains all the DoFs this tolerance actually controls. The contents of the Tolerance class are: tolerance type, tolerance value (fValue), Diameter modifier (Diam_Symbol), Target feature pointer (eTarget), Target type (eTargetType), and a Linked List representing the DoFs. A union of all the DoFs associated with the tolerances specified on a feature should be equal to the set of DoFs contained in the list held by the feature itself.

Implementation

The SCTF model has been implemented using the C++ language, commercial geometric kernel *ACIS*, and commercial geometric constraint solver *DCM 3D*. Written in C++, ACIS provides an open architecture framework for wireframe, surface, and solid modeling from a common, unified data structure. DCM 3D provides dimension-driven, constraint-based design functionality to CAD/CAM/CAE applications; it enables the efficient use of dimensions and constraints to position parts in assemblies and mechanisms, to control the shape of parts, and to produce 3D sketches.

With the attributed CAD model the SCTF-Graph Based Model can be automatically created, as explained below:

(1) Traverse the attributed CAD model to retrieve all the GD&T information, i.e. geometric constraints and the associated tolerances, mating conditions.
(2) Check all the GD&T data to find out all the geometric entities (real physical features) involved, and group the entities according to their owning parts.
(3) Create the general tree to capture the assembly hierarchy information, and populate this tree in the order of subassembly, part, constraint, tolerance, and DoF.
(4) Create the real physical features for the visualization purpose.
(5) Abstract the real physical features to the trimmed features, which correspond to the geometric primitives and their combinations. This is where the feature recognition technique can come into play.
(6) Create all the trimmed features and populate all the GD&T data (geometric constraints and their dependent tolerances, mating conditions) to generate the SCTF-Graph Based Model.

It is important to note that the same model is always created for the same attributed CAD model, regardless of what independent parameter is being analyzed. With the SCTF-Graph Based Model automatically created, it is possible to conduct different types of tolerance analyses right on top of this model. See (Shen 2005) for how different types of tolerance analyses are performed driven from the same SCTF model.

Figure 18 shows the SCTF-Graph model for the assembly in Fig. 5. It can be seen the assembly hierarchy information

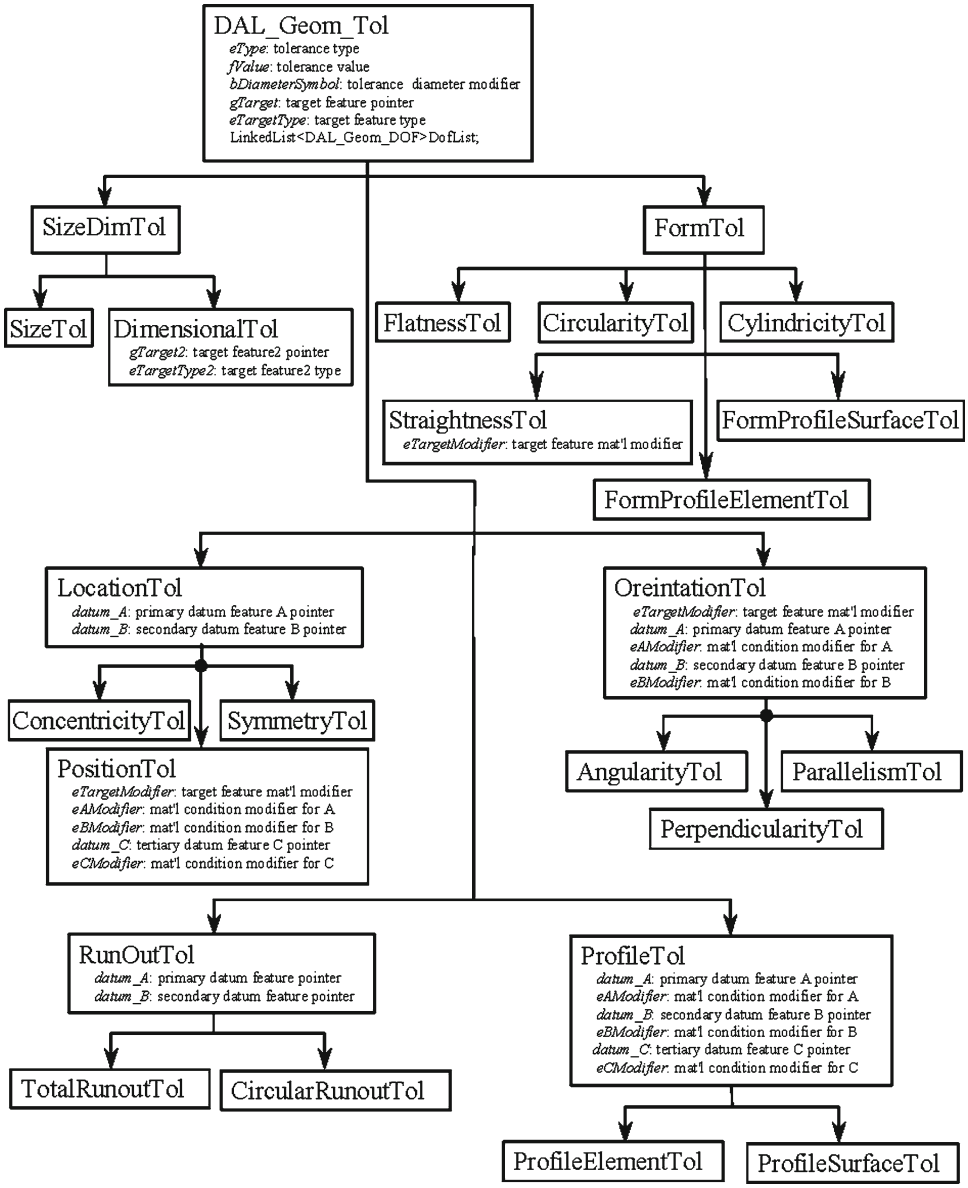


Fig. 17 Representation of the tolerance information

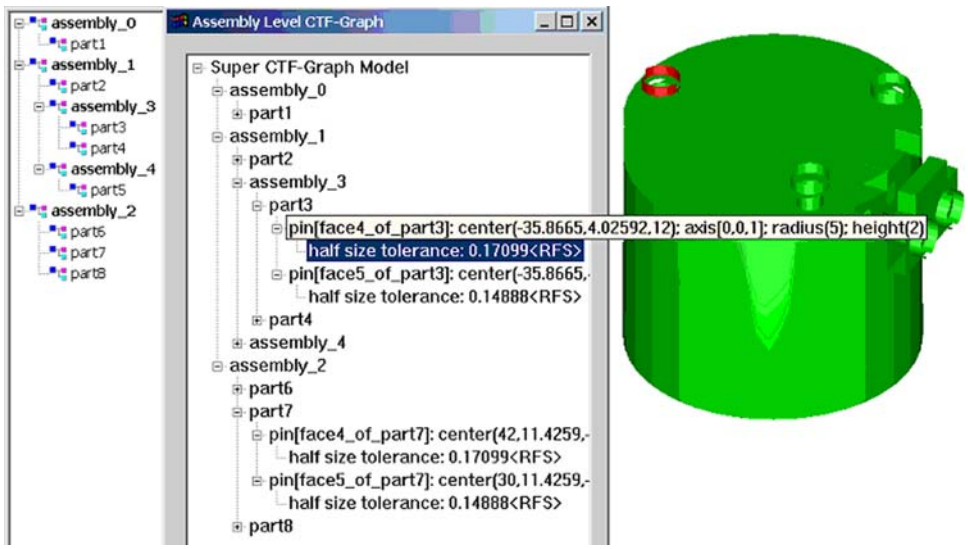


Fig. 18 The SCTF-Graph Model for assembly shown in Fig. 5

Fig. 19 Output data file of the SCTF-Graph Based Model shown in Fig. 18

```
#0=FILE('E:\ASU_GDTtestbed\sat\assembly0_test_code_B.sat');                    (A)
#1=PART('part1',#2,#3,#4,#5);
#2=RECTANGULAR_PLANE('face13_of_part1',(28.4335, 4.02592, 0),[0, 0, 1],29.5973, 5,[0, 1, 0]);
#3=CIRCULAR_PLANE('face4_of_part1',(-4.06647, 4.02592, -56),[0, 0, -1],76, OPTIONAL[1, 0, 0]);
#4=PIN('face16_of_part1',(-4.06647, 4.02592, 0),[0, 0, -1],38, 15);
#5=RECTANGULAR_PLANE('face18_of_part1',(25.9335, 4.02592, -7.5),[1, 0, 0],29.5973, 15,[0, 1, 0]);
#6=PART('part2',#7,#8,#9,#10);
...
#11=PART('part3',#12,#13);
...
#14=PART('part4',#15,#16);                                                    (B)
...
#17=PART('part5',#18,#19);
...
#20=PART('part6',#21,#22);
...
#23=PART('part7',#24,#25);
...
#26=PART('part8',#27,#28,#29,#30,#31,#32,#33,#34,#35,#36,#37);
...
#37=RECTANGULAR_PLANE('face9_of_part8',(38.4335, -7.8734, 0),[0, 0, 1],5.79865, 5,[0, 1, 0]);
#38=CST_DISTANCE(56,#2,#3);
#39=METRIC_RELATIONSHIP(#38, CST_DISTANCE, (56,#2[PLANE],#3[PLANE]));
...                                                                           (C)
#72=CST_M_AGAINST(#5,#32);
#73=METRIC_RELATIONSHIP(#72, CST_M_AGAINST, (#5[PLANE],#32[PLANE]));
#74=T_SIZE(#4,(nFI, 0.33619, RFS));
#75=DOF(#74,(SIZE_DOF, SHAPE_DOF));
...                                                                           (D)
#122=T_FLATNESS(#37,(nFI, 0.01, RFS));
#123=DOF(#122,(SHAPE_DOF));
#124=ASSEMBLY('assembly_0',#1)
#125=ASSEMBLY('assembly_3',#11,#14)
#126=ASSEMBLY('assembly_4',#17)                                               (E)
#127=ASSEMBLY('assembly_1',#6,#125,#126)
#128=ASSEMBLY('assembly_2',#20,#23,#26)
#129=MODEL(#124,#125,#128)
```

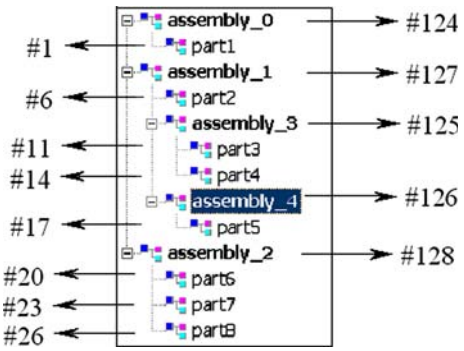


Fig. 20 Assembly relationships in the CTF-Graph compared with Sec. E of the data file shown in Fig. 18

has been transferred from the B-Rep CAD model. The Model in Fig. 18 can be output to a textual file shown in Fig. 19. This output can be used by any type of tolerance analysis program regardless of the techniques, whether Min/Max Charts, Monte Carlo Simulation or T-maps, etc (Shen 2005). The textual file has five sections A–E. *Section A* contains the name of the original B-Rep model file (i.e. ACIS *.sat file with additional information attached as attributes) from which the CTF-Graph Based model has been created. It occupies one line only. Other sections always occupy multiple lines. *Section B* contains all the geometric information of all the parts in the CTF-Graph Based model. Each part is composed of a certain number of features (of *CGeometry* type). Data of each feature is output in one line. In this case, Section B occupies lines from #1 to #37. *Section C* contains all the constraint and metric relationship information, including mating conditions, in the CTF-Graph Based model. In this case, Section C occupies lines from #38 to #73. *Section D* contains all the tolerance and DoF information in the CTF-Graph Based model. In this case, Section D occupies lines from #74 to #123. *Section E* contains all the assembly hierarchy information in the CTF-Graph Based model. In this case, Section E occupies lines from #124 to #129, which match very well with the original assembly hierarchy information, as further shown in Fig. 20. Comparing Section E in Fig. 19 with Fig. 20, we can find out the assembly hierarchy information is exactly transferred from the original CAD model to the CTF-Graph Based model. Line#124 states an assembly, named 'assembly_0', is composed of one part or assembly defined in Line#1. Line#125 states an assembly, named 'assembly_3', is composed of parts or assemblies defined in Line#11 and Line#14. Line#126 states an assembly, named 'assembly_4', is composed of one part or assembly defined in Line #17. Line#127 states an assembly, named 'assembly_1', is composed of parts or assemblies defined Line#6, Line#125, and Line#126. Line#128 states an assembly, named 'assembly_2', is composed of parts or assemblies

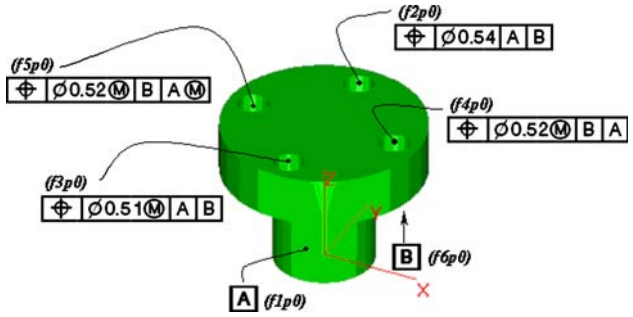


Fig. 21 A CAD model with GD&T for testing DoF information in the model

defined in Line#20, Line#23, and Line#26. Finally, Line#129 states that the whole CTF-Graph Based model is composed of parts or assemblies defined in Line#124, Line#125, and Line#128.

An additional example is provided (see Fig. 21) to explain the textual file format and contents. Note that the distance constraint between the four holes and the datum pin feature A are not shown in Fig. 21. Neither are the size constraints for the pin and hole features. From the CAD model shown in Fig. 21, the SCTF-Graph Based model can be automatically created, as shown in Fig. 22. The data contained in the model can be output to a textual file as shown in Fig. 23. The following will explain how to interpret the data file using this example.

In *Section A* (Line#0), Line#0 contains the original B-Rep CAD model from which the CTF-Graph Based model has been created.

In *Section B* (Line#1–Line#7), there is only one part. Line#1 states that this PART is named 'part0', an d composed of 6 constituent features defined respectively in Line#2–Line#7. Line#2 states a PIN feature, named 'FACE1_of_part0', has root point (0, 0, 0), axial direction [0, 0, 1], radius of 10, and height of 20. Similar interpretation can be applied to the hole features defined in Line#3–Line#6. Line#7 states a CIRCULAR_PLANE feature, named 'FACE6_of_part0', has root point (0, 0, 20), normal direction [0, 0, −1], and radius of 40. Feature data comes from the class definition of the corresponding feature.

In *Section C* (Line#8–Line#15), there are four constraints in the model. Line#8 states there is a CST_DISTANCE of value 15 between the features defined in Line#3 and Line#2; Line#9 states the METRIC_RELATIONSHIP regarding the constraint in Line#8 is of type CST_DISTANCE, valued 15, between a Line feature (i.e. axis of HOLE), defined in Line#3 and the other Line feature (i.e. axis of PIN) defined in Line#2. Similar interpretation can be applied to the other lines.

In *Section D* (Line#16–Line#25), there are five tolerances and the DoFs controlled by the tolerance specification. Line#16 states a size tolerance on the feature defined in Line#2, and its value is Ø(±0.5) on the radius. RFS applies on size tolerance. Line#17 states the size tolerance defined in Line#16 controls both SIZE DoF and SHAPE DoF of the target feature. Line#18 states a position on the feature defined in Line#3, with value of Ø0.54. Material condition modifier (RFS) is applied to the target feature. The primary datum (a Line feature defined in Line#2) and the secondary datum (a Plane feature defined in Line#7) are both referenced at RFS. Line#19 states the DoF information related to the tolerance defined in Line#18: the primary datum Line feature (defined in Line#2) controls the target feature's TDOF along [1, 0, 0], TDOF along [0, 1, 0], RDOF around [1, 0, 0] and RDOF around [0, 1, 0]; the secondary datum Plane feature (defined in Line#7) controls the target feature's TDOF along [0, 0, −1]. Similar interpretation applies to Line#20 and Line#21; Line#22 and Line#23; Line #24 and Line#25. Note that different DoFs of the target feature are controlled by its datums when the primary datum and the secondary datum type swapped (i.e. datum precedence).

In *Section E* (Line#26–Line#27), the assembly hierarchy information is provided. Line#26 states an assembly, named 'assembly_0', is composed of one part defined in Line#1. Line#27 states that the whole CTF-Graph Based model is composed of parts or assemblies defined in Line#26.

Note that the output format can be programmed to suit the application's use or requirements.

GDT testbed

Using the SCTF-Graph Based Model, a tolerance analysis testbed has been developed; its main modules are shown in Fig. 24. As a whole, the SCTF-Graph Based Model is a GD&T representation model with validation functionality. With the input of the hybrid attributed CAD model, the neutral GD&T representation model, i.e. the SCTF-Graph Based Model, is automatically. This serves as the common data model for all types of tolerance analyses, e.g. automatic charting, simulation-based analysis and T-Maps based analysis. Using this Testbed, we can conduct tolerance analyses using different approaches on the same problem and compare the results. However, the comparison study, theoretical and quantitative, is published in a different paper. In this paper, we would like to show a specific example of toleranced part to demonstrate the neutral model itself rather than its application.

Recall the SCTF-Graph Based Model contains the T-Graph, which is a directed graph representing the datum-target relationship between different features in the Model. This directed graph facilitates downstream GD&T processing like tolerance analysis. The CTF-Graph Based model for this part shown in Fig. 26 is illustrated in Fig. 25.Note that

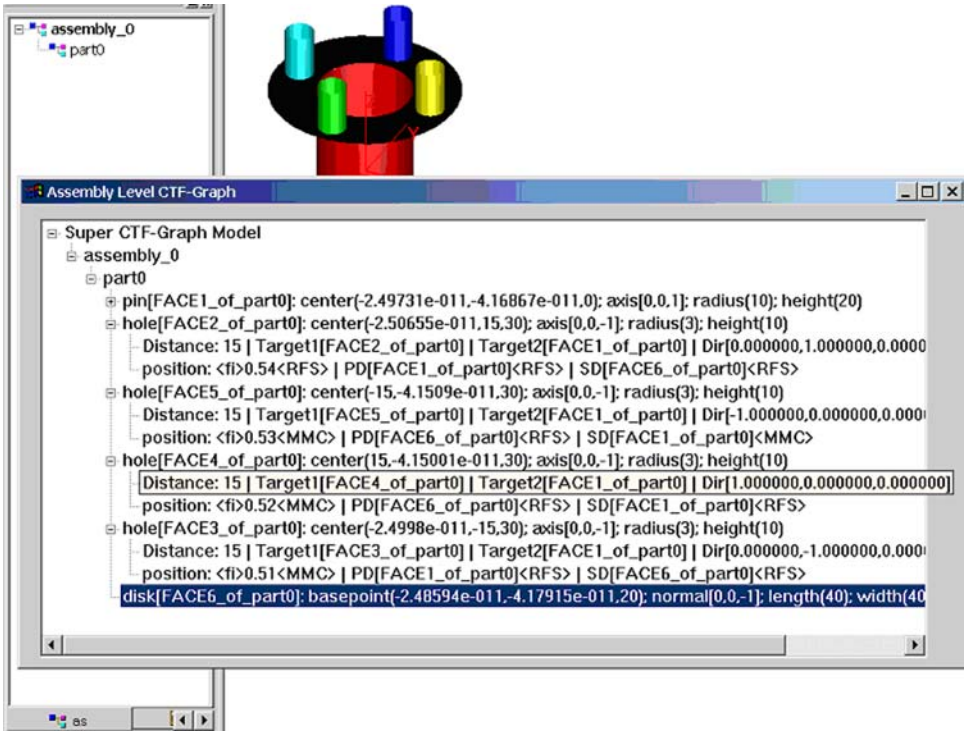


Fig. 22 The SCTF-Graph Based Model for the part shown in Fig. 21

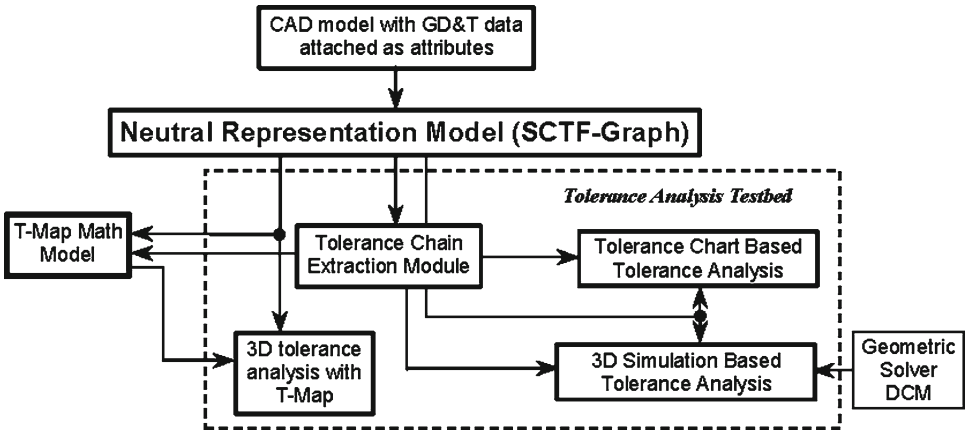
Fig. 23 The output textual file for the part shown in Fig. 21

```
#0=FILE('E:\ASU_GDTtestbed\sat\DOF_3_y14p5_DRF_part_new.sat');
#1=PART('part0', #2, #3, #4, #5, #6, #7);
#2=PIN('FACE1_of_part0', (0, 0, 0), [0, 0, 1], 10, 20);
#3=HOLE('FACE2_of_part0', (0, 15, 30), [0, 0, -1], 3, 10);
#4=HOLE('FACE5_of_part0', (-15, 0, 30), [0, 0, -1], 3, 10);
#5=HOLE('FACE4_of_part0', (15, 0, 30), [0, 0, -1], 3, 10);
#6=HOLE('FACE3_of_part0', (0, -15, 30), [0, 0, -1], 3, 10);
#7=CIRCULAR_PLANE('FACE6_of_part0', (0, 0, 20), [0, 0, -1], 40);
#8=CST_DISTANCE(15, #3, #2);
#9=METRIC_RELATIONSHIP(#8, CST_DISTANCE, (15, #3[LINE(axis of HOLE)], #2[LINE(axis of PIN)]));
#10=CST_DISTANCE(15, #4, #2);
#11=METRIC_RELATIONSHIP(#10, CST_DISTANCE, (15, #4[LINE(axis of HOLE)], #2[LINE(axis of PIN)]));
#12=CST_DISTANCE(15, #5, #2);
#13=METRIC_RELATIONSHIP(#12, CST_DISTANCE, (15, #5[LINE(axis of HOLE)], #2[LINE(axis of PIN)]));
#14=CST_DISTANCE(15, #6, #2);
#15=METRIC_RELATIONSHIP(#14, CST_DISTANCE, (15, #6[LINE(axis of HOLE)], #2[LINE(axis of PIN)]));
#16=T_SIZE(#2, (FI, 0.5, RFS));
#17=DOF(#16, (SIZE_DOF, SHAPE_DOF));
#18=T_POSITION(#3, (FI, 0.54, RFS), PD(#2, RFS), SD(#7, RFS));
#19=DOF(#18, (#2, TDOF[1, 0, 0], TDOF[0, 1, 0], RDOF[1, 0, 0], RDOF[0, 1, 0]), (#7, TDOF[0, 0, -1]));
#20=T_POSITION(#4, (FI, 0.53, MMC), PD(#7, RFS), SD(#2, MMC));
#21=DOF(#20, (#7, RDOF[1, 0, 0], RDOF[0, 1, 0], TDOF[0, 0, -1]), (#2, TDOF[1, 0, 0], TDOF[0, 1, 0]));
#22=T_POSITION(#5, (FI, 0.52, MMC), PD(#7, RFS), SD(#2, RFS));
#23=DOF(#22, (#7, RDOF[1, 0, 0], RDOF[0, 1, 0], TDOF[0, 0, -1]), (#2, TDOF[1, 0, 0], TDOF[0, 1, 0]));
#24=T_POSITION(#6, (FI, 0.51, MMC), PD(#2, RFS), SD(#7, RFS));
#25=DOF(#24, (#2, TDOF[1, 0, 0], TDOF[0, 1, 0], RDOF[1, 0, 0], RDOF[0, 1, 0]), (#7, TDOF[0, 0, -1]));
#26=ASSEMBLY('assembly_0', #1)
#27=MODEL(#26)
```

in reality, the arcs just contain the pointers to the corresponding objects (e.g. constraints, tolerances, mating conditions), which, in turn, hold pointers to the target features and datum features. This way, the travel can be two-way.

With such a model, it is trivial to find the target-datum relationship of all the features involved. If we want to perform a tolerance analysis, say a simulation-based analysis, on this part, the simulation can start from the primary datum of the part (not a specific tolerance), i.e. the datum feature that does not reference any other features as datums. Then the simulation continues upwards along the target-datum relationship, in one or more paths, until all the involved features are varied. In this example, the simulation starts with A, then it follows the paths: $A \rightarrow C$ and $A \rightarrow B \rightarrow D$. All intermediate simulation data are saved in arrays for computing the analysis results.

Fig. 24 System architecture



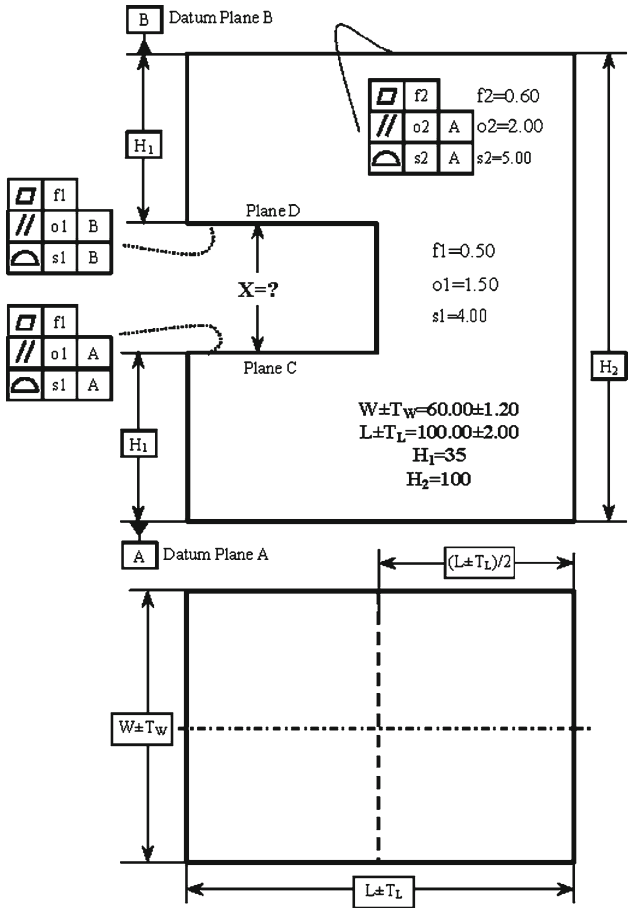


Fig. 25 A U-shaped block with GD&T specification

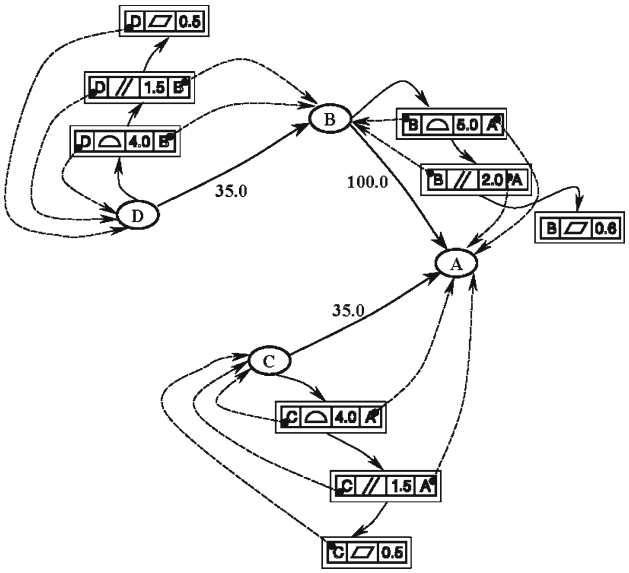


Fig. 26 Partially-expanded CTF-Graph for the part shown in Fig. 25

In the scenario of a multi-chain assembly, the variation of features within a part is performed first, and then all features within one part are treated as one rigid set, which will participate the assembly level constraint solution process. Note that in the SCTF model, mating conditions and assembly level constraints are all treated as special constraints, and they are built in the graph with no problem. Over-constraints may occur in multi-chain assemblies, but then their priorities need to be specified to find a solution. But an over-constrained multi-chain assembly should be avoided in design as much as possible.

Conclusion

This paper has developed a neutral GD&T representation model, the SCTF-Graph, which overcomes the shortcomings of the other models that are method-specific. This lean model holds just enough information that is needed for the representation and use of the data. To be specific, this neutral model contains all the geometric, constraint, assembling, and tolerance information needed for different types of tolerance analyses. This neutral model can be used for any types of tolerance analysis, making the analysis itself independent of the GD&T modeling.

Acknowledgements This study was supported in part by the US National Science Foundation Grant *DMI-9821008* and Grant

DMI-0245422. The opinions expressed in the paper are those of the authors and not endorsed by the US NSF.

References

ASME, Standard. (1994). *Dimensioning and tolerancing, ASME Y14.5M-1994*. New York: American Society of Mechanical Engineers.

Chase, K. W., Magleby, S. P., & Gao, J. S. (2000). Tolerance analysis of 2-D and 3-D mechanical assemblies with small kinematic adjustments, Retrieved from http://adcats.et.byu.edu/WWW/Publication.

Clément, A., Riviére, A., & Serre, P. (1995). A declarative information model for functional requirement. In *Proceedings of the 4th CIRP Design Seminar*, April 5–6, 1995, pp. 1–16, Tokyo, Japan.

Desrochers, A., & Maranzana, R. (1995). Constrained dimensioning and tolerancing assistance for mechanisms. In *Proceedings of the 4th CIRP Design Seminar*, April 5–6, 1995, pp. 17–30, Tokyo, Japan.

Desrochers, A., & Riviere, A. (1997). A matrix approach to the representation of tolerance zone and clearances. *Source, International Journal of Advanced Manufacturing Technology, 13*(9), 630–636.

Hillyard, R. C., & Braid, I. C. (1978). Analysis of dimensions and tolerances in computer aided mechanical design. *Journal of Computer Aided Design, 10*(3), 161–166.

Jayaraman, R., & Srinivasan, V. (1989). Geometric tolerancing: I. Virtual boundary requirements. *IBM Journal of Research and Development, 33*(2), 90–104.

Johnson, R. H. (1985). Dimensioning and tolerancing—Final report, R84-GM-02–2, CAM-I. Arlington, Texas, USA.

Kandikjan, T., & Shah, J. (1998). A computational model for geometric dimensions and tolerances consistent with engineering practice. In *Proceedings of DETC'98*, September 13–16, 1998, Atlanta, GA.

Kandikjan, T., Shah, J., & Davidson, J. (2001). A mechanism for validating dimensioning and tolerancing schemes in CAD systems. *Journal of Computer Aided Design, 33*, 721–737.

Krishnan, K. K., Eyada, O. K., & Ong J. B. (1997). Modeling of manufacturing processes characteristics for automated tolerance analysis. *International Journal of Industrial Engineering, 14*(3), 187–196.

Maeda, T., & Tokuoka, N. (1995). Toleranced feature modeling by constraint of degree of freedom for assignment of tolerance. In *Proceedings of 4th CIRP Design Seminar* April 5–6, 1995, pp. 89-103, Tokyo, Japan.

Ranyak, P. S., & Fridshal, R. (1988). Features for tolerancing a solid model. *ASME Computers in Eng. Conf.*, Vol. 1, pp. 262–274.

Requicha, A. A. G. (1983). Toward a theory of geometric tolerancing. *International Journal of Robotics Research, 2*(4), 45–60.

Rivest, L., Fortin, C., & Desrochers, A. (1993). Tolerance modeling for 3D analysis-presenting a kinematic formulation. In *Proceedings of 3rd CIRP Seminars on Computer Aided Tolerancing*, April 27–28, 1993, pp. 51–74, France.

Roy, U., & Liu, C. R. (1993). Integrated CAD frameworks: Tolerance representation scheme in a solid model. *Computers & Industrial Engineering, 24*(3), 495–509.

Roy, U., & Fang, Y. C. (1996). Tolerance representation scheme for a three-dimensional product in an object-oriented programming environment. *IIE Transactions, 28*, 809–819.

Shah, J. J., & Miller, D. (1990). A structure for supporting geometric tolerances in product definition systems for CIM. *Manufacturing Review, 3*(1), 23–31.

Shah, J., Yan, Y., & Zhang, B.-C. (1998). Dimension and tolerance modeling and transformations in feature based design and manufacturing. *Journal of Intelligent Manufacturing, v9*(5), 475–488.

Shen, Z. (2005) *Development of a framework for a set of computer-aided tools for tolerance analysis*. Ph.D. thesis, Dept. of Mech. & Aerospace Eng., Arizona State Univ, Tempe, AZ.

Shen, Z., Ameta, G., Shah, J. J., & Davidson, J. K. (2005a). A comparative study of tolerance analysis methods, ASME Transactions. *Journal of Computing & Information Science in Engineering, 5*(3), 247–256.

Shen, Z., Shah, J. J., & Davidson, J. K. (2005b). A complete variation algorithm for slot and tab features for 3D simulation-based tolerance analysis. In *CDROM Proceedings of DETC'05*, September 25–28, Long Beach, CA.

Shen, Z., Shah, J. J., & Davidson, J. K. (2005c). Simulation-based tolerance and assemblability analyses of assemblies with multiple pin-hole floating mating conditions. In *CDROM Proceedings of DETC'05*, September 25–28, Long Beach, CA.

Shen, Z., Shah, J. J., & Davidson, J. K. (2006). Virtual part arrangement in automatic tolerance chart based tolerance analysis at the assembly level. In *CDROM Proceedings of DETC'06*, September 10–13, 2006, Philadelphia, PA.

Shen, Z., Shah, J. J., & Davidson, J. K. (2008). Automatic generation of min/max tolerance charts for tolerance analysis from CAD models. *International Journal of Computer-Integrated Manufacturing*. (in press).

Tsai, J. C., & Cutkosky, M. R. (1997). Representation and reasoning of geometric tolerances in design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 11*, 325–341.

Turner, J. U. (1993). A feasibility space approach for automated tolerancing. *Journal of Engineering for Industry, 115*(3), 341–346.

Wu, Y. Y. (2002). *Development of mathematical tools for modeling geometric dimensioning and tolerancing*. Ph.D. thesis, Dept. of Mech. & Aerospace Eng., Arizona State Univ., Tempe, AZ.

Wu, Y. Y., Shah, J. J., & Davidson, J. K. (2003). Computer modeling of geometric variations in mechanical parts and assemblies. ASME Transactions. *Journal of Computing & Information Science in Engineering, special issue on GD&T, 3*(1), 54–63.

Zhang, B. C. (1992). *Geometric modeling of dimensioning and tolerancing*. Ph.D. thesis, Dept. of Mech. & Aerospace Eng., Arizona State Univ, Tempe, AZ.