1991

Automating Tolerance Synthesis: A Framework and Tools

Stephen C.-Y. Lu

Robert G. Wilhelm

Published in *Journal of Manufacturing Systems* 10:4 (1991), pp. 279–296; doi: 10.1016/0278-6125(91)90023-U

Published online August 21, 2008.

Automating Tolerance Synthesis: A Framework and Tools

Stephen C.-Y. Lu and Robert G. Wilhelm

University of Illinois at Urbana-Champaign, Urbana, Illinois, USA

Abstract

This paper describes CASCADE-T—a new approach to tolerance synthesis that uses a complete representation of the conditional tolerance relations that exist between features of a part under design. Conditional tolerances are automatically determined from functional requirements and shape information. Tolerance primitives based on the virtual boundary requirements approach to tolerance representation are composed to form more complex tolerance relationships. Artificial intelligence techniques, including a constraint network, frame-based system, and dependency tracking are used to support flexible and detailed computation for tolerance analysis and synthesis.

Keywords: tolerancing, synthesis, geometric modeling, mechanical tolerance analysis, concurrent product and process design, computer-aided design, computer-aided manufacturing, artificial intelligence, constraint-based reasoning

Introduction

A geometric tolerance describes the degree to which a nominal design feature can vary while satisfying functional requirements. For example, a lever transmits force when the applied force exceeds equilibrium conditions and the lever is sized to carry the transmitted force. If a designer requires a lever that transmits forces in the range of one to ten pounds, the minimum lever cross-section will often be determined by the worst-case loading condition of this design requirement.

All lever cross-sections that exceed this minimum will satisfy the design requirement. If the designer chooses the minimum cross-section as nominal, larger cross-sections would be acceptable because they satisfy the design requirement. In this example, the area of the cross-section is a geometric feature and the force range is a functional requirement. The tolerance on the nominal feature would be –O and +oo.

A geometric tolerance also describes the variation that is allowed when fabricating parts from a design. Manufacturing processes have particular and repeatable ranges of variation. The tolerance associated with a process depends on expected process variation and the techniques used for measurement. It is most cost effective to choose geometric tolerances and manufacturing processes so that allowable design variations are larger than expected manufacturing variations. In this example, the positive tolerance on the nominal feature, +oo, would be reduced once a fabrication process is selected.

Using slightly different interpretations, designers and manufacturing engineers utilize the same kinds of information, geometric features, and tolerances to establish designs and fabrication plans. As computer-aided design and manufacturing tools improve, designers and manufacturing engineers will be able to share and concurrently modify this geometric information. In this paper, computer-based representation and synthesis of geometric tolerances is presented as a means by which design and manufacturing requirements, stated via geometric tolerances, can be shared by various product engineers. Extending recent work in the theory of geometric tolerances and artificial intelligence techniques, this approach provides consistent methods for sharing information and shows potential for reducing design and manufacturing costs by automatically detecting tolerancing errors and determining cases where tolerances can be relaxed.

Geometric tolerances are generated by considering the functional requirements for a part and relating these requirements to the geometry of the part under design. The current practice in many design activities is to specify only those tolerances deemed important. Each tolerance is determined via engineering analysis[1] or by using standards[2,3] developed for common parts and assemblies. All remaining tolerances are then determined according to defaults associated with the drawing or fabrication process.[4] The specified tolerances, along with all of the default tolerances, form a tolerance specification that describes all of the variations allowed in the geometry of a part. This approach is expedient but allows several different errors.

Unchecked default tolerances may allow variations that contradict the intentions of the designer. As well, tolerances specified by the designer may be unattainable in manufacturing. These are errors of *validity* since the tolerance specification describes parts that do not meet design or manufacturing requirements.

A precise tolerance may be specified with respect to a geometric feature that is toleranced more loosely. This is an error of *consistency* since variations allowed by one tolerance do not coincide with variations allowed by another. At best, consistency errors result in tighter tolerances and higher costs, but invalid specifications may also occur.

When a geometric feature is not toleranced at all, manufacturing planning decisions may lead to geometric variations that do not satisfy the functional requirements of the design. This is an error of *sufficiency* since the tolerance specification does not completely communicate design and manufacturing requirements. This error takes on greater importance in computer approaches to tolerance synthesis where redundant checking may not be performed.

In current practice, these errors are corrected during design reviews. Prevention is difficult, however, because design and planning documents do not contain information that explicitly connect tolerances with functional requirements and manufacturing capabilities.

Formal theories of geometric tolerancing[5–8] provide a good starting point for the development of computer-based tools that prevent the tolerancing errors described earlier. The approach described in this paper is based on such a theory of geometric tolerancing. The approach and computation methods are part of a computer-based tolerance synthesis system called concurrent computer-automated methods for the synthesis of competing design 280 elements and geometric tolerances (CASCADE-T). Synthesis is emphasized because CASCADE-T allows tolerance specifications to be generated and checked automatically during the design process.

A tolerance synthesis system uses part geometry (the geometric description of a part) and functional requirements as input, and results in tolerance specifications. This task is difficult because many different solutions are equally acceptable, and checking procedures are not well understood for all geometric features. Tolerance synthesis can be computationally intensive as well since many different geometric features must be checked each time an additional geometric feature or detail is added to the design.

CASCADE-T employs several artificial intelligence (AI) techniques to address these problems. A detailed account is maintained of functional requirements, the relationships among geometric features that are implied, and checking computations. Computations that check for consistency, validity, and sufficiency are done only for related geometric features. Hence, the overall computational requirements are reduced. This detailed account also provides a consistent method for the sharing of engineering knowledge among designers and manufacturing engineers. From the detailed account, several different but equally accurate interpretations of the part can be presented.

Review of Related Work

Computer-based approaches to tolerance analysis and synthesis fall into several different categories. The earliest work focused on the manipulation of dimensions and the analysis of tolerance specifications. Later approaches have accommodated dimensions, representation of tolerance information, techniques for testing feasibility, and synthesis for prespecified geometric models. More recent work has focused primarily on the underlying theory of tolerancing, composition, and properties such as validity and sufficiency.

In many computer-aided design (CAD) systems, it is common to preserve some information about dimensions and tolerances. For tolerances or dimensions specified on perfect-form shapes, Hillyard & Braid[9] define the notion of an admissible dimensioning scheme where each dimension is neither overdefined nor underdefined. Procedures to test for admissible dimensions are described and a sensitivity analysis is shown for the relationship between the nominal shape and small changes in the dimensions. Gossard et al.[10] describe a system for dimension-driven geometry where changes in part dimensions, specified by the designer, are automatically translated into changes in the part geometry. The shape of a part can be refined in steps, incrementally, as dimensions are established. These systems use a perfect-form characterization for parts under design and have limited support for tolerance information. Nonetheless, their use of incremental refinement and measures for admissible dimensions has influenced the design of CASCADE-T.

The PADL solid modeler has been extended to maintain information and relationships[11] that describe tolerance specifications. This work demonstrates the utility of maintaining consistency among tolerance variables such as allowed variation for dimensions, and solid model attributes such as nominal dimensions of primitive solids. In the analysis of electrical networks, nominal and bounding values for component behavior are often specified with limited accuracy. An approach using fuzzy sets[12] demonstrates how calculations over sets of quantities are useful in these circumstances. Refinements of these approaches are used in CASCADE-T.

Many different improvements have been suggested for the representation of tolerance specifications. Ranyak and Fridshal[13] have developed a hierarchical approach to feature modeling, and a dimension and tolerance model for representing tolerance specifications. Their implementation can represent many of the ANSI tolerances and has been demonstrated with a process planning application. Roy and Lui[14] propose a hybrid representation that combines constructive solid geometry, boundary representation, and structured face-adjacency graphs. Their model can provide multiple levels of abstraction and support reasoning about relationships between geometric features. Etesami[15] describes a method for including manufacturing information within the part model. Each part feature is linked with a bounding solid constructor—a volume that encloses the measured part surface. These constructors and their accompanying information can then verify that manufactured parts meet tolerance specifications. With further development, it is suggested that this approach will automate inspection procedures used for verification. Shah and Miller[16] discuss the requirements of representing tolerance specifications and describe an implementation that enhances the system developed by Ranyak.[13] Each of these systems seeks to provide adequate support for the ANSI standard and provide information that would be appropriate for tolerance analysis. Analysis capabilities are not provided in these systems.

Working with part geometry and dimensions, a theory of tolerance modeling[8] has been implemented by Turner and Wozny with a CAD system to choose the best tolerances for particular dimensions. In this approach, linear approximations and linear optimization techniques are employed. The focus here is on modeling methods that are reasonably accurate and computationally tractable. However, in some cases the linear approximation requirements are too limiting.

Two recently proposed theories of tolerancing provide a good base for further work. The issues behind Requicha's effort[6,17] serve as a starting point for CASCADE-T. Several key differences concerning composition and computability, however, favor the virtual boundary requirements (VBR) approach described by Jayaraman and Srinivasan.[5,7] In particular, Requicha's work defines composite features that can't be uniformly handled, procedures for measured entities that do not work well with nonconvex shapes, and incomplete conditions for position tolerance. Further, for particular classes of requirements, the VBR approach defines necessary and sufficient conditions for acceptable tolerances that are quite useful for tolerance synthesis.

While much tolerance analysis is concerned with worst-case conditions, part assemblies and statistical requirements must also be considered. In analyzing an assembly of parts, it is common to add up the tolerance zones for adjacent parts. This sum of tolerances, or stack-up, describes the total variation for the assembly. In practice, statistical approaches are often employed when specifying tolerance stack-up for assemblies since the risk of parts that can't be assembled may be small compared to the cost of tight tolerances. Recent work in this area includes more accurate models for estimating tolerance stack-up[18] by considering process characteristics. Although CASCADE-T directly addresses worst-case tolerance stack-up, these practical requirements have influenced the representation detail and connections to manufacturing requirements provided in CASCADE-T.

Computer-Automated Tolerance Synthesis

When CAD systems include tolerance synthesis procedures, geometric tolerances may be automatically inferred from functional requirements and shape information for the part under design. The object of the work described in this paper is to provide such tolerance synthesis procedures in a manner similar to the procedures currently available for building complex geometric solids from combinations of primitive solids. Final tolerance specifications may then be synthesized according to available manufacturing processes, vendor specifications, and other performance goals such as cost or reliability.

Previous computer-based approaches to tolerance specification have focused on representation and analysis, that is, how tolerance information is stored and whether specified tolerances satisfy the functional requirements. Procedures for the synthesis of tolerance specifications require additional knowledge and reasoning. For example, in determining tolerances for a complex mechanical assembly , one must connect functional requirements with design features, satisfy all relationships between requirements and design features, and insure that the tolerances prevent undesirable performance. Engineers specifying tolerances may use specialized knowledge and consider only portions of the total design. The tolerance specification produced is considered acceptable if it satisfies functional requirements and falls within manufacturing capabilities. A specification is arrived at by incremental refinement in which particular details are added and then checked as well as by global synthesis in which many different parameters are chosen together to maximize a particular objective.

Clearly, the current efforts in computer-aided tolerance specification only begin to address these requirements for tolerance synthesis. CASCADE-T uses a framework of seven computing and data representation tasks to support computer-based tolerance synthesis. The framework, as shown in Figure 1, includes tolerance representation, links to functional requirements, validity, sufficiency, explanation, composition, and synthesis. As the figure shows, computational elements such as sufficiency and validity calculations, and data elements such as geometric data for nominal solids in the design are required. Additionally, some data elements (indicated by hatched-lines) depend on the data that is present as well as procedures previously selected for calculation. For example, the data elements in the tolerance representation are valid only when validity and sufficiency conditions are satisfied.

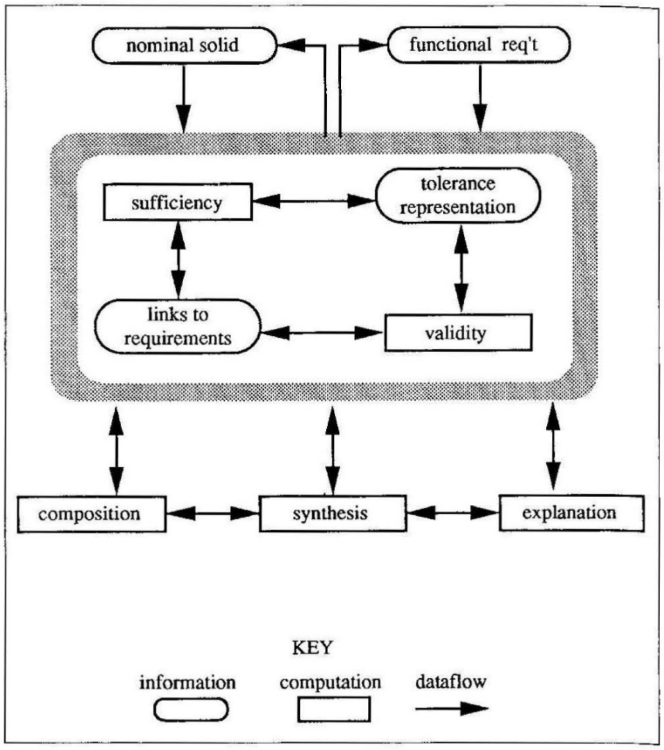


Figure 1. CASCADE-T: A Framework for Tolerance Synthesis

It is the interaction and integration among these representation and computing tasks that contribute most significantly to automatic tolerance synthesis; hence, a framework is emphasized rather than a progression of required software functionality. The remainder of this section describes each part of the framework. Additional sections then detail how CASCADE-T can be implemented to automate tolerance synthesis.

Tolerance Theory

A representation theory for tolerancing should provide unambiguous and consistent descriptions of tolerances. Further, such a theory should furnish adequate information to test for errors in a tolerance specification. Minimally, a tolerance theory[6] allows for the description of nominal solids, restatement of the nominal solids as features pertinent to tolerancing, and the description of geometric relationships or tolerance assertions that must hold in order for tolerances to be satisfied.

Most often, nominal solids such as prisms and cylinders are restated as a collection of surfaces that bound the material-side of the object that is represented. Tolerance assertions are then posed relative to these surfaces or the two half-spaces that each surface defines. Tolerance specifications posed in this way describe a volume or class of variations that encloses the boundary of each part that satisfies the tolerances.

When using a tolerance specification based on such a representation theory, of immediate concern is the question of conformance; that is, for a particular specification and given a manufactured part is the part in the variational class described by the specification? The tolerance representation must maintain knowledge about geometric shapes and functional requirements as well as gaging criteria to test for conformance.

Distinction is often made between parametric and nonparametric representation for tolerance specifications. The former directly uses the representation of nominal solids to define perfect form tolerance zones. The latter generates tolerance zones by taking the difference between offsets of nominal solids. The solids are used for the difference, but their representation does not influence the calculation. Nonparametric approaches are more general and avoid some ambiguity problems. Nonetheless, some manner of parameterization is required to efficiently measure for conformance. It has been suggested by Jayaraman and Srinivasan[5,7] that a conversion is necessary to practically support measurement of conformance. Using their VBR approach, geometric surfaces and relationships between surfaces describe the tolerance specification. The description is then converted to conditional tolerances to provide for measurement of conformance. The conversion, while applicable to any set of design features, is generally done for commonly occurring features such as primitive geometric solids.

This representation scheme provides several interesting advances in computer-aided design environments. Each conversion results in a conditional tolerance relationship that describes some collection of functional requirements for a set of geometric surfaces. CASCADE-T extends this approach to provide tolerance primitives that allow these conditional tolerances to be used by designers while specifying tolerance relationships. Additionally, tolerance specifications can be synthesized automatically from a set of these primitives.

Of particular note, the VBR approach allows for incremental composition of, and reasoning about, tolerance specifications. Beyond the conformance question, the designer may now use the tolerance representation during a design session to determine whether the design satisfies the tolerance assertions, how the design can be changed to satisfy tolerance assertions, and how the tolerance assertions can be changed to accommodate the design.

CASCADE-T also extends the system described by Requicha and Chan.[11] Their approach used a variational graph embedded in a constructive solid geometry modeler. A facility for specifying and storing tolerance information was provided. Using tolerance primitives, conditional tolerance equations (parts of a specification) are generated according to each primitive and its associated attributes of the solid. As discussed in the section on constraint networks, propagation of tolerance variables and solid model parameters is handled consistently each time a value is set or changed.

Linking Requirements to Geometry

To infer geometric tolerances from functional requirements, links must be established between the two. Jayaraman[5] has shown that for some general functional requirements such as the ability to assemble and material bulk, general relationships and validation criteria can be proven. Ability to assemble means that all surfaces bounding a part can be positioned to satisfy a mating specification, that is, cup is against base, boss is within hole, etc. Note that the ability to assemble does not always imply that mating can be successfully achieved while avoiding spatial interference. This is a more difficult problem as discussed in reference 19. Material bulk refers to requirements on volume of material present for mechanical properties, for orientation of volume, and for fit or interaction between volumes.

When functional requirements are described using VBRs, conditional tolerance relationships may be derived to link functional requirements with geometric tolerances. The resulting relationships also describe procedures for testing the validity of the tolerance specification. In some cases, however, it may be useful to specify requirements where complete validation criteria are not known. For design, these include performance requirements and operating ranges, as well as induced and allowable loads. For manufacturing, these requirements include process distributions and limitations of processing steps or sequence.

CASCADE-T allows functional requirements to be linked to geometric tolerances at three levels of detail. Tolerance primitives have particular functional requirements embedded within them that are complete by definition. That is, each functional requirement implies certain geometric relationships, and the presence of these geometric relationships provides sufficient evidence that the functional requirement is satisfied. Additional functional requirements may also be defined and added using the constraint network. In this case, the functional requirement implies certain constraints, but the presence of a particular collection of constraints may not imply a particular functional requirement. Finally, performance criteria, operating ranges, and other requirements may be enforced during synthesis.

Validity

A valid tolerance specification describes a variational class that includes the object under design. Equally important, the tolerance specification must not include any requirements for perfect form or exact position. That is, variations of 0.0 are not valid since they cannot be attained. In practice, avoidance of perfect form and exact position are replaced by avoidance of unattainable tolerances.

When evaluating the variational class for validity, two questions must be answered. Does the class represent a valid solid? and does the class describe only parts that satisfy the functional requirements? While the former can be treated as a strict geometric problem, the latter can be evaluated only by considering the particular requirements of the design. In many cases, design requirements specified as VBRs map directly to primitive solids used in a solid modeling environment. When this occurs, the relationships between functional requirements and tolerance variables can be handled directly. In other cases, a solid will be implied but not directly represented by the tolerance specification. Here, a solid must be generated as the union of the volume defined by the tolerance specification and the volume defined by the nominal solids. This union is then evaluated for validity.

When functional requirements are linked with part geometry, requirements for perfect form and position can be immediately detected. While it is often useful to correct design details that are the immediate cause of these unattainable requirements, for any nontrivial design it is equally important to consider earlier decisions that have contributed to such a conflict. It may be more fruitful to slightly modify several parameters rather than insist on a tight tolerance for one parameter.

Perfect form and position requirements are invalid tolerances and can be prevented by enforcing validity conditions on all tolerances that are specified for a design. CASCADE-T supports the enforcement of validity conditions via constraint propagation. At all times, a description of all feasible solutions is maintained by working with intervals of real values. Additionally, an explanation facility provides information to correct invalid specifications that occur as designs or manufacturing plans are developed.

Sufficiency

A sufficient tolerance specification describes bounded variational classes. An unbounded variational class implies several problems. When an unbounded variational class describes one feature, the position or at least one dimension of the feature is not determined—any value is acceptable. When two or more features are described by an unbounded variational class, there is opportunity to confuse not only size and position but form as well. An unbounded variational class results from an underconstrained specification—from incompleteness or error.

A criteria for sufficiency is not currently known for the specification of tolerance. Requicha[6] suggests that sufficiency can be determined by checking that each variational class is strictly bounded and restricted in position.

Once again, to support tolerance synthesis, knowledge of the equations and variables influencing an unbounded variational class is required. For example, beyond evaluating a variational class, it is useful to know what parameters must be determined to bound the variational class and whether changes to these parameter will influence other bounded variational classes.

CASCADE-T provides this kind of knowledge through an explanation facility. The detailed representation provided by a frame and constraint system flexibly maintains the information required for this type of reasoning.

Composition and Coupling

In the course of a design, a variety of primitives or tolerance requirements are associated with the part under design. In practice, it is common to add dominant requirements early and then check the feasibility of these requirements as more detailed items are added. An acceptable composition includes a set of requirements that are simultaneously satisfied.

In some cases, as requirements are added, couplings come into play and must be handled. Figure 2 illustrates a simple example of this—two peg-and-hole fits each have independent conditional tolerance relations until they are defined relative to a common datum. Sharing the same datum, an additional conditional tolerance exists between the orientation parameters of the two fits.

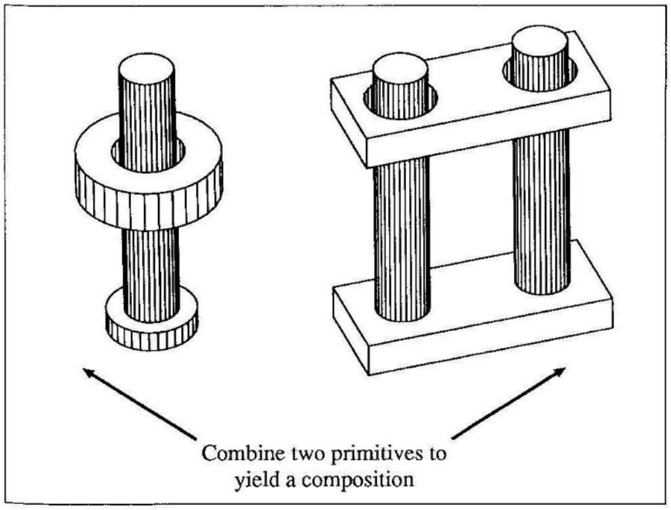


Figure 2. Coupling between Two Features or Primitives

In CASCADE-T, this kind of condition results in additional tolerance requirements being added to the representation after descriptions have been added by the user. This presents two different problems. First, it is necessary to monitor the design underway for coupling conditions and add additional tolerance requirements; later, as the design changes, it may be necessary to remove them. A frame system is used to provide database, pattern matching, and rule-based techniques to address this problem. The dependency tracking capability is employed to allow for sound recovery when coupling conditions are removed during a design. Second, additions made without user intervention must be understandable to the user, which the explanation facility provides.

Explanation

Tolerance specifications, complex on their own, are only part of the very sophisticated and interconnected representation used during product design and manufacturing. These specifications are developed and used by many different engineers of different disciplines. While the specifications evolve over time, they are likely to be geometrically incomplete since sufficiency may be enforced only for dominant design features. These characteristics demand consistency management and facilities for abstraction or explaining the representation in varying degrees of detail.

To support tolerance synthesis, explanations such as how a value was determined, how a value could be determined, what parameters will be affected by changing a particular parameter, what parameters caused an invalid solid or variational class, and what parameters must be set or changed to produce a sufficient variational class must be provided.

When unique relationships are available, the explanation for a value can be thought of as the inverse of the relations or equations that yielded the value. For example, if position p_1 is determined via $p_1 = p_2 + p_3$ then p_1 can always be stated as the sum of p_2 and p_3. However, if $p^2{}_1 = p_2$, p_1 could be stated as the square root of p_2 only when $p_2 \geq 0$. In this case, where the relationships are not unique, the calculation order and input values must be used to provide an explanation.

Roy and Lui[14] discuss the necessity for varying levels of abstraction when representing tolerances and show methods for maintaining particular low-level entities and determining particular high-level features. CASCADE-T extends this abstraction capability from predefined hierarchies to any useful relationship described by the user.

Tools for Tolerance Synthesis

In meeting the challenges posed by tolerance synthesis, CASCADE-T relies heavily on several key contributions in tolerance theory and artificial intelligence. The theory and approach of Jayaraman and Srinivasan[5,7] are used for tolerance representation, and tolerance primitives are derived to represent the conditional tolerance relations that must hold for parameterized geometric features. Functional requirements such as ability to assemble, performance requirements, and material bulk are either included in the conditional tolerance relations or added with additional equations. Composition of tolerance primitives is supported by building and enforcing valid connecting relationships.

The representation and computation problems of tolerance synthesis call for detailed, flexible computing. A detailed representation of tolerance relations and connections is required, and validity must be enforced. At the same time, it is important to provide an environment that flexibly supports common engineering design activities, including incremental refinement and interactions between several engineers. Most importantly, the system must be easy to use—the details of the calculations need not be the immediate concern of users. A system of frames and constraints is used to represent and calculate tolerance relations. Experience with this approach[20] suggests that natural user interaction methods can be supported while maintaining a detailed representation. Other artificial intelligence tools including dependency-tracking and explanation facilities are also used.

Tolerance Primitives

Tolerance primitives result from the conditional tolerance relations that can be derived for parameterized geometric features. Currently, a small number of primitives—such as pin and hole, key and slot, and adjacent planes (as shown in Figure 3)—are used in CASCADE-T. More primitives will be added to provide a library of primitives that designers may work with.

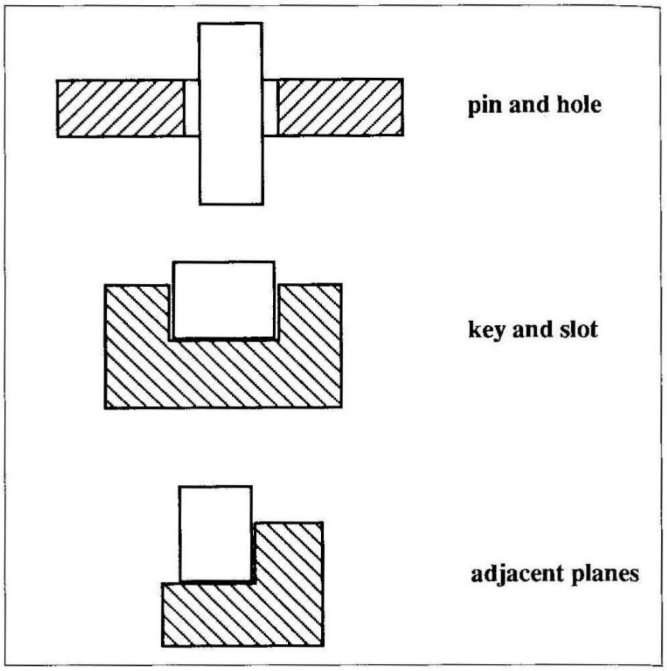


Figure 3. Tolerance Primitives

For each primitive, a parameter space is first defined, and conditional tolerance relationships between parameters are then derived. Currently, this effort is external to the software of CASCADE-T. This approach differs significantly from the composite features that Requicha has proposed[6] since each tolerance primitive represents detailed relations between parameters that may limit the breadth of composition.

A primitive includes data and procedures. Data is in the form of a parametric representation, associated datums, and conditional tolerance relationships that must be maintained for the parameters of the primitive. Procedures include the methods for arriving at values for parameters.

For example, consider the two-dimensional peg and hole system shown in Figure 4 as analyzed by Srinivasan and Jayaraman.[7] The parameters maintained for the primitive include locational parameter c_1, orientation parameter c_3, nominal diameter S_{N1}, actual diameter S_1, and length or slab parameter l_1. The primary datum is also stored.

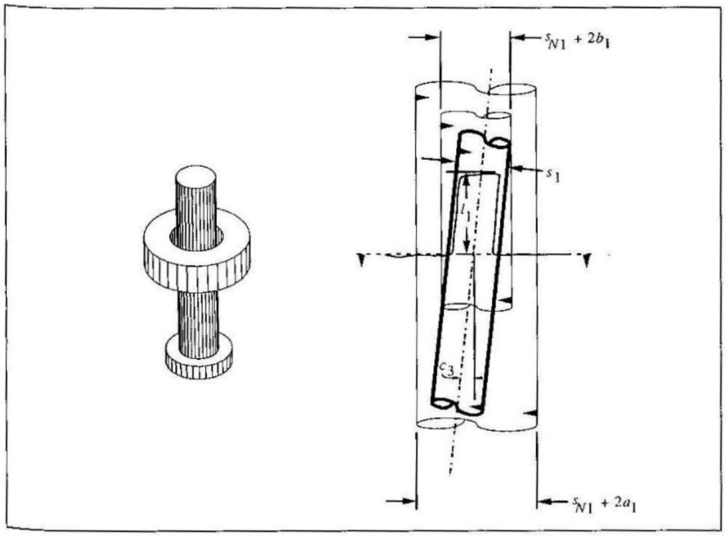


Figure 4. The Peg and Hole Tolerance Primitive

Methods for arriving at values for orientation relations include:

$$0 \leq c_3 \leq 2\, tan^{-1}\left[\frac{\frac{(s_{N1}+2\,|a_1|)-s_1}{l_1}}{1+\sqrt{1+\left(\frac{(s_{N1}+2|a_1|+s_1)}{l_1}\right)\left(\frac{(s_{N1}+2|a_1|-s_1)}{l_1}\right)}}\right] \qquad (1)$$

For tolerance synthesis, it is sometimes useful to determine allowable tolerances, such as a_1 when other parameters such as c_3 are known. As this may entail calculating an input variable from a specified output variable, the procedure for calculation is called an inversion. In simple cases, inversion may be accomplished by solving each equation for each variable in the equation. This is not always possible, however, for tolerance calculations. In some cases, difficult trigonometric equations may be solved by considering only first quadrant values or small values in general. In other cases, incremental calculations allowed by a constraint network are used to find inverse values.

Frame-Based Description

The tolerance representation of CASCADE-T must maintain many details of data and calculation methods for each tolerance primitive. In some cases, more complex objects and relations are composed from the primitive representation. In light of these requirements, a frame-based representation is employed[21] to provide the following benefits. Descriptions for different types of data and accompanying calculations are done only once—improving ease of use and ensuring consistency. Hierarchies of data types can be defined so that one data type can be defined as a specialization of another. Default values are easily handled, and only differences from the default values are actually stored for each occurrence or instance of a data type. Actual data type instances are identified uniquely. Last, powerful pattern matching is provided with flexibility equivalent to the general unification of Prolog while allowing for optimization of particular matches. This frame-matching facility can be used to compare datums associated with different instances of primitives and to recognize couplings.

Computation via Constraint Network

As tolerance primitives are associated with features of a part under design, more and more relations (or equations) must hold among the parameters of each primitive, interconnected primitives, and functional requirements specified for the part or assembly. CASCADE-T represents all of these relations completely throughout the course of the design sequence. As dimensions and properties are specified for part features, calculations may be done using the relations and values that are known.

In some sense, this approach follows the general idea of least-commitment described by Marr.[22] The main goal, however, is to support the common problem-solving strategies employed by engineers in which unknowns are determined from dominant relations and known quantities, while details are added late and checked for validity.

There are several computational difficulties inherent with this approach. Flexibility is required for many different relations where some will be added or deleted and some will be changed. An efficient mechanism is also needed to track all relations. Information must be maintained to determine what to calculate and when—avoiding complete recalculation. Finally, calculations may occur from either side of an equation.

To address these requirements, CASCADE-T employs a constraint propagation network similar to that described in reference 23. One or more constraints are defined for each relation and then enforced for each tolerance primitive. The constraint network is composed of constraint nodes, value nodes, and connections. A constraint may only touch values, and values may only touch constraints.

A constraint is defined according to data used for calculations and the different relations that must hold between the data. For example, the multiplication constraint definition shown in Table 1 has three data elements referred to as *m*1, *m*2, and *product*. The calculation rules that include *product* may be calculated when *m*1 and *m*2 are known. The element *m*1 may be calculated when *product* and *m*2 are known, and several zero relationships that force a zero *product* when one multiplicand is known to be zero.

Table 1. Multiplication Constraint Definition

```
(defprim  (multiplier   (* / / ))   (product  ml  m2)
    (product  (m1)   (if  (csend m1 zerop)   i0  @dismiss))
    (product  (m2)   (if  (csend m2 zerop)   i0  @dismiss))
    (product  (ml  m2)   (csend m1  *  m2))
    (m2  (m1  product)   (if  (not  (csend m1 zerop))
                              (csend product  /  m1)
                              @dismiss))
    (m1  (m2  product)   (if  (not  (csend m2 zerop))
                              (csend product  /  m2)
                              @dismiss)) )
```

The standard set of constraints used in CASCADE-T includes the constraints shown in Figure 5. Each symbol on the left side of the figure corresponds to the mathematical operation typed to the right. The lines, or pins, emanating from each symbol define the number of parameters involved in the calculation. Beyond this standard set, additional constraints may be added at any time during system development and use by stating the required data elements and calculating rules. The rules must be consistent but need not be complete.

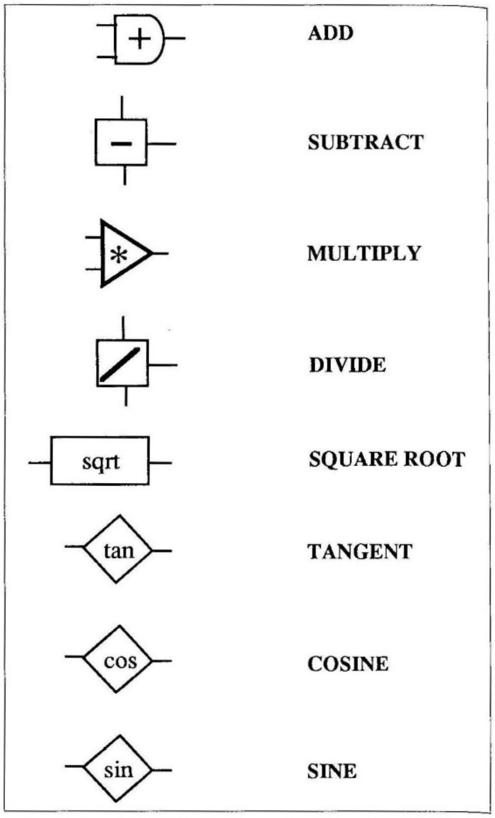


Figure 5. Common Constraints and Symbols

Value nodes are defined by parameters of the tolerance primitives and by intermediate calculation values in the constraint network. For example, when a new multiplication operation is specified, three value nodes are automatically generated for the three multiplication data elements. Value nodes are also generated automatically for each tolerance primitive.

Constraints may be defined at several different levels. Currently, most of the constraint types describe arithmetic and trigonometric functions. The tolerance primitive relationships are then built up as networks of these primitives. These tolerance primitive relationships can be described with a single constraint of many parameters.

The equations that represent part of the peg and hole primitive are shown as a constraint network in Figure 6. For example, the left-hand side of the figure corresponds to the subexpression

$$\frac{(S_{N1}) + 2|a_1| - S_1}{l_1}$$

from Equation 1. Defining and managing these detailed relationships manually would be quite tedious. However, by using the facilities of the frame and constraint environment, the details of defining instance after instance of each primitive are handled automatically.

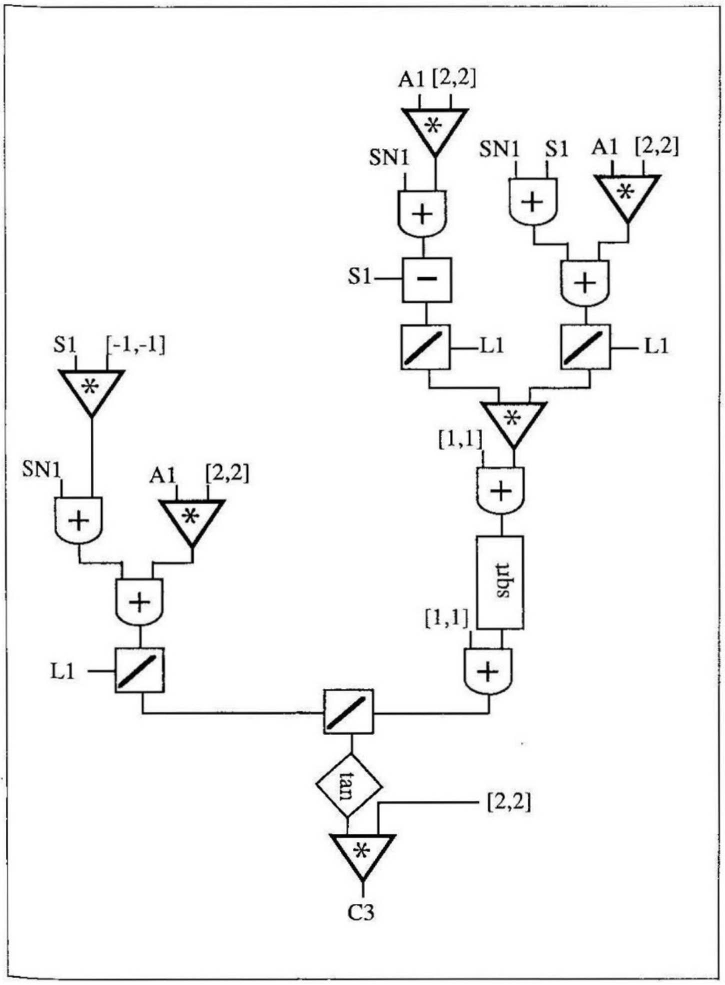


Figure 6. Constraints Describing Peg and Hole Relationships

Calculations for the various tolerance primitives are carried out by propagating values through the network of constraints and values. For example, the length of a feature might be specified by the user. When a value is set for a feature, the calculation rules of neighboring data elements are checked. New values are calculated when appropriate and then set. This process continues until no more new values may be set.

Interval Values

Numerical computation may be done with a variety of different types of numbers including integers, real numbers, sets of numbers, and ranges of values. CASCADE-T uses intervals (or ranges of real values) for several reasons. First, early in the design, the goal of tolerance synthesis is to keep track of all good solutions and avoid choosing design features that are impossible to build. The emphasis is on maintaining a feasible solution space. Until all calculations are complete, it is useful to maintain extensive information about all possible combinations. Second, when working with the inexact goals and imperfect forms of manufactured parts, the specification of ranges is often a more natural way to describe features and dimensions. Third, the representation maps directly to tolerancing information and collapses easily into single real or integer numbers. A bidirectional tolerance of ± .01 on a nominal of 1.5 can easily be expressed as [1.49, 1.51] with respect to the nominal of [1.5, 1.5]. More importantly, early in a design this tolerance could be represented as [1.4, 1.6], allowing many different smaller ranges to be considered in all calculations.

Many computations used frequently with real numbers and vectors can also be applied to intervals.[24,25] However, some properties of real numbers do not extend to intervals. Solutions to multiplication equations, such as $A * X = B$, may not be unique. For example, in $[-2,1) * [x_1, x_2] = [-4, 2]$, two different interval solutions, [2, 2] and [–1, 2] are possible. Further, the distributive law applies only for bounding intervals and does not guarantee unique solutions. In CASCADE-T some of these ambiguities are handled with calculation rules. Remaining questions are approached by considering the way that calculations have proceeded to yield a value-making use of dependency information.

Dependency Tracking

The conditional tolerance equations used in CASCADE-T and the intervals used for calculations do not guarantee unique solutions. When multiple solutions are possible, the order of calculation, which is strictly determined by the designer, will influence the calculated results or the selection of one nonunique solution over another. A dependency tracking system is used in CASCADE-T to manage these complex issues and maintain information useful for explaining exactly how to arrive at calculations and relations.

Dependency systems, or truth-maintenance systems, have been used for several different purposes in artificial intelligence and engineering design applications. On a wider scale, a truth-maintenance system can support the recording of design rationale.[26] In CASCADE-T, a truth-maintenance system[27] is used to track dependencies between calculated values, record how values are set and calculated, and provide detailed explanations of calculations.

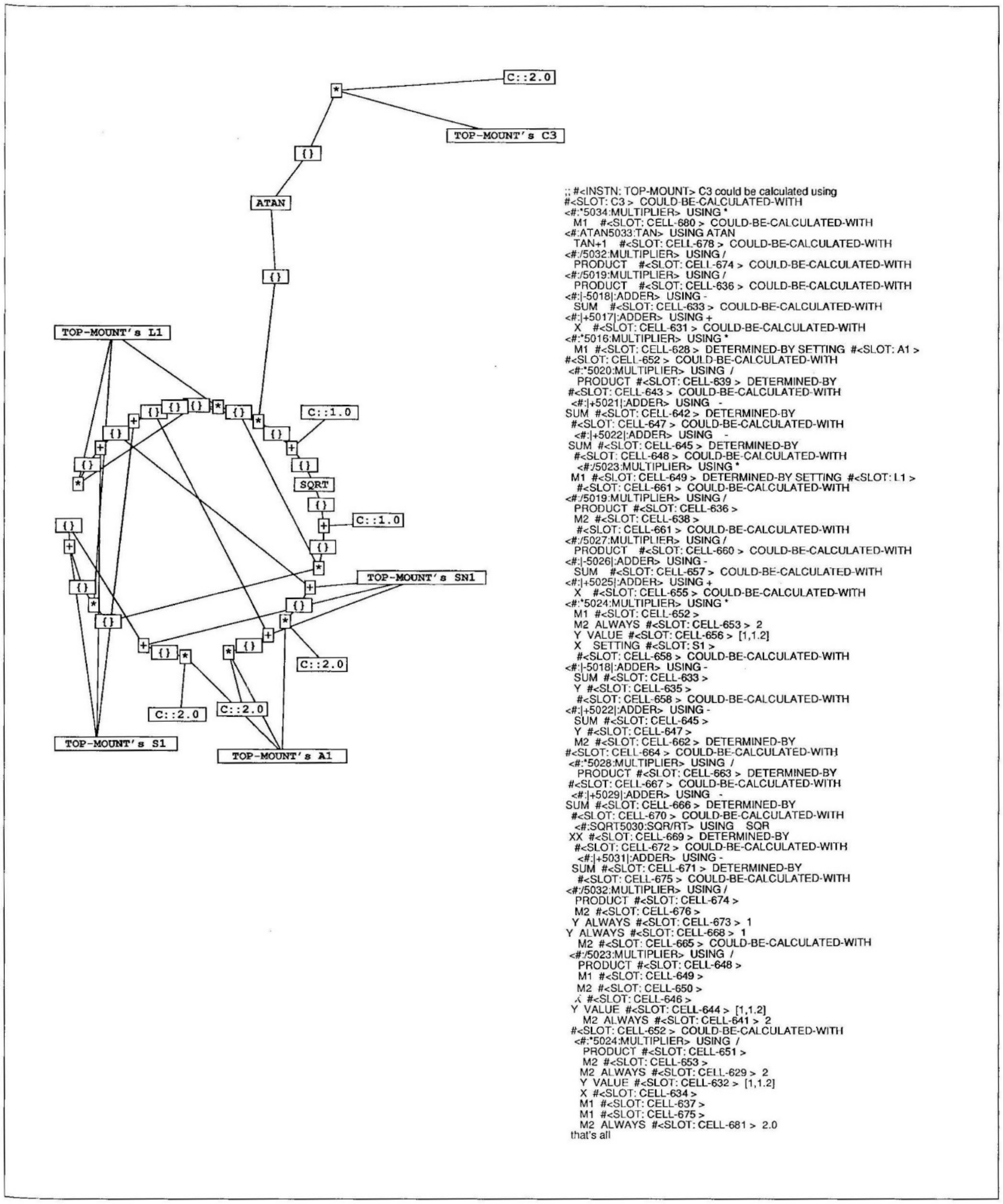
For the peg example, an allowed orientation might be calculated according to a length specified for the peg. Later, by changing the peg length, the orientation tolerance is also influenced. By recording the fact that the calculated orientation depends on the specified length, if changes are made to the peg length, the dependency system will insure that the orientation tolerance is recalculated as well.

When there is no possible solution to the conditional tolerance equations associated with a design, some portion of the current specification must be changed. For example, several catalog parts with very loose tolerances might be added to an assembly and produce unacceptable tolerance requirements for a mating part in the assembly. At this point in the design, some part of the specification must be withdrawn and respecified. Dependencies indicate which values or specifications caused a problem and all of the calculations that must be undone when part of the design is changed. This is of particular concern when there are nonunique solutions for the conditional tolerance equations.

Explanation Facility

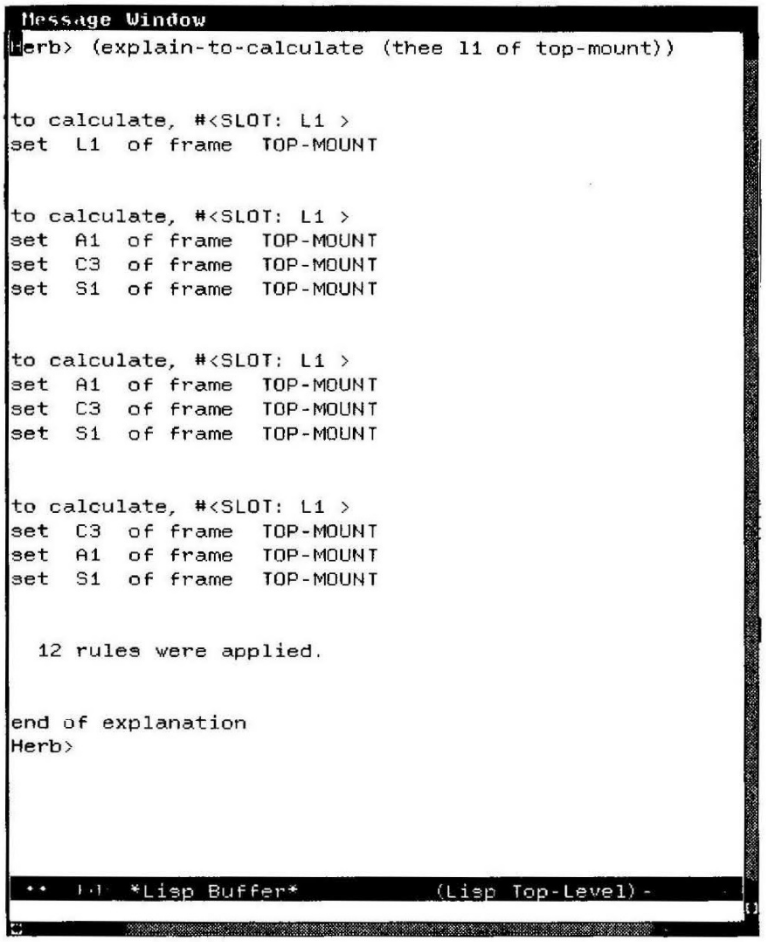
CASCADE-T maintains a computer-based representation that preserves the minute detail upon which conditional tolerances are founded. Obviously, for any interesting part, the description quickly grows quite complex. To maintain a complete description and present information that various engineers can quickly understand, an explanation facility is provided to generate descriptions from the detailed representation. According to the task at hand and the engineer involved, different types of explanations can be generated.

Figure 7 shows the relations and values that exist when a peg tolerance primitive is added to the design, assuming that nominal diameter alone has been specified, and demonstrates the degree of detail that is maintained. The graph shows all of the constraints associated with the tolerance primitive. Each box or node represents either a constraint or a variable. Each line or arc represents a connection between inputs and outputs of the constraints. The text provides a written explanation of the graph. Since this level of detail might not be useful to many engineers, CASCADE-T provides rule-based mechanisms for generating simpler explanations. Figure 8 shows the simplest explanation for setting the length of the peg along with several other possible calculations. In this case, the repeated explanations with $A1$, $C3$, and $S1$ correspond to three different orderings of calculation that could be used to arrive at $L1$, the length. Similar explanations are also available to describe how values have been calculated. This is quite useful when changes are being considered for a design or manufacturing plan.



```
;; #<INSTN: TOP-MOUNT> C3 could be calculated using
#<SLOT: C3 >  COULD-BE-CALCULATED-WITH
<#:*5034:MULTIPLIER>  USING *
  M1    #<SLOT: CELL-680 >  COULD-BE-CALCULATED-WITH
<#:ATAN5033:TAN>  USING ATAN
  TAN+1    #<SLOT: CELL-678 >  COULD-BE-CALCULATED-WITH
<#:/5032:MULTIPLIER>  USING /
  PRODUCT    #<SLOT: CELL-674 >  COULD-BE-CALCULATED-WITH
<#:/5019:MULTIPLIER>  USING /
  PRODUCT    #<SLOT: CELL-636 >  COULD-BE-CALCULATED-WITH
<#:|-5018|:ADDER>  USING -
  SUM    #<SLOT: CELL-633 >  COULD-BE-CALCULATED-WITH
<#:|+5017|:ADDER>  USING +
  X    #<SLOT: CELL-631 >  COULD-BE-CALCULATED-WITH
<#:*5016:MULTIPLIER>  USING *
  M1  #<SLOT: CELL-628 >  DETERMINED-BY SETTING  #<SLOT: A1 >
#<SLOT: CELL-652 >  COULD-BE-CALCULATED-WITH
 <#:*5020:MULTIPLIER>  USING  /
   PRODUCT  #<SLOT: CELL-639 >  DETERMINED-BY
 #<SLOT: CELL-643 >  COULD-BE-CALCULATED-WITH
  <#:|+5021|:ADDER>  USING   -
SUM  #<SLOT: CELL-642 >  DETERMINED-BY
 #<SLOT: CELL-647 >  COULD-BE-CALCULATED-WITH
  <#:|+5022|:ADDER>  USING   -
 SUM  #<SLOT: CELL-645 >  DETERMINED-BY
  #<SLOT: CELL-648 >  COULD-BE-CALCULATED-WITH
   <#:/5023:MULTIPLIER>  USING *
 M1  #<SLOT: CELL-649 >  DETERMINED-BY SETTING  #<SLOT: L1 >
   #<SLOT: CELL-661 >  COULD-BE-CALCULATED-WITH
<#:/5019:MULTIPLIER>  USING /
  PRODUCT  #<SLOT: CELL-636 >
  M2  #<SLOT: CELL-638 >
   #<SLOT: CELL-661 >  COULD-BE-CALCULATED-WITH
<#:/5027:MULTIPLIER>  USING /
  PRODUCT    #<SLOT: CELL-660 >  COULD-BE-CALCULATED-WITH
<#:|-5026|:ADDER>  USING -
  SUM    #<SLOT: CELL-657 >  COULD-BE-CALCULATED-WITH
<#:|+5025|:ADDER>  USING +
  X    #<SLOT: CELL-655 >  COULD-BE-CALCULATED-WITH
<#:*5024:MULTIPLIER>  USING *
  M1  #<SLOT: CELL-652 >
  M2  ALWAYS  #<SLOT: CELL-653 >  2
  Y  VALUE  #<SLOT: CELL-656 >  [1,1.2]
  X    SETTING  #<SLOT: S1 >
   #<SLOT: CELL-658 >  COULD-BE-CALCULATED-WITH
<#:|-5018|:ADDER>  USING -
  SUM  #<SLOT: CELL-633 >
  Y  #<SLOT: CELL-635 >
   #<SLOT: CELL-658 >  COULD-BE-CALCULATED-WITH
<#:|+5022|:ADDER>  USING -
  SUM  #<SLOT: CELL-645 >
  Y  #<SLOT: CELL-647 >
  M2  #<SLOT: CELL-662 >  DETERMINED-BY
#<SLOT: CELL-664 >  COULD-BE-CALCULATED-WITH
 <#:*5028:MULTIPLIER>  USING  /
   PRODUCT  #<SLOT: CELL-663 >  DETERMINED-BY
 #<SLOT: CELL-667 >  COULD-BE-CALCULATED-WITH
  <#:|+5029|:ADDER>  USING   -
SUM  #<SLOT: CELL-666 >  DETERMINED-BY
 #<SLOT: CELL-670 >  COULD-BE-CALCULATED-WITH
  <#:SQRT5030:SQR/RT>  USING    SQR
 XX  #<SLOT: CELL-669 >  DETERMINED-BY
  #<SLOT: CELL-672 >  COULD-BE-CALCULATED-WITH
   <#:|+5031|:ADDER>  USING -
 SUM  #<SLOT: CELL-671 >  DETERMINED-BY
   #<SLOT: CELL-675 >  COULD-BE-CALCULATED-WITH
<#:/5032:MULTIPLIER>  USING /
  PRODUCT  #<SLOT: CELL-674 >
  M2  #<SLOT: CELL-676 >
 Y  ALWAYS  #<SLOT: CELL-673 >  1
Y  ALWAYS  #<SLOT: CELL-668 >  1
  M2  #<SLOT: CELL-665 >  COULD-BE-CALCULATED-WITH
 <#:/5023:MULTIPLIER>  USING  /
   PRODUCT  #<SLOT: CELL-648 >
   M1  #<SLOT: CELL-649 >
   M2  #<SLOT: CELL-650 >
  X  #<SLOT: CELL-646 >
 Y  VALUE  #<SLOT: CELL-644 >  [1,1.2]
    M2  ALWAYS  #<SLOT: CELL-641 >  2
 #<SLOT: CELL-652 >  COULD-BE-CALCULATED-WITH
  <#:*5024:MULTIPLIER>  USING  /
    PRODUCT  #<SLOT: CELL-651 >
    M2  #<SLOT: CELL-653 >
   M2  ALWAYS  #<SLOT: CELL-629 >  2
   Y  VALUE  #<SLOT: CELL-632 >  [1,1.2]
   X  #<SLOT: CELL-634 >
   M1  #<SLOT: CELL-637 >
   M1  #<SLOT: CELL-675 >
   M2  ALWAYS  #<SLOT: CELL-681 >  2.0
 that's all
```

Figure 7. Detailed Description of Peg Length Equations



```
Herb> (explain-to-calculate (thee l1 of top-mount))


to calculate, #<SLOT: L1 >
set  L1  of frame  TOP-MOUNT


to calculate, #<SLOT: L1 >
set  A1  of frame  TOP-MOUNT
set  C3  of frame  TOP-MOUNT
set  S1  of frame  TOP-MOUNT


to calculate, #<SLOT: L1 >
set  A1  of frame  TOP-MOUNT
set  C3  of frame  TOP-MOUNT
set  S1  of frame  TOP-MOUNT


to calculate, #<SLOT: L1 >
set  C3  of frame  TOP-MOUNT
set  A1  of frame  TOP-MOUNT
set  S1  of frame  TOP-MOUNT


  12 rules were applied.


end of explanation
Herb>
```

Figure 8. Explanations for Possible Length Calculations

Synthesis

In most cases, design and manufacturing functional requirements can be used to derive conditional tolerance relationships and feasible ranges for a tolerance specification. For nondominant design features, however, appreciable variation may be allowed. At the same time, available manufacturing processes as well as cost and reliability constraints may further limit these nondominant design features. A tolerance specification can be completed only after these requirements are considered.

Tolerance synthesis requires two different levels of computation. Design and manufacturing functional requirements can be imposed on the tolerance specification as design continues for better understood and stable processes, operating environments, and marketing conditions. In CASCADE-T, these conditions are handled with tolerance primitives. In other cases, when knowledge is incomplete, design and manufacturing requirements may be applied most efficiently as design steps are finished or at the end of the design process. These conditions can be handled with optimization techniques. While many of the equations of the conditional tolerance relations are nonlinear, by incrementally solving subproblems in the constraint network and using linear approximations when appropriate, tractable computations are possible.

In composing tolerance specifications with constraints and tolerance primitives, a feasible solution space is generated for each tolerance and design variable. This space reflects the conditional tolerance relations of the tolerance primitives. For reliable designs and efficient fabrication, these feasible ranges must then be further constrained according to available manufacturing processes, vendor specifications, and performance goals such as cost or reliability. This final selection, or synthesis, then provides a tolerance specification that conforms to functional requirements and may be used for manufacturing planning.

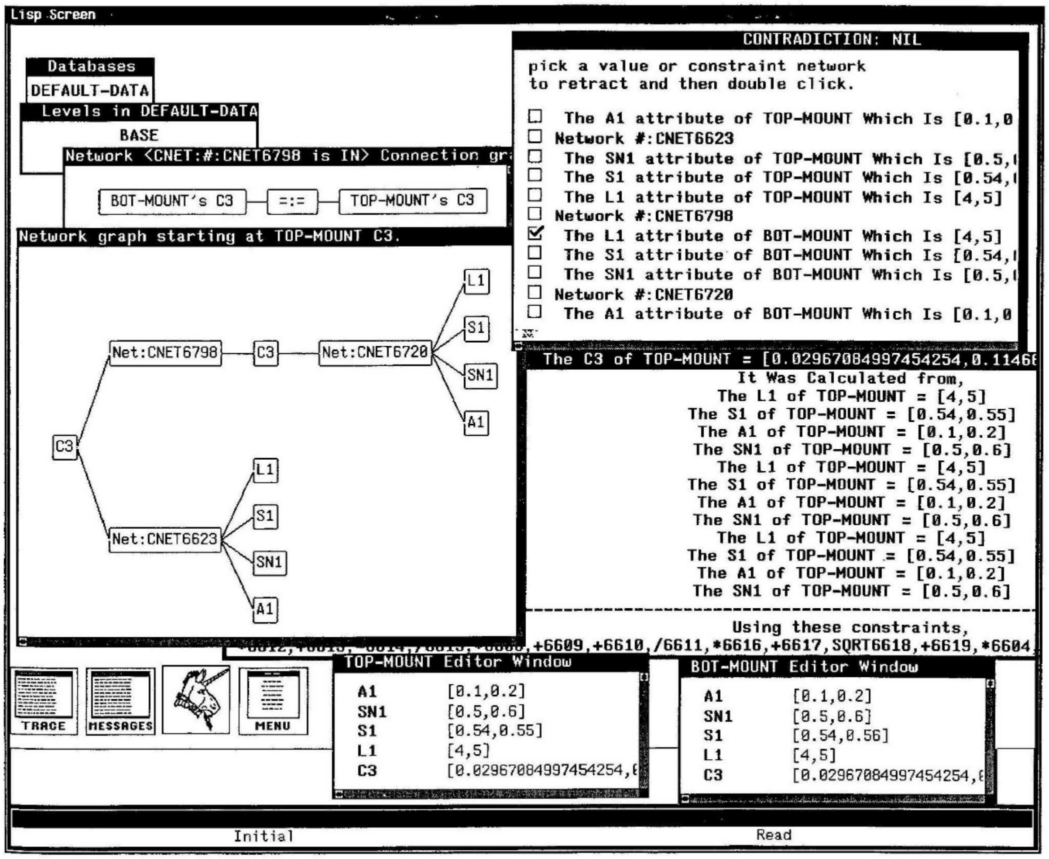
Implementation for a Design Environment

CASCADE-T is currently being implemented to support tolerance synthesis based on the framework and tools described earlier. The software design and development is guided by three principle goals. A very detailed and consistent representation of relationships between functional requirements and geometry is required. Incremental reasoning about tolerance requirements for geometric shapes or parts in mechanical assemblies must be supported. The system must also be easily integrated with a CAD modeler while independent of any particular modeling scheme.

A software environment that supports frame-based reasoning and constraint networks has been developed to support several different projects undertaken by the Knowledge-Based Engineering Systems Research Laboratory.[20,28] A truth-maintenance system provides dependency tracking. Calculations using real intervals are supported using object-oriented values. Recent extensions have been made to improve constraint propagation and dependency tracking as well as the explanation facility.

Tolerance primitives are represented as networks of constraints connected to frames associated with each instance of the primitive. Several primitives such as the peg and hole primitive (shown in Fig. 4) are now defined and used in the system. Graphical and text explanations are supported along with a rule-based mechanism for simplifying and directing explanations.

A typical display from the prototype system is shown in Figure 9. To the lower right, several frames display peg and hole tolerance primitives and parameters that may be manipulated by the user. To the upper right, a contradiction among competing requirements is shown. To the left center, a graphical display shows the connections for one of the tolerance primitives. To the right center, explanations are presented in a text window.



```
CONTRADICTION: NIL
pick a value or constraint network
to retract and then double click.

□  The A1 attribute of TOP-MOUNT Which Is [0.1,0
□ Network #:CNET6623
□  The SN1 attribute of TOP-MOUNT Which Is [0.5,
□  The S1 attribute of TOP-MOUNT Which Is [0.54,
□  The L1 attribute of TOP-MOUNT Which Is [4,5]
□ Network #:CNET6798
☑  The L1 attribute of BOT-MOUNT Which Is [4,5]
□  The S1 attribute of BOT-MOUNT Which Is [0.54,
□  The SN1 attribute of BOT-MOUNT Which Is [0.5,
□ Network #:CNET6720
□  The A1 attribute of BOT-MOUNT Which Is [0.1,0
```

```
 The C3 of TOP-MOUNT = [0.02967084997454254,0.1146
            It Was Calculated from,
          The L1 of TOP-MOUNT = [4,5]
      The S1 of TOP-MOUNT = [0.54,0.55]
       The A1 of TOP-MOUNT = [0.1,0.2]
       The SN1 of TOP-MOUNT = [0.5,0.6]
          The L1 of TOP-MOUNT = [4,5]
      The S1 of TOP-MOUNT = [0.54,0.55]
       The A1 of TOP-MOUNT = [0.1,0.2]
       The SN1 of TOP-MOUNT = [0.5,0.6]
          The L1 of TOP-MOUNT = [4,5]
      The S1 of TOP-MOUNT = [0.54,0.55]
       The A1 of TOP-MOUNT = [0.1,0.2]
       The SN1 of TOP-MOUNT = [0.5,0.6]
-------------------------------------------
            Using these constraints,
,+6609,+6610,/6611,*6616,+6617,SQRT6618,+6619,*6604
```

```
TOP-MOUNT Editor Window
A1      [0.1,0.2]
SN1     [0.5,0.6]
S1      [0.54,0.55]
L1      [4,5]
C3      [0.02967084997454254,
```

```
BOT-MOUNT Editor Window
A1      [0.1,0.2]
SN1     [0.5,0.6]
S1      [0.54,0.56]
L1      [4,5]
C3      [0.02967084997454254,
```

Figure 9. Typical Display in Current Implementation

In the next development stage, more sophisticated tools for choosing primitives will be added and the parameters for each primitive will be associated with features in a solid modeling system. As the system is developed, the following design scenario will be possible.

1. Shapes for several components of an assembly are chosen in the solid modeling environment by the designer.
2. Using a library of tolerance primitives that correspond to different kinds of shapes and functional requirements, the designer specifies relationships between assembly components.
    a) select appropriate features
    b) select appropriate datums
    c) connect features of solid and parameters of tolerance primitive
3. As tolerance primitives are chosen and associated with solids, governing equations and parameters are added to the constraint network and effects are propagated.

During the design session, the system will support several different kinds of responses to user queries and perform several kinds of monitoring.

1. As primitives are chosen, coupling conditions are automatically recognized and added to the constraint network.
2. Validity and sufficiency are monitored as the design progresses.
3. The designer may query for insufficiently constrained features and suggestions for accomplishing appropriate constraint. The system responds with useful explanations suited to the particular application.
4. When a functional requirement implies a perfect form or position, the system signals a contradiction and suggests different combinations of features that might be modified to correct the problem.

This system represents a first step in efficiently using a complex and consistent design representation for tolerance specification. With further development, additional enhancements will exploit this detailed representation. Functional requirements and nominal solid geometry can be linked to the detailed tolerance representation. In the preceding scenario, it was suggested that a designer participate actively to specify tolerance relationships by choosing many details—features for tolerancing, appropriate datums, and connecting relations. For particular combinations of geometric features and tolerance primitives, it is expected that parts of this task can be assisted or handled by mechanized knowledge sources operating on functional requirements and nominal solid geometry. In addition to merging constraints, the constraint network can be used to enforce validity requirements. The detailed representation can also be used for incrementally measuring sufficiency. Finally, feasible ranges for tolerance variables can be further constrained with optimization techniques to yield the best tolerance specifications. Though focused on synthesis, this system also supports analysis; as a part and its features are composed, a tolerance specification is generated that may then be compared with prespecified requirements.

Conclusion

CASCADE-T, a novel framework for addressing tolerance synthesis and analysis, has been described. At the heart of this framework is a very detailed description of the part under design, its functional requirements, and associated tolerance relations. Intervals and equations are used to describe each tolerance relation. Facilities for composition and synthesis are provided along with support for many different types of abstraction and explanation. A variety of artificial intelligence tools have been integrated to support the representation and computation requirements of CASCADE-T. Frames, rule-based reasoning, constraint networks, and dependency tracking are used.

Although this work is still in progress, it is useful to evaluate some of its fundamental contributions.

1. CASCADE-T is oriented toward synthesis of tolerance specifications and provides tolerance primitives that can be used to build tolerance specifications in a manner similar to the procedures available for building complex solids from combinations of primitive solids.
2. The constraint network flexibly represents a rich collection of tolerance relations. Most previous approaches have focused on static representation of a limited number of prototypes.
3. CASCADE-T maintains sets of feasible solutions and yields detailed equations that can be used by optimization methods to select tolerance specifications according to particular objectives.
4. While good methods for determining validity and sufficiency are not yet known, the constraint network and dependency tracking allow for the representation and use of all knowledge that is available.
5. CASCADE-T allows designers and planners to compose rather than evaluate tolerance specifications; design and manufacturing goals can be considered relative to each other.
6. Though the representation employed is very detailed and complex, explanation facilities insure that the knowledge held in the representation may be accessed and used effectively for many different types of engineering tasks.

This approach might also be used for other engineering problems and to enhance existing computer-aided engineering systems. Computer-based tools are commonly developed to enforce validity of prespecified design features. While these systems are useful for analysis, they do not adequately support synthesis. When using these systems, a designer is always faced with looking for functional requirements that will coincide with valid design features. In effect, the designer is asked to find a design problem that has one of the specified solutions.

The approach of CASCADE-T can extend this analysis-based paradigm when relationships between functional requirements and design features can be expressed with equations or sets. The detailed representation that is employed can be used to assist designers in determining which solutions are consistent with the design problem under consideration. Further manipulation of the constraint network can then be performed to choose an optimal design while remaining true to the original functional requirements.

Acknowledgments – We thank Allen Herman of KBESRL and Keith Hummel of Allied Signal Corporation who have developed much of the software that supports this work. We also acknowledge Dr. Placid Ferreira of UIUC and Dr. Ramesh Srinivasan of the IBM T. J. Watson Research Center who have provided several useful comments and suggestions for improvements. Finally, we thank all of the members of the Knowledge-Based Engineering Systems Research Laboratory who have contributed to our work through many interesting discussions of their related efforts.

This research was supported by the US Army Research Office under Contract DAAL 03-87-K-0006 with the University of Illinois Advanced Construction Technology Center, and by the National Science Foundation (DMC-86-57116). We thank Rockwell International Science Center for providing research facilities.

References

1. E. T. Fortini, *Dimensioning for Interchangeable Manufacture*, Industrial Press, 1967.
2. *Dimensioning and Tolerancing*, American Society of Mechanical Engineering, New York, NY, ANSI Yl4.5M, 1982.
3. H. G. Conway, *Engineering Tolerances: A Study of Tolerances, Limits, and Fits for Engineering Purposes, with Full Tables of All Recognized and Published Tolerance Systems*, Pitman and Sons, 1966.
4. J. L. Gadzala, *Dimensional Control in Precision Manufacturing*, McGraw-Hill, 1959.
5. R. Jayaraman and V. Srinivasan, "Geometric Tolerancing: I, Virtual Boundary Requirements," *IBM Journal of Research and Development*, Vol. 33, No. 2, March 1989, pp. 90–104.
6. A. G. Requicha, "Toward a Theory of Geometric Tolerancing," *The International Journal of Robotics Research*, Vol. 2, No. 4, 1983, pp. 45–60.
7. V. Srinivasan and R. Jayaraman, "Geometric Tolerancing: II, Conditional Tolerances," *IBM Journal of Research and Development*, Vol. 33, No. 2, March 1989, pp. 105–125.
8. J. U. Turner and M. J. Wozny, "A Mathematical Theory of Tolerancing," *Geometric Modeling for CAD Applications*, North Holland, 1988.
9. R. C. Hillyard and I. C. Braid, "Analysis of Dimensions and Tolerances in Computer-Aided Mechanical Design," *Computer-Aided Design*, Vol. 10, No. 3, May 1978, pp. 161–66.
10. D. C. Gossard, R. P. Zuffante, and H. Sakurai, "Representing Dimensions, Tolerances, and Features in MCAE Systems," *IEEE Computer Graphics and Applications*, Vol. 8, No. 2, March 1988, pp. 51–59.
11. A. G. Requicha and S. C. Chan, "Representation of Geometric Features, Tolerances, and Attributes in Solid Modelers Based on Constructive Geometry," *IEEE Journal of Robotics and Automation*, Vol. 2, No. 3, September 1986, pp. 156–66.
12. R. Jain, "Tolerance Analysis Using Fuzzy Sets," *International Journal of Systems Science*, Vol. 7, No. 12, 1976, pp. 1393–1401.
13. P. S. Ranyak and R. Fridshal, "Features for Tolerancing a Solid Model," *ASME Computers in Engineering Conference*, Vol. 1, August 1988, pp. 263–74.
14. U. Roy and C. R. Liu, "Feature-Based Representational Scheme of a Solid Modeler for Providing Dimensioning and Tolerancing Information," *Robotics and Computer-Integrated Manufacturing*, Vol. 4, No. 3/4, 1988, pp. 335–45.
15. F. Etesami, "Tolerance Verification Through Manufactured Part Modeling," *Journal of Manufacturing Systems*, Vol. 7, No. 3, 1988, pp. 223–32.
16. J. J. Shah and D. Miller, "A Structure for Supporting Geometric Tolerances for Computer Integrated Manufacturing," *Transactions of the North American Manufacturing Research Institution of SME*, May 1989, Dearborn, MI, pp. 344–51.
17. A. G. Requicha, "Representation of Tolerances in Solid Modeling: Issues and Alternatives," *Solid Modeling by Computers*, Plenum, 1984.

18. W. H. Greenwood and K. W. Chase, "A New Tolerance Analysis Method for Designers and Manufacturers," *Transactions of ASME, Journal of Engineering for Industry*, Vol. 109, No. 2, May, 1987, pp. 112–16.
19. T. C. Woo, "Automatic Disassembly and Total Ordering in Three Dimensions," *Intelligent and Integrated Manufacturing Analysis and Synthesis*, American Society of Mechanical Engineers, 1987, pp. 291–304.
20. A. E. Herman and S. C.-Y. Lu, "A New Modeling Environment for Developing Computer-Based Intelligent Associates with an Application in Machining Operation Planning," in *Transactions of the North American Manufacturing Research Institute of SME*, Dearborn, MI, May 1989, pp. 260–65.
21. M. Minsky, "A Framework for Representing Knowledge," *The Psychology of Computer Vision*, McGraw-Hill, 1975, pp. 211–77.
22. D. Marr, "Artificial Intelligence—A Personal View," *Artificial Intelligence*, Vol. 9, No. 1, 1977.
23. G. J. Sussman and G. L. Steel, "CONSTRAINTS—A Language for Expressing Almost-Hierarchical Descriptions," *Artificial Intelligence*, Vol. 14, No. 1, 1980, pp. 1–39.
24. G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, 1983.
25. R. E. Moore, *Interval Analysis*, Prentice-Hall, 1966.
26. S. C.-Y. Lu and J. B. Thompson, "A Distributed Artificial Intelligence Approach to Integrated Engineering Design," *The First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Tullahoma, TN, June 1988, pp. 438–46.
27. J. de Kleer, "An Assumption-Based TMS," *Artificial Intelligence*, Vol. 28, pp. 127–62, 1986.
28. A. E. Herman and R. G. Wilhelm, "The Integration of Frames and Constraints for Engineering Problem Solving," Technical Report 89-2 (in preparation), University of Illinois at Urbana-Champaign.

Author Biographies

Stephen C.-Y. Lu received his MS and PhD degrees from the Department of Mechanical Engineering and the Robotics Institute of Carnegie-Mellon University, Pittsburgh, PA. He is currently a tenured associate professor and the founding director of the Knowledge-Based Engineering Systems Research Laboratory in the Mechanical and Industrial Engineering Department at the University of Illinois at Urbana-Champaign (UIUC). He is also a research associate professor of the Computer Science Department and the Beckman Institute for Advanced Science and Technology at UIUC. His research interests are in the development of artificial intelligence–based techniques for advanced engineering automation and in the integration of these techniques with traditional engineering methods. Currently, he is developing knowledge processing technology to support various concurrent engineering and system management tasks. He has published over 100 technical papers, reports, and book chapters in this area and has served as a keynote speaker for several national and international conferences. Professor Lu is an associate editor of the *Journal of Engineering for Industry*, *ASME Transactions*, and the *International Journal of Computer-Integrated Manufacturing Systems*. He also serves on several editorial committees including *SME Transactions*, *Journal of Intelligent Manufacturing*, and the *International Journal of System Automation*. He has organized many technical conferences and workshops, and was the chairperson of the 1986 ASME Symposium on Knowledge-Based Expert Systems for Manufacturing and the 1989 ASME Symposium on Concurrent Product and Process Design. He is an active member of ASME, SME, IEEE, and AAAI, and is a corresponding member of the International Institute for Production Research (CIRP). He has served as a technical consultant and expert panelist to various industries and federal agencies. He received the Presidential Young Investigator (PYI) Award from the National Science Foundation in 1987, the Outstanding Young Manufacturing Engineer Award from the Society of Manufacturing Engineers in 1988, and was selected as an Outstanding Young Man of America in 1988. In 1990, he received the Xerox Senior Faculty Research Award from UIUC for his high-quality research accomplishments over the previous five years. In the same year, he was also appointed as a University Scholar for his excellent contributions to scholarly activities.

Robert G. Wilhelm is a PhD candidate in the Department of Mechanical and Industrial Engineering at the University of Illinois at Urbana-Champaign. His research interests include geometric modeling, tolerance analysis and synthesis, design theory, computer-aided design, knowledge-based systems, optimization, statistical process control, and manufacturing systems. He is currently associated with the Rockwell International Science Center and has previously worked for Cincinnati Milacron, NAVSEA, and several other engineering organizations. His engineering experience includes development of robotic systems for computer-aided construction and automated assembly, software development for computer-integrated manufacturing, manufacturing and process engineering for machine tool and computer controls manufacture, and structural engineering for ships and bridges. He received his BS in industrial engineering from Wichita State University in 1981 and his MS degree in Industrial Engineering from Purdue University in 1984. As a Rotary Foundation Fellow, during 1981 and 1982, Mr. Wilhelm studied the history of science and technology at the University of Leicester, Great Britain, and the Ironbridge Gorge Museum.