Automating dimensional tolerancing using Ripple Down Rules (RDR)

Ghassan Beydoun
University of Wollongong, beydoun@uow.edu.au

Achim Hoffmann
University of New South Wales

Ramsey Hamade
The American University of Beirut

Automating dimensional tolerancing using Ripple Down Rules (RDR)

Abstract

We propose to use a knowledge based approach to assist in mechanical design focusing on *dimensional tolerancing.* To illustrate our approach, we capture the knowledge which human designers utilize in order to specify dimensional tolerances on shafts and mating holes in order to meet desired classes of fit as set by relevant engineering standards. The software system we developed would help mechanical designers become more effective in the time-consuming dimensioning and tolerancing process of their designs in the future. In doing this, the paper makes a twofold contribution to the field of knowledge acquisition: firstly, interface was adjusted to receive mathematical functions with their specifications prior and during the KA process to propose an approach to exploit relationships among several classes with respect to certain numerical features of the cases in order to accelerate the convergence of the RDR knowledge acquisition process by generating artificial cases which are likely to trigger the addition of exception rules. Secondly, it introduces the above problem domain of determining suitable tolerances for mechanical parts in a design as a knowledge acquisition problem.

Keywords

Automating, Dimensional, Tolerancing, using, Ripple, Down, Rules, RDR

Disciplines

Physical Sciences and Mathematics

Publication Details

Beydoun, G., Hoffmann, A. & Hamade, R. (2010). Automating dimensional tolerancing using Ripple Down Rules (RDR). Expert Systems with Applications, 37 (7), 5101-5109.

Automating Dimensional Tolerancing using Ripple down Rules (RDR)

G. Beydoun [1], A. Hoffmann [2], R. F. Hamade[3]

1: beydoun@uow.edu.au, School of Information Systems and Technology (SISAT), Faculty of Informatics, University of Wollongong, Wollongong NSW 2522 Australia

2: achim@cse.unsw.edu.au, School of Computer Science and Engineering, University of New South Wales, Sydney NSW 2052, Australia

3: rh33@aub.edu.lb, Department of Mechanical Engineering, The American University of Beirut (AUB), P.O. Box 11-0236, Riad El-Solh, Beirut 1107 2020, Lebanon.

Abstract. We propose to use a knowledge based approach to assist in mechanical design focusing on *dimensional tolerancing.* To illustrate our approach, we capture the knowledge which human designers utilize in order to specify dimensional tolerances on shafts and mating holes in order to meet desired classes of fit as set by relevant engineering standards. The software system we developed would help mechanical designers become more effective in the time-consuming dimensioning and tolerancing process of their designs in the future. In doing this, the paper makes a two fold contribution to the field of knowledge acquisition: firstly, interface was adjusted to receive mathematical functions with their specifications prior and during the KA process to propose an approach to exploit relationships among several classes with respect to certain numerical features of the cases in order to accelerate the convergence of the RDR knowledge acquisition process by generating artificial cases which are likely to trigger the addition of exception rules. Secondly, it introduces the above problem domain of determining suitable tolerances for mechanical parts in a design as a knowledge acquisition problem.

Keywords: knowledge acquisition, automation, design, fits and tolerances, knowledge based systems, RDR.

1. Introduction

The theoretical and methodological foundation of artificial intelligence (AI) as applies to aspects of mechanical design is of great interest [1]. For example, the extended general design theory [2] by Tomiyama and Yoshikawa has served as a basis for the reasoning process in developing expert systems for computer-aided design and as relates to the organization of design knowledge in intelligent computer-aided design (CAD) environments [3]. The idea of enhancing the performance of design software in general -and CAD in particular- by empowering the tools with human-like rule-based reasoning remains of great interest [4-7].

Mechanical CAD systems represent parts by using geometric primitives, all of which describe ideal shapes. However, actual manufactured parts are necessarily imperfect approximations to those ideal shapes. Therefore, it is necessary to specify tolerancing information during design so that it can be decided whether a manufactured part is acceptably close to the designed ideal during inspection. In the US, the standard adopted by ANSI/ASME (American National Standards Institute / American society of mechanical engineers) Y14.5-1994 [8] defines a dimension as '*numerical value expressed in appropriate units of measure and indicated on a drawing and in other documents along with lines, symbols, and notes to define the size or geometric characteristic; or both, of a part or part feature'*. In regards to tolerancing, ANSI/ASME Y14.5-1994 specifies "that a dimensional tolerance is the total amount by which a specific dimension is permitted to vary from nominal".

Today, many methods of tolerancing are available for designers. Prime of the tolerancing methods are the dimensional tolerancing method and the geometric dimensioning and tolerancing (GD&T) method. While the former is the classical method which works in a fairly linear fashion, the more demanding GD&T methodology requires that the complete description of the part should contain meaningful geometric attributes: dimensions, tolerances, as well as factors of form (flatness, squareness, etc.) [9-11]. In this work, we consider only the knowledge related to dimensional tolerancing. At a later stage of this research, we aim to apply the methodology outlined here to the standards of geometric dimensioning and tolerancing (GD&T) where to have this 'GD&T symbology' become an integral part of the design stage similar to what is outlined in this paper in a fashion that reflects the design's true intent and functionality.

What designers typically refer to as the ‘tolerance budget’ represents the difference between the maximum and minimum dimensional limits. The size of a component as well as the allowed variation (tolerance) is specified on the part drawing for the purpose of fabrication and, ultimately, documentation. Dimensional tolerances are set by design engineers after taking into consideration the component’s functional requirements, manufacturing process limitation, and cost with the first being of prime importance. For instance, setting the proper dimensions for a shaft and its mating hole is governed by strict rules to guarantee that when these dimensional deviations from nominal are properly set, the shaft/hole pair will function as envisioned by the designer. The resulting deviations of the shaft and the hole diameters vary depending on the desired ‘fit scheme’. Consequently, several fit designations exist in order to meet the variety of desired functions. Classical fit types are clearance, transition, and interference fits. One example of clearance fit is designated RC1 [12] and described as “close sliding fits and are intended for the accurate location of parts which must be assembled without perceptible play”. Locational fits include three sub-families of fits namely: locational-clearance fits (designated LC1 through LC9), locational-transitional fits (designated LT1 through LT6), and locational-interference fits (designated LN1 through LN3). The latter, for instance, represent those fits “used where accuracy of location is of prime importance and are intended for parts requiring rigidity and alignment with no special requirements for bore pressure. These fits are not intended for parts that must transmit frictional loads to one another”. Lastly, interference fits (FN1 through FN5). For example, FN1 describe light-drive fits “requiring light assembly pressures and produce more or less permanent assemblies”.

Although proper dimensioning and tolerancing is critical to the success or failure of the functioning of mechanical designs, tolerancing remains a time-consuming and human-intensive bottleneck in the design and manufacturing processes. As correctly pointed out in [13], “manual charting is tedious and error prone, hence, attempts have been made for automation”. Therefore, the idea of utilizing computer-based techniques (mainly computer-aided tolerancing, CAT) [14-15] for the purpose of automating tolerance generation [16-18] to enhance the process of specifying proper tolerances is an area of active research.

This work aims to build an expert design (dimensional tolerance) knowledge base targeted towards capturing expert tolerancing knowledge. We demonstrate this system and the knowledge acquisition process by specifying dimensional tolerances on shafts and mating holes in order to meet desired classes of fit as set by relevant engineering standards. We propose to efficiently build an effective KB, we dub the ***Design Assistant***, to incrementally capture expert designer’s prescription in dimensional tolerancing. The knowledge base would ensure that such a ‘smart’ software system would help mechanical designers become more effective in the time-consuming dimensioning and tolerancing process of their designs in the future with such implicit benefits as:

1) shortened product development process cycle when compared with a traditional dimension-and-tolerance-by-hand approach,
2) engineers can identify and avoid potential design conflicts and interferences early in the development process, reducing downstream errors, and engineering change orders (ECO's)
3) product lead times will be significantly reduced while improving quality and increasing the product’s performance-to-cost ratio.

2. The knowledge acquisition tool

An essential requirement of the workbench is the *ease* of acquisition and maintenance of the design knowledge, hence we use Ripple Down Rules as a starting point for the implementation of the knowledge base and the learning module of the system. We first give a brief description of Ripple Down Rules (RDR) which we use as foundational representation for our workbench and then describe our integrity constraints as used for an RDR knowledge base

Ripple Down Rules (RDR) is a knowledge acquisition method which proved very successful for developing large knowledge bases for classification tasks [20]. With RDR, knowledge maintenance is a simple process which can be done by the user without guidance of a knowledge engineer. there have been a number of extensions to the original idea, such as MCRDRs [19], or Nested RDR [21]. An RDR tree is a collection of simple rules organised in a tree structure. Every rule can have two branches to two other rules: A false and a true branch. An example is shown in Figure 1. When a rule applies a true branch is taken, otherwise a false branch is taken. The root node of an RDR tree contains the default rule whose condition is always satisfied. The root node is of the form "*If true then default conclusion*". The default rule has only a true-branch. In RDR, if a 'true-branch' leads to a terminal node *t* and the condition of *t* is not fulfilled the conclusion of the rule in the parent node of *t* is taken. If a 'false-branch' leads to a terminal node *t* and the condition of *t* is not fulfilled, then the conclusion of the last rule satisfied 'rippling down' to *t* is returned by the knowledge base. The knowledge base is guaranteed to return a conclusion as at least the default rule is satisfied 'rippling down' to *t.* Hence the inference is handled implicitly within the structure of the knowledge. When the expert disagrees with the conclusion returned by the knowledge base, the knowledge base is said to fail and requires modification.

An important strength of RDRs is the fact that they can be easily modified in order to become consistent with a new case without becoming inconsistent with previously classified cases. This is because every time a rule *r* is added to a parent rule *p*, *r* classifies the case which triggered its addition (the so-called corner stone case) correctly, and excludes all cases which are correctly classified by *p.* In their simple form, RDRs use simple attribute-value combinations as conditions for the rules [19-21]. When the expert enters a new rule *r*, he/she chooses the conditions of *r* from the so-called 'difference list' [22]. This list contains attributes satisfied by the case which triggered the addition of *r*, and it excludes all attributes satisfied by any of the cases covered by the parent of *r.*

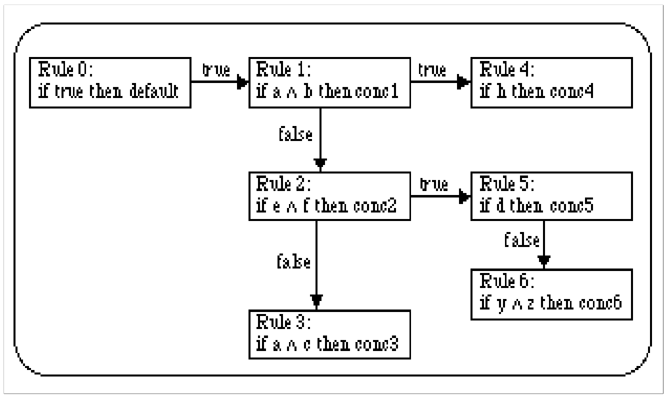


Figure 1. An example of ripple down rules.

In [23], Beydoun and Hoffmann presented an approach to incrementally capture search knowledge in general based on collecting expert's justifications for their decisions. In this paper, we apply this approach to the tolerancing problem in mechanical design proposing a system architecture that is outlined below. Moreover, we propose an automatic integrity check of the knowledge base based on monotonicity relationships among different object classes with respect to certain input dimensions (attributes of cases). While integrity checking has been done before, see e.g. [25], our proposal appears to be the first time monotonicity constraints in numerical domains across different classes has been proposed. More concretely, in the domain of tolerancing there are many classes of fit depending on the intended function. For certain subgroups of those classes one would normally expect monotonic relationship among the associated upper and lower tolerance limits. See Figure 4 for an example where there is an increasing tolerance limit for the RC*n* classes of fit. Such relationships can be exploited by an RDR system to ensure faster convergence of the knowledge base towards correct classification. I.e. while Figure 4 shows numerical values for standard operating conditions and standard material, these numbers vary depending on a number of copntextual factors, such as temperature range of the operating environment of the design object, pressures, materials used, cost of repair and maintenance, etc. For instance, if a mechanical part in a satellite wears out prematurely and needs replacement it is vastly more expensive than if that happens to a car or vacuum cleaner. However, the monotonic relationship between the RC*n* classes are still likely to hold, even if the numbers vary for another scenario. In order to accelerate the knowledge acquisition process in the RDR system, we propose to allow the explicit specification of likely monotonicity constraints as indicated above. Then once the expert is not happy with the classification or with the numerical tolerance limits provided by the RDR system, he/she would formulate an exception rule based on the non-standard operating conditions. The new rule may then violate a monotonicity constraint with existing rules for related classes. If so, the RDR system would then automatically retrieve the respective corner stone case of the firing rule from the related class and present it to the expert to validate or reject accordingly given the non-standard circumstances.

More formally, constraints for RDR knowledge base look as follows:

Let us consider consider a domain where each case be described by a set of attributes *Atts={A_1, ..., A_n}*. Each attribute *A_i* assumes a value from its respective domain *D_i={v_1,...,v_k}* or *DN_i*=**Z** (set of integer numbers) in case of a numerical attribute.

A rule *r* is a conjunction of conditions upon some or all of those attribute values in *Atts* together with a class label that is assigned to a case, if all conditions are satisfied. I.e. a rule is of the form

r= if *A_i=v_i* ∧ ... ∧ *A_j* ● *n_j* then *c*,

where ● stands for any of the relations in for any of the relations in {=,<,≤,≥,>,≠} and *A_j* has a numerical domain. Then a class *c_1* is upper bound by class *c_2* on attribute *A*, if for all cases *x_1* for which the RDR knowledge base produces class *c_1* there is no case *x_2* such that *x_2* differs from *x_1* only in attribute *A* and *x_2* having a greater value than *x_1*.

The inverse relationship can be defined analogously as a class *c_1* being lower bound by class *c_2* on attribute *A*.

3. System architecture of the *Design Assistant* software

The overall system architecture is shown in Figure 2 illustrating how the complete KA ***Design Assistant*** system works. In what follows, a brief discussion explaining the function of each subsystem is given.

Case verification: This module performs case validation (see section 5.3)

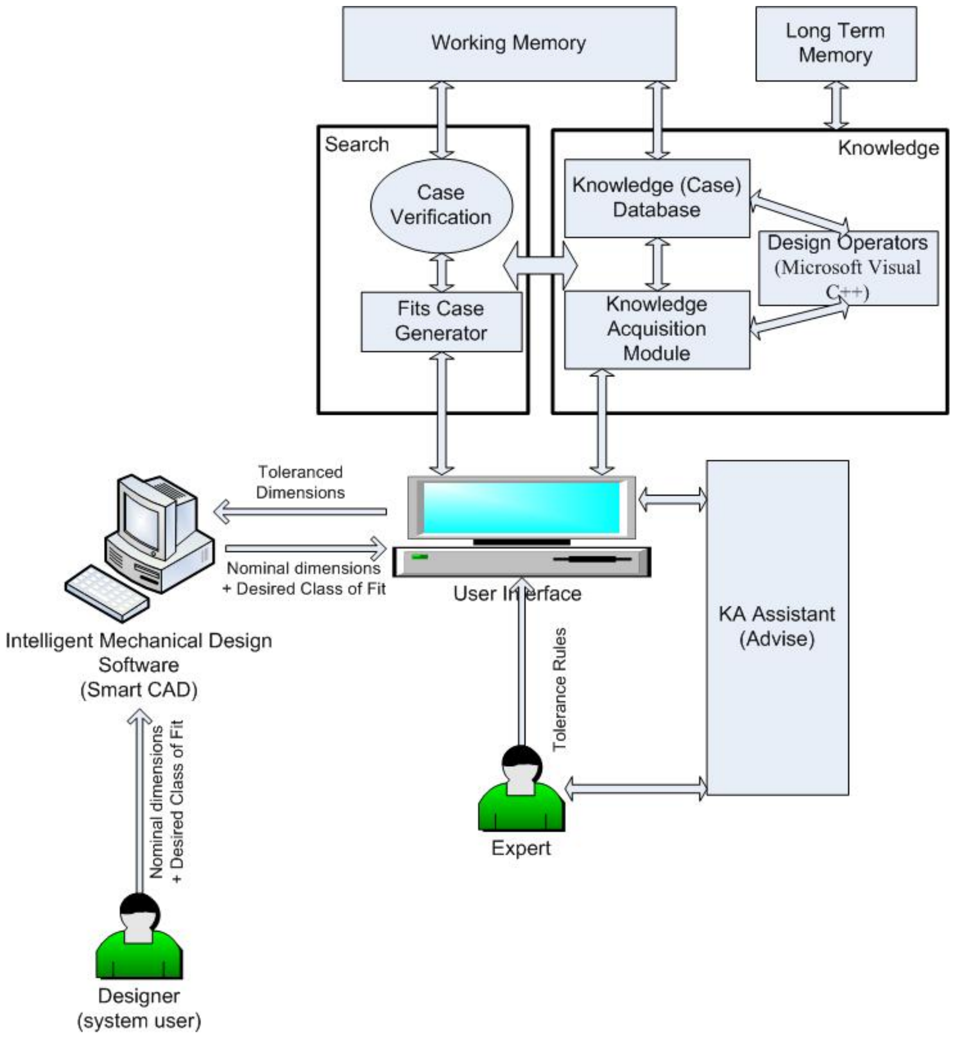
Design operators: The module contains a set of search operators forming an instance generator (Microsoft visual C++ module) and is where mathematical functions are declared. In our application, the conditions of the rules are given by these mathematical functions which are entered into the RDR interface prior to the start of the KA process.

Fits generator: generates cases of hole, shaft, and diameter that are read by the RDR program. These cases are such that each one corresponds to a class of fit. They are subsequently presented to the expert to build the knowledge base (as an RDR tree).

KA assistant: provides hints to the expert to which parts of the knowledge base may need to be modified while ensuring the consistency of the knowledge base with the case database. It relies on past interactions with the expert stored in the case data base to give these hints.

KA module: gets the expert input through the user interface. It maintains the knowledge base as well as the case data base.

Knowledge Base: stores what the expert expresses as his/her search control knowledge. This module contains the larger part of the domain knowledge. This knowledge base is built during the actual knowledge acquisition process.



Working Memory
Long Term Memory
Search
Knowledge
Case Verification
Knowledge (Case) Database
Design Operators (Microsoft Visual C++)
Fits Case Generator
Knowledge Acquisition Module
Toleranced Dimensions
Nominal dimensions + Desired Class of Fit
User Interface
KA Assistant (Advise)
Intelligent Mechanical Design Software (Smart CAD)
Tolerance Rules
Nominal dimensions + Desired Class of Fit
Expert
Designer (system user)

Figure 2. An illustration of the complete ***Design Assistant*** system.

Long term memory: contains all cases classified by the expert. It allows retrieval of these cases according to their classifications time stamp. Thus, this data base contains a complete history of the interactions with the expert. Although, not all of the interactions affect the knowledge base development, they are essential for the functionality of the knowledge acquisition assistant which is described below.

Mechanical Design Module: Contains the part's engineering description: shape, size (nominal dimensions), dimensional tolerances, and factors of fit, form and function. In the case of this example, it inputs nominal dimensions into the intelligent modules of the KA system and retrieves the upper and lower values on the nominal value, i.e., the desired tolerances.

User interface: This module reads the expert input and displays the system's answer to a search request. Further, it provides graphical representation of the knowledge base and graphical output of the automatic assistant to the expert.

Working Memory: stores the progress of the design, which is often used by the expert to explain his/her decisions. Solving a component placement problem, a designer chooses his/her next step based on a rough plan; this plan prevails in the progress towards finding a problem solution. Consequently, this progress is also used by the knowledge base to make decisions. The working memory also stores higher order features of steps of the evolving design. This reduces computational requirements as these features can get used again at a later stage of the design.

To fully implement the system shown in Figure 2 and to fulfil its goals, a number of knowledge modelling and acquisition tasks needs to be undertaken.at two layers of abstraction of the sought domain. At layer 1, the knowledge involved in making the atomic decisions is modelled and acquired. This involves choosing a fit according to the size of the shaft, the hole and the tolerances required. This decision also involves the orientation of the shaft with respect to the rest of the design. At layer 2, there are the tasks involved in modelling the knowledge of how various hole-and-shaft fits relate to each other and the overall geometry of the design. At these more complex decisions are, relations between current state of the design and previous design states need to be taken into account.

One RDR knowledge base will be required to model the knowledge at each of the two layers of abstractions. As well as leading to the completion of the design, each of these knowledge bases will also contribute to the generation of the KA cases to hasten their convergence (as described in Section 2). The completion of knowledge base describing the knowledge at the lower layer, is crucial to the generation of cases prior to the start of the knowledge acuiqistion process to complete the higher level knowledge base. To complete the system, we these stages are required:

- Stage 1: Model the primitives required to describe the knowledge at layer 1.
- Stage 2: Under take the KA process to develop the KB at layer 1.
- Stage 3: Undertake the case generator implementation and development.
- Stage 4: Undertake a KA process to develop KB at layer 2.
- Stage 5: Test and validate the system.

In this paper, we describe our substantial progress in completing stages 1 and 2 above. Section 4 will describe the knowledge involved in designing the primitives required to develop KB of Stage 2. Our system design allows iterations between stages 1 and 2, mathematical primitives can be added during the KA process. Section 5 will describe the KB developed for Stage 2.

4. The classical fit problem

A classical example common to the discipline of mechanical engineering is matching the relative fit of a shaft to a hole, as studied in [24]. Depending on the desired mating functionality between a shaft and a hole, different classes of interference/clearance fits are used. These classes cover a wide range of cases varying from loose clearance fit (sliding or running, RC) to tight interference fit (force, FN). In between, there are several variations of locational classes of fit namely: locational clearance (LC), locational transitional (LT), and locational interference (LN). These values for lower and upper tolerance limits are shown graphically in **Figures 3 and 4**. In order to calculate the upper and lower tolerance bounds on the diametrical dimension, the limit values L and U are multiplied by the nominal dimension (D, diameter of shaft and hole) raised to a power of 0.333 as follows:

$$\begin{aligned} &\textit{Lower Bound Tolerance} = L * D^{0.333} \\ &\textit{Upper Bound Tolerance} = U * D^{0.333} \end{aligned} \qquad (1)$$

The resulting tolerance values have units of mils (1/1000 of an inch). The toleranced dimension is, therefore, arrived at as a bounded value between the lower and the upper tolerance bounds:

$$Toleranced_{Dimension} = Nominal_{Dimension}{}^{+Upper_{bound_{tolerance}}}_{-Lower_{bound_{tolerance}}} \qquad (2)$$

Given a desired class of fit (e.g. LC1, LC2), the scheme gives the upper and lower tolerance bounds for the nominal diameter of the shaft/hole. This is conventionally described as the forward scheme. The backward scheme, on the other hand, is described as follows: given actual diametrical dimensions for the shaft and hole, it is desired to correctly identify the resulting class of fit. This latter scheme is demonstrated in the section below where we describe our application in *tolerancing.* We highlight the domain specific features. We then discuss the functionality of the system and show actual results.

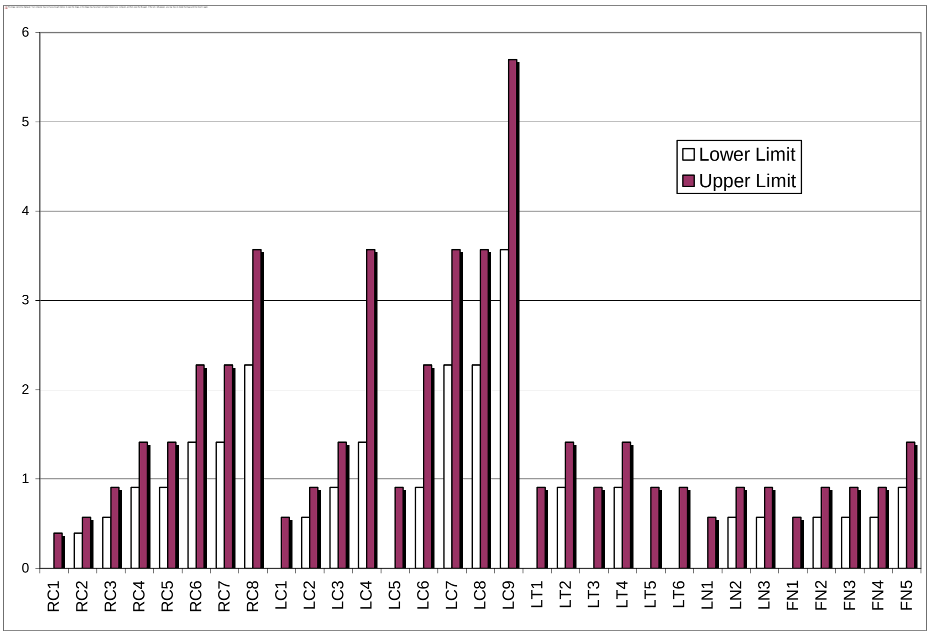


Figure 3. Hole: lower (L) and upper limit (U) values for all 31 fit cases.

5. Application of *Design Assistant* in Tolerancing

We now describe our work towards acquiring and applying knowledge as applies to dimensional tolerancing. We consider the classical example of the shaft-in-a-hole mechanical fit problem as introduced in Section 4. In the following scheme, the user feeds in one desired diametrical dimension for the shaft and another dimension for the mating hole. To this, the software returns a match to one of 31 possible fit criteria.

A function is needed for checking if the actual value of the hole or shaft lies within the limits of a certain class of fit. This first function is based on **Equation (1)** where L, U, and D are as defined above and R stands for the actual value of the case for the hole or the shaft:

$$L * D^{1/3} < (\mathrm{R-D}) * 1000 < \mathrm{U} * D^{1/3} \qquad \textbf{(3)}$$

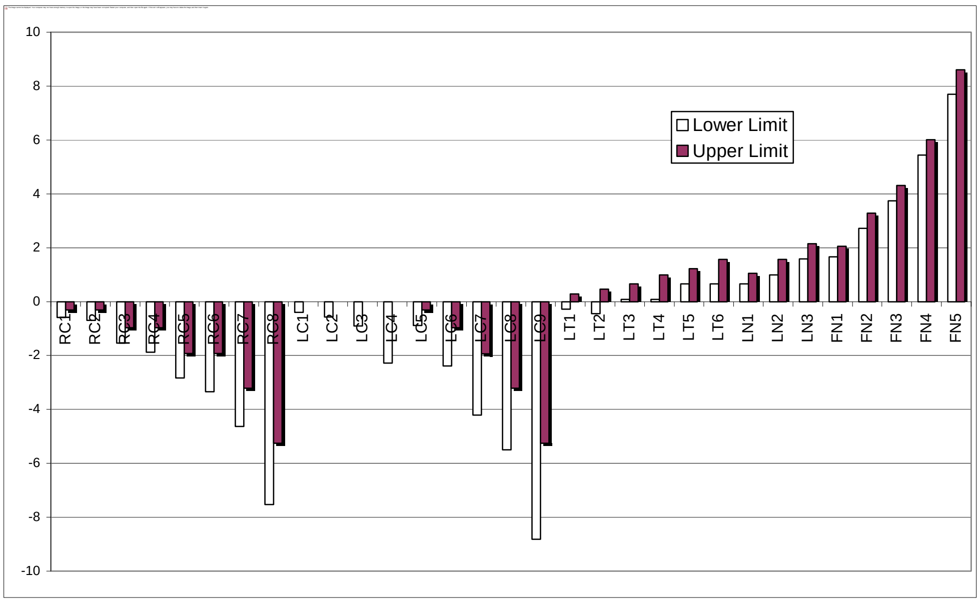
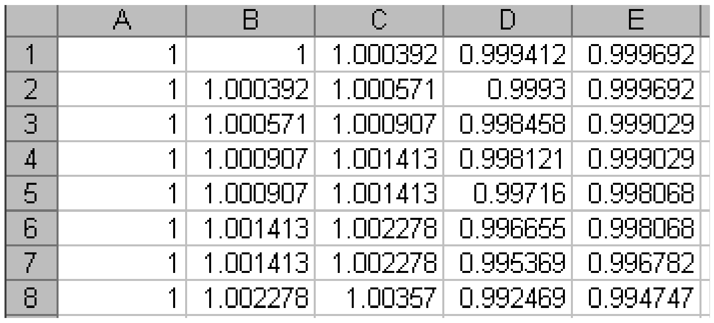


Figure 4. Shaft: lower (L) and upper limit (U) values for all 31 fit cases.

This checks whether or not the case may be classified as *Locational*.

5.1. The Fits Generator module

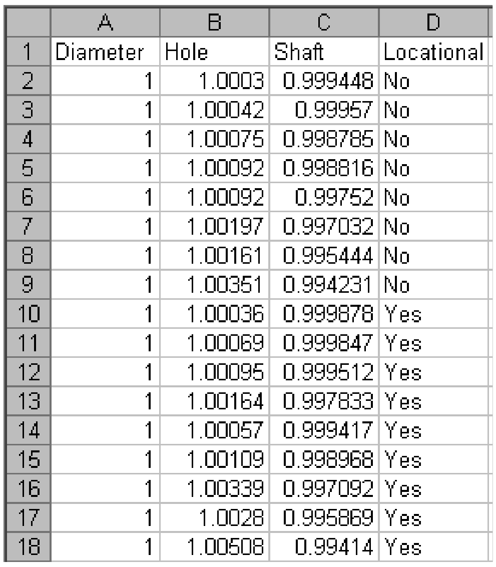
The ***Fits Generator*** module was developed to generate fit cases that can be loaded by the RDR program, one for each class of fit. With the exception of case LN1 (which was found to fall completely within the limits of the class designated LT5), the tree was built to contain all cases. The tree had a separate rule for each class. (For the exceptional case with common limits, the conclusion of either of the two classes LN1 or LT5 will be given). The module reads from its needed values from two files in order to generate the cases. The first file has limits values of the classes for which cases are to be generated. It also contains the nominal diameter of the case.



	A	B	C	D	E
1	1	1	1.000392	0.999412	0.999692
2	1	1.000392	1.000571	0.9993	0.999692
3	1	1.000571	1.000907	0.998458	0.999029
4	1	1.000907	1.001413	0.998121	0.999029
5	1	1.000907	1.001413	0.99716	0.998068
6	1	1.001413	1.002278	0.996655	0.998068
7	1	1.001413	1.002278	0.995369	0.996782
8	1	1.002278	1.00357	0.992469	0.994747

Figure 5: Example of limits file showing only the first 8 fit cases RC1-RC8. Hole's (columns B and C) and shaft's (columns D and E) upper and lower limits for a one inch (1") nominal dimension (column A).

Figure 5 is an example of cases generated where the nominal diameter = 1 inch (1st column). The 2nd and 3rd columns are the lower and upper limits on the hole, respectively, while the 4th and 5th columns contain those of the shaft. For each class, a case is generated with random hole and shaft values that fit within the given limits. The second file needed for the fits generator has the attributes of the generated cases.



	A	B	C	D
1	Diameter	Hole	Shaft	Locational
2	1	1.0003	0.999448	No
3	1	1.00042	0.99957	No
4	1	1.00075	0.998785	No
5	1	1.00092	0.998816	No
6	1	1.00092	0.99752	No
7	1	1.00197	0.997032	No
8	1	1.00161	0.995444	No
9	1	1.00351	0.994231	No
10	1	1.00036	0.999878	Yes
11	1	1.00069	0.999847	Yes
12	1	1.00095	0.999512	Yes
13	1	1.00164	0.997833	Yes
14	1	1.00057	0.999417	Yes
15	1	1.00109	0.998968	Yes
16	1	1.00339	0.997092	Yes
17	1	1.0028	0.995869	Yes
18	1	1.00508	0.99414	Yes

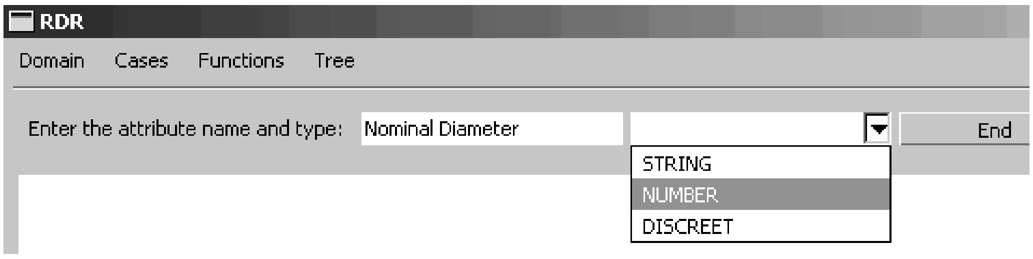
Figure 6: Example of the < fits_cases.txt> file (showing only the first 17 fit cases for illustration purposes).

Fits Generator output contains a random case for each class of fit introduced by the file of the limits. The output file name is the same name of the domain: *Fits*. **Figure 6** is an example of such a file where the first row contains the case attribute names and the rest of the rows contain the cases.

5.2. Adjusting DA to the Domain

Having acquired domain instances, a domain is engineered in DA. This includes specifying the names and types of the attributes and defining any higher order functions required by the designer to express his design knowledge. In this section, we illustrate these steps.

The attribute *Nominal Diameter* represents the nominal diameter of the hole-shaft system. This is defined as a number from the list as shown in **Figure 7**. Other attributes are *Hole* (actual hole diameter) and *Shaft* (actual shaft diameter) are similarly defined. Cases generated from the ***Design Assistant***$_{\textit{Fits\_Generato}}$ module are then loaded.

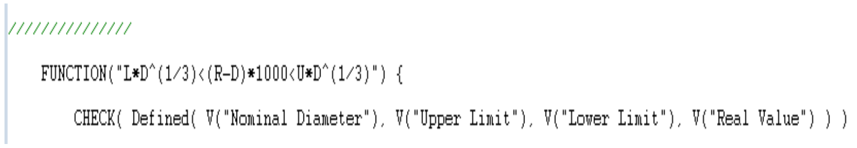


Enter the attribute name and type: Nominal Diameter

Figure 7. Attribute definition facility in the ***Design Assistant***$_{\textit{GUI}}$

Defining functions such as **(3) and (4)** involves providing a function header showing its parameters and their specifications. Function parameters include *Lower Limit* of a class of fit for the Hole or the Shaft, *Upper Limit* of a class of fit for the Hole or the Shaft, *Real Value* of the hole or the shaft. For function **(4)**, both attributes A and B are 'String' type. Declaring these functions requires this sequence of operations:

1. In the workspace from the FunctionBox folder provide the function header as input (as shown in **Figure 8)**.



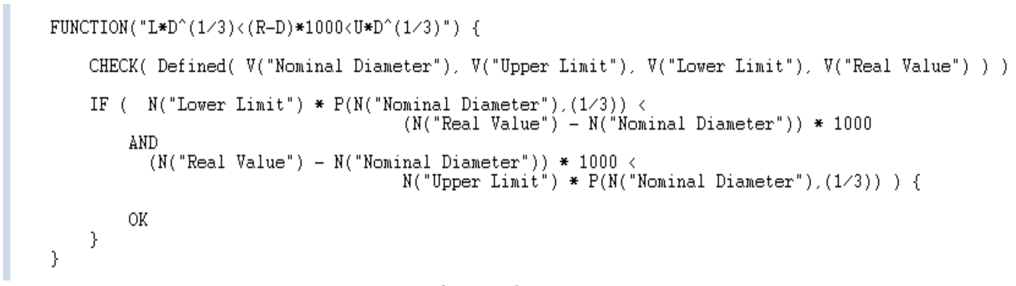
```
///////////////

   FUNCTION("L*D^(1/3)<(R-D)*1000<U*D^(1/3)") {

      CHECK( Defined( V("Nominal Diameter"), V("Upper Limit"), V("Lower Limit"), V("Real Value") ) )
```

Figure 8. Fit function declaration and check.

2. Verify that all attributes are defined by using a function check mechanism within DA: *CHECK (Defined (V("Nominal Diameter"), V("Upper Limit"), V("Lower Limit"), V("Real Value")))* Where '**Defined'** is a function that returns true if all the attributes passed to it are pre-defined.
3. To declare the condition in the function, the attributes are to be used either as text, or as numbers. N("<attribute name>") will return the value of the attribute named <attribute name> in double precision number format. T("<attribute name>") will return the value of the attribute named <attribute name> as text. P(<first number>, <second number>) will return the <first number> raised to the power <second number>. <first number> and <second number> should be numbers and not text. There are two comparisons to be made. Those comparisons are joined with the logical AND operator. '**OK'** means the function should return true. If the condition is not satisfied, false is returned. **Figure 9.a and 9.b** shows the resulting window declaring the Functions in **(3 and 4)**.
4. After declaring the functions in C++, recompile, and link.



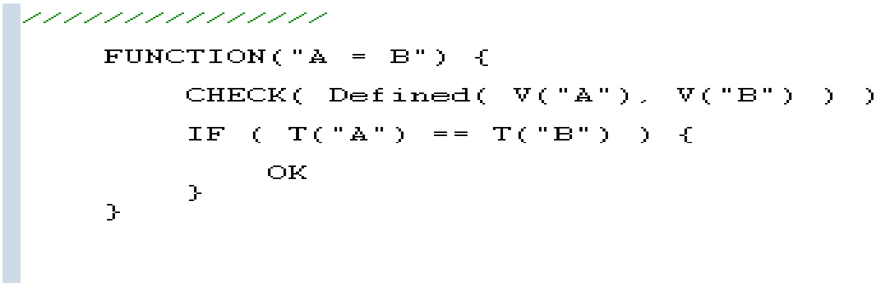
```
FUNCTION("L*D^(1/3)<(R-D)*1000<U*D^(1/3)") {

    CHECK( Defined( V("Nominal Diameter"), V("Upper Limit"), V("Lower Limit"), V("Real Value") ) )

    IF (  N("Lower Limit") * P(N("Nominal Diameter"),(1/3)) <
                                   (N("Real Value") - N("Nominal Diameter")) * 1000
        AND
          (N("Real Value") - N("Nominal Diameter")) * 1000 <
                                   N("Upper Limit") * P(N("Nominal Diameter"),(1/3)) ) {

        OK
    }
}
```

(a) Defining function in (3)



```
//////////////////
     FUNCTION("A = B") {
          CHECK( Defined( V("A"), V("B") ) )
          IF ( T("A") == T("B") ) {
               OK
          }
     }
```

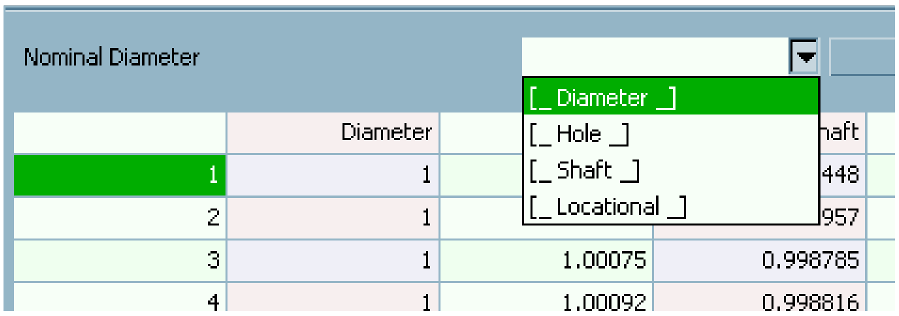
Figure 9. (b) Defining function in (4)

5.3 Case Validation

Next, the ***Case verification*** module performs case validation sequentially. Double click on the case number of the case to validate. The first case belongs to the class of fit designated RC1 which will be used thereafter for the purpose of demonstrating the software. In the GUI, the screen contains the entries exactly as in the fits_cases.txt file shown in **Figure 6**. Clicking on case RC1, a window pops-up giving the conclusion of the tree, and asking if the expert accepts the conclusion. A '**DEFAULT'** conclusion corresponds to the first rule of the tree that is always true. This default conclusion only appears if RDR concludes that no other plausible conclusion exists. Selecting '**Yes'** will keep the conclusion resulting in no changes to the tree. Selecting '**No'** requires justification of the refusal of the conclusion.

5.3.1 Adding Rules

A function would have to be selected in order to add a new rule. **Figure 11** shows the functions as they appear in the GUI. The first function is needed to add a condition for the size of the hole. The value of an attribute of a function can be a value entered as text or number. The value will be stored in the rule, and used whenever the rule is used. The value of an attribute of a function can also be an attribute of a case. The value of the cases will not be stored in the rule, but the name of the attribute will be stored. The function will use the value of the specified attribute of the case being evaluated. *Nominal diameter* is an attribute of the function. The value of this attribute should be the value of the diameter of the case. The list of the combo box contains the attributes of the case. Select **'_ Diameter _'** from the list as shown in **Figure 12**.

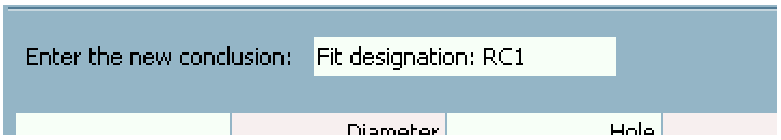


Nominal Diameter

	Diameter		haft
1	1		448
2	1		957
3	1	1.00075	0.998785
4	1	1.00092	0.998816

Figure 12. Case attributes as they appear in the ***User interface***

The next attribute of the function is ‘Lower Limit’. The lower and upper limit values of the hole for the class RC1 are set to 0 and 0.392, respectively. The value of the next attribute, the real value of the hole of the case ‘Real Value’, should be retrieved from the case. Select the second entry **‘_ Hole _’** from the list as shown above in **Figure 12**. The first condition for the hole is added to the rule. The rule is not yet complete. The condition for the shaft needs to be added (**Figure 13)**. Press the “Add Condition” button to add a new condition. Select the first function and similarly add the condition for the shaft. Select ‘**_ Shaft _**’ instead of the ‘**_ Hole _**’. The lower limit of the shaft for RC1 is -0.588. The upper limit is -0.308. After adding the condition, select “**Add Condition**” to specify whether or not the class of fit is ‘Locational’.



Enter the new conclusion: Fit designation: RC1

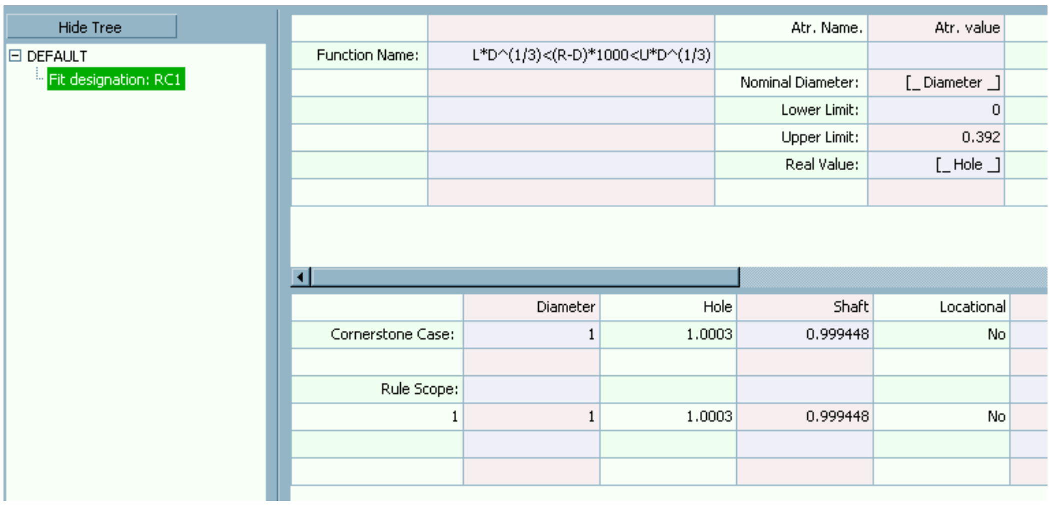
Figure 13. Adding conditions to functions in the ***User interface***

The second function is then selected and since RC1 is not Locational, the condition should be that the attribute ‘Locational’ has the value ‘No’. Select for the attribute ‘A’ of the function ‘A = B’ attribute ‘Locational’ of the case. For the second attribute of the function, set the text value to ‘No’. To finish creating the rule, enter the condition:

“*Fit designation: RC1*”. The first rule has just been created. Having validated this case, any situation that belongs to the RC1 class of fit will return the conclusion “***Fit designation: RC1***”**.** The rest of the rules are added in the same fashion.

5.4 Using the Tree

Having defined all the rules, the fully populated tree is saved and will be available for later loading and viewing. Upon viewing, the tree will appear as shown in **Figure 14** complete with the condition(s) of each rule, the cornerstone case, and the scope of the rule.



		Atr. Name.	Atr. value
Function Name:	L*D^(1/3)<(R-D)*1000<U*D^(1/3)		
		Nominal Diameter:	[_ Diameter _]
		Lower Limit:	0
		Upper Limit:	0.392
		Real Value:	[_ Hole _]

	Diameter	Hole	Shaft	Locational
Cornerstone Case:	1	1.0003	0.999448	No
Rule Scope:				
1	1	1.0003	0.999448	No

Figure 14. A populated tree as seen on the ***User interface*** complete with the condition(s) of each rule, the cornerstone case, and the scope of the rule.

6. Discussion and Future Work

In this paper we proposed a framework for the acquisition of expertise on proper tolerancing of mechanical parts within a design. We developed a system ready to be integrated into an existing CAD environments. We demonstrated the feasibility of acquiring expertise in this domain quite effectively by building a decent-sized knowledge base (using a Mechanical Engineer who regularly consults industry in this field). While the appropriate tolerances do not only depend on the intended function of the involved parts but also on operating conditions, such as temperature ranges, etc. as well as on materials being used a much more sophisticated knowledge base would need to be built for unusual materials or operating conditions, such as for equipment to be used in space, in deep water, etc. However, the apparent ease of knowledge acquisition for standard operating conditions suggests that the same RDR approach can also be used to acquire more refined knowledge for special conditions. We propose to aid the knowledge acquisition process in cases like those by using monotonicity constraints to assist the faster acquisition of knowledge for related classes. Further types of constraints can be developed that could be applied to our mechanical design problem and other domains. One possible further constraint could specify minimum quantitative differences for monotonically related quantities across related classes. Another possible future direction for our intelligent Design Assistant is to also assist in optimizing the tolerances to be specified with respect to the entire system design. It is important to note that generally, the smaller the specified tolerances, the more expensive it becomes to actually manufacture parts to specifications. In order to minimize costs, the largest tolerance margins (with the associated class of fit) would be desirable. However, in order to ascertain proper functioning and sufficient durability of the overall system the optimal trade-offs between the chosen classes of fit (and their allowed tolerances) for different parts need to be found. While prima facie an engineer would still need to decide what alternative classes of fit are acceptable for a group of parts an constraint satisfaction problem solver could be employed for finding the optimal choice. However, in specifying alternative classes of fit for different parts or sets of parts intelligent assistance is highly desirable due to the time-consuming process of manually considering alternative classes of fit. In order to assist here an RDR based system appears suitable once more to quickly provide alternative tolerancing schemes to the human designer to confirm or reject.

References

1. Chapman, M.P., 1999, *Design engineering- a need to rethink the solution using knowledge based engineering* , Knowledge-Based Systems, 12, 257-267.

2. Tomiyama, T. and Yoshikawa H., 1987, *Design Theory for CAD*. Amsterdam: North-Holland.

3. Tomiyama, T. and Yoshikawa H., 1987, *Expert Systems for Computer-aided Design*, Amsterdam: North-Holland.

4. Crosnier, A. and Rharmaoui, A., 1994, *Knowledge-based CAD/CAM system for modelling the design process*, American Society of Mechanical Engineers, Petroleum Division (Publication) PD, "Methodologies, Techniques, and Tools for Design Development", 64(5), 81-87.

5. Grabowski, H., Kunze, H., Lossack, R., and Michelis, A., 2002, *Interpretation of low-level CAD data for knowledge extraction in early design stages*, Graphics Recognition. Algorithms and Applications. 4th International Workshop, GREC 2001. Selected Papers (Lecture Notes in Computer Science Vol.2390), p 13-24.

6. Kesheng Wang, Meng Tang, Yi Wang, Estensen, L., Sollie, P.A., and Pourjavad, M., 2002, *Knowledge-based CAD/CAPP/CAM integration system for manufacturing*, Digital Enterprise Challenges. Life-Cycle Approach to Management and Production. IFIP TC5/WG5.2 & WG5.3 Eleventh International PROLAMAT Conference on Digital Enterprise - New Challenges, p 406-15.

7. Chandra, C. and Kumar, S. 2003, *Enhancing manufacturing operations effectiveness through knowledge based design*, Integrated Manufacturing Systems, 14 (3) 278-292.

8. American National Standards Institute (ANSI), 1430 Broadway, New York, N.Y., 10018

9. American National Standard (ANSI-Y14.5.1M), *Mathematical Definition of Dimensioning & Tolerancing*. The American Society of Mechanical Engineers, 1994.

10. American National standard ANSI Y14.5M-1994; *Dimensioning & Tolerancing*. The American Society of Mechanical Engineers, ASME Press, 1994.

11. Meadows, J.D., *Geometric Dimensioning and Tolerancing- Applications and Techniques for use in Design, Manufacturing, and Inspection*,. 1995: Marcel Dekker, Inc.]

12. *Preferred limits and fits for cylindrical parts*, ANSI B4.1-1967 (R1979), ANSI Publications, 1430 Broadway, New York, N.Y., 10018
13. Shen, Z., Ameta, G., Shah, J.J., and Davidson, J.K., 2005, *A Comparative Study Of Tolerance Analysis Methods,* Journal of Computing and Information Science in Engineering, 5(3)247-256]
14. Chiesi F. and Governi, L., 2003, *Tolerance Analysis with eM-TolMate*, Journal of Computing and Information Science in Engineering 3 (1) 100-105.
15. Shen Z., *Tolerance Analysis with EDS/VisVSA*, Journal of Computing and Information Science in Engineering, 2003, 3(1)95-99.
16. Wang, N. and Ozsoy, T.M., 1993, *Automatic generation of tolerance chains from mating relations represented in assembly models*, Journal of Mechanical Design, Transactions Of the ASME, 115(4), 757-761.
17. King, D.A. and de Sam Lazaro, A. 1994, *Process and tolerance considerations in the automated design of fixtures*, Journal of Mechanical Design, Transactions Of the ASME, 116(2), 480-486.
18. Hu, J. and Peng, Y., 2007, *Tolerance modelling and robust design for concurrent engineering*, Proceedings of the I MECH E Part C Journal of Mechanical Engineering Science, 221 (4), 455-465.
19. Kang, B. Validating Knowledge Acquisition: Multiple Classification Ripple Down Rules. PhD Thesis, University of New South Wales, Sydney, Australia, 1996.
20. Beydoun G., Hoffmann, A., Fernández Breis, J.T., Martinez Béjar, R., Valencia-Garcia, R., Aurum, A., 2005, *Cooperative Modeling Evaluated*, International Journal of Cooperative Information Systems, World Scientific, 14 (1), 45-71.
21. 27. Beydoun, G., Hoffmann, A., Simultaneous modeling and knowledge acquisition using NRDR. PRICAI'98: Topics in Artificial Intelligence. Lecture Notes in Computer Science. Volume 1531. 1998. Springer. 83-95.
22. Compton, P. and Jansen. R., 1990, *A philosophical basis for knowledge acquisition.* Knowledge Acquisition, **2**, 241-257.
23. Beydoun, G. and Hoffmann, A., 2000, *Incremental Acquisition of Search Knowledge.* International Journal of Human Computer Studies, **52**(3), 493-530.
24. *Standard Handbook of Machine Design*, 1986, ed. J.E. Shigley and C.R. Mischke. McGraw-Hill Book Company.
25. Evaluation of the FastFIX prototypes 5Cs CARD system. Proceedings of the 8th PRICAI 2006, Guilin, China, pp.106-117.