2010

A dimensional tolerancing knowledge management system using Nested Ripple Down Rules (NRDR)

Ramsey Hamade
The American University of Beirut

V. C. Moulianitis
University of Patras

D. D'Addonna
University of Naples Federico II

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

A dimensional tolerancing knowledge management system using Nested Ripple Down Rules (NRDR)

Abstract

This paper proposes to use a knowledge acquisition (KA) approach based on Nested Ripple Down Rules (NRDR) to assist in mechanical design focusing on dimensional tolerancing. A knowledge approach to incrementally model expert design processes is implemented. The knowledge is acquired in the context of its use, which substantially supports the KA process. The knowledge is captured 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 in order to demonstrate the presented approach. The developed dimensional tolerancing knowledge management system would help mechanical designers become more effective in the time-consuming tolerancing process of their designs in the future.

Disciplines

Physical Sciences and Mathematics

Publication Details

Hamade, R., Moulianitis, V., D'Addonna, D. & Beydoun, G. (2010). A dimensional tolerancing knowledge management system using Nested Ripple Down Rules (NRDR). Engineering Applications of Artificial Intelligence, 23 (7), 1140-1148.

Engineering Applications of Artificial Intelligence 23 (2010) 1140–1148

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

A dimensional tolerancing knowledge management system using Nested Ripple Down Rules (NRDR)

R.F. Hamade [a,*], V.C. Moulianitis [b], D. D'Addonna [c], G. Beydoun [d]

[a] Department of Mechanical Engineering, The American University of Beirut (AUB), P.O. Box 11-0236, Riad El-Solh, Beirut 1107 2020, Lebanon
[b] Mechanical Engineering and Aeronautics Department, University of Patras, Patras 26500, Greece
[c] Department of Materials & Production Engineering, University of Naples Federico II Piazzale Tecchio 80, I-80125 Naples, Italy
[d] School of Information Systems and Technology (SISAT), Faculty of Informatics, University of Wollongong, Wollongong NSW 2522, Australia

ARTICLE INFO

Article history:
Received 6 May 2009
Received in revised form
30 September 2009
Accepted 30 October 2009
Available online 4 January 2010

Keywords:
Nested Ripple Down Rules, NRDR
Knowledge acquisition
Design
Fits and tolerances
Knowledge-based systems

ABSTRACT

This paper proposes to use a knowledge acquisition (KA) approach based on Nested Ripple Down Rules (NRDR) to assist in mechanical design focusing on dimensional tolerancing. A knowledge approach to incrementally model expert design processes is implemented. The knowledge is acquired in the context of its use, which substantially supports the KA process. The knowledge is captured 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 in order to demonstrate the presented approach. The developed dimensional tolerancing knowledge management system would help mechanical designers become more effective in the time-consuming tolerancing process of their designs in the future.

1. Introduction

Proper dimensional tolerancing is critical to the success or failure of the functioning of mechanical designs. Mechanical systems are represented by parts 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.

While many of the actions during the detailed design process are automated, dimensional tolerancing, involve intense decision-making, and therefore, remains a time-consuming and human-intensive activity in the design and manufacturing processes. Typically, upon building the preliminary design, a designer sequentially spends a significant amount of time re-dimensioning features and 'annotating' certain tolerancing information. As correctly pointed out in (Shen et al., 2005), "manual charting is tedious and error prone, hence, attempts have been made for automation". Radack and Sterling (1994) lamented that "the designer is left with the responsibility of ensuring that the tolerances are complete and consistent. The systems do not ensure that tolerances are reasonable or meaningful". This is certainly true in the case of traditional 2-D drawing-based manufacturing as well as the up-and-coming route of releasing the mechanical database in electronic form (Rezayat, 2000). For such a purpose, standards developed under ISO (International Organization for Standardization) such as ISO 10303 Product Data Representation and Exchange (STEP) Part 42 "Geometric and Topological Representation" (ISO, 1999) and Part 47 "Shape variation tolerances" (ISO, 1997) emphasize the proper definition and representation of shapes, dimensions, and tolerances. Parameters necessary for the proper implementation of automatic tolerancing schemes and geometric data exchange are also controlled. Such definitions and practices include: tolerances as constraints on the shape characteristics of a product, representing geometric and plus–minus tolerances, representation of tolerance values, synthesis and analysis of tolerances, dimensioning and tolerancing practices, and presentation of tolerances on engineering drawings.

To mimic the designer-expert, some best-of-class mechanical CAD/CAE/CAM platforms have received their fair share in making them generally more intelligent through the use of artificial intelligence (AI) techniques (Roy, 1994; Finger et al., 2000). Given such applications, the use of AI as applied to design and engineering disciplines has of late become indispensable. One significant product of this synergy is what has become to be generically known as knowledge-based systems (KBS). The

* Corresponding author. Tel.: +961 1 350 000x3481.
E-mail address: rhamade@aub.edu.lb (R.F. Hamade).

0952-1976/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2009.10.007

terminology broadly refers to *intelligent* software programs that apply expert knowledge to the solution of problems. The theoretical and methodological foundation of AI as applies to aspects of mechanical design in general is of great interest (Chapman, 1999). Specific to tolerancing, the idea of utilizing computer-based techniques (mainly computer-aided tolerancing, CAT) (Chiesi and Governi, 2003; Shen, 2003) for the purpose of automating tolerance generation and modelling to enhance the process of specifying proper tolerances is an area of active research (Wang and Ozsoy, 1993; King and de Sam Lazaro, 1994; Desrochers, 2003; Wu and Rao, 2004; Hu and Peng, 2007). The idea being is to develop rule-based expert systems to help the designer create *complete* and functional designs with appropriate dimensions and tolerances *in* the design stage. The completeness of the design database being a manufacturing requirement, the 3-D database description of the part should contain meaningful geometric attributes: dimensions, tolerances, and factors of form (flatness, squareness, etc.) as alluded to in the above-mentioned ISO standards.

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.) (ANSI, 1994a; ANSI, 1994b; Meadows, 1995). For proper dimensional tolerancing, it is this domain of tolerancing expertise that must be captured and represented as a design knowledge base (DKB). Ripple Down Rules (RDR) are used for such a foundational representation is this work. While most design rationale frameworks such as IBIS, COQ or DRL are focused on the initial development of the knowledge base of a system (see, for example the review paper (Hu et al., 2000)), RDR shifts the development emphasis to maintenance by blurring the distinction between initial development and maintenance (Kang et al., 1998). An essential requirement for the development of the KBS is the ease of acquisition and maintenance of the knowledge. RDR (Gaines, 1991) is a knowledge acquisition method which proved very successful for developing large knowledge bases for classification tasks (Beydoun et al., 2005). With RDR, knowledge maintenance is a simple process which can be done by the user without guidance of a knowledge engineer (Beydoun and Hoffmann, 2001). Furthermore, RDR is optimized for maintenance of propositional rule-bases and ensures very high behavior coverage as the systems evolves (Menzies and Compton, 1995) so that there is no need to import the rule-base in a truth maintenance system (TMS).

In this paper, an approach for acquiring knowledge using RDR for dimensional tolerance is presented. A knowledge base targeted towards capturing expert tolerancing knowledge is built. The system is demonstrated by specifying dimensional tolerances on shafts and mating holes in order to meet desired classes of fit as set by relevant engineering standards. This paper begins with the methodology used to construct the KBS presenting the method of acquiring the design knowledge and the requirements of the knowledge acquisition environments. The knowledge acquisition tool and the software tool are followed. Then, the problem and an illustrative example are presented. Finally the paper closes with concluding remarks.

2. Methodology

2.1. Acquiring design knowledge

In order to construct the knowledge base the design knowledge must be acquired. In this work a spiral process of knowledge acquisition is envisaged, similar to (Linster, 1993) of coming stepwise closer and closer to an operationalization of the knowledge in question. The present approach follows the work on knowledge acquisition which allows knowledge acquisition and maintenance without a knowledge engineer (Compton et al., 1994; Kang et al., 1998).

Experts are usually able to explain their reasoning process on a particular problem instance in rather general terms that cover at least the given concrete next step in their design process. However, their explanation may be quite inaccurate in the sense that for other design problems their explanation would not deliver the design step they would actually take. Either their explanation would not cover the step they would take or their explanation would suggest design steps they would not actually consider. Thus, an approach similar to (Hoffmann and Thakar, 1991) it is pursued, which allows to incrementally acquiring complex concept definitions without demanding an operational definition from the expert. Rather, the expert is merely required to judge whether the concept applies to particular instances. This is a much more natural task for an expert than to articulate general rules on how to judge on any particular instance.

In (Beydoun and Hoffmann, 2000), 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, this approach is applied to the tolerancing problem in mechanical design following the method outlined below. Consider this simple expert design process in mechanical engineering for illustration purposes. An expert designer, thinking aloud, may report the following:

> I have this new injection molding machine to be designed. In it, there these four (4) shafts that guide the lateral motion of the movable platen which in turn supports the mold's core half. In this platen, I have to locate each one of these 2-inch nominal diameter shafts. Now, each one of these holes should be small enough to allow the shafts to be properly located yet be large enough so that each shaft moves freely without much friction. I guess I better look up the class fit tables in the design handbook so that I can (1) pick a suitable fit class that meets these constraints and then to (2) calculate the upper and lower limits on the nominal 2-inch dimension for the shaft and its mating hole on the platen. From there, I will dimension the shaft and hole on the drawing so that they can be fabricated to the proper size …
>
> But oh, on a second thought, if I can only use the same platen from this older design. I would have to check the actual measurements of the holes diameter on this old platen as well as the actual shaft diameter that we have in stock. Then, for these actual dimensions, I will look up the dimensional tolerance values in the fit tables to determine the resulting fit. If the fit class is good enough tolerance-wise, I think the design will work and lots of money will be saved by not having to fabricate a new platen … hmm, this just might work out …

Such an expert design process involves more complex reasoning than just the association of design sequences which were useful in other design instances. For example it involves some causal reasoning on a rather abstract level. However, it seems difficult to devise a general inference mechanism, which could accommodate such expert reasoning. This is particularly the case, since much of such reasoning will not be at a conscious level to the expert. More applicable to tolerancing, the example illustrates that such a dimensional tolerancing knowledge management system can be utilized either to (1) given a class fit, specify tolerances for nominal dimensions or (2) given actual dimensions, back out the corresponding fit. In Section 5 below, these two

schemes are referred to as the 'forward scheme' and the 'backward scheme', respectively.

2.2. Requirements for a KA environment

A suitable learning environment for the design software system has to support the following steps for the building of a KBS:

1. The criteria being used to select design steps worthwhile must be very flexible, i.e. the following options should be available:
   - The expert can freely characterise design problems as well as design actions applied to these problems.
   - The revision or modification or amendment of initial characterization of problems and actions must be possible.
2. As indicated in the example of a mechanical engineer design process, how design proceeds may depend on the findings of intermediate design steps encountered earlier in the design process. To accommodate this sort of reasoning, the system should log the intermediate design steps. This reduces computational requirements as many characteristics persist over sequences of design steps. Furthermore, the expert definable selection criteria for design steps must allow conditions which involve such findings of earlier encountered design steps.
3. A design task may have components and the expert's comments will often describe relations between these components. Thus, a representation that is more powerful than propositional logic to represent expert comments about intermediate design steps is required. This representation should be adaptable easily for various design tasks. To address these issues the RDR (Beydoun and Hoffmann, 2000) is extended with the ability to accommodate domain specific primitives during the knowledge acquisition process.
4. The workbench must allow and accommodate the expert's suggestions. That is the expert will not only comment on the solutions found by the system, she/he must suggest and justify a solution to the system when it fails to find one. When a suggestion is made, new design actions may get introduced to the system. Hence accepting suggestions ensures that the design actions used by the system are effective and speeds the knowledge acquisition process.

The fulfilment of the above requirements within the present framework is discussed in the following sections.

3. The knowledge acquisition tool

Ripple Down Rules (RDR) are used as foundational representation for the present workbench. An essential requirement of the workbench is the *ease* of acquisition and maintenance of the search knowledge. For this purpose, RDR is used as a starting point for the implementation of the KBS and the learning module.

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). Examples are shown in Fig. 1 (Beydoun and Hoffmann, 1997) where every block represents a simple RDR. 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 cornerstone 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 (Gaines, 1991; Beydoun and Hoffmann, 2001; Beydoun et al., 2005). When the expert enters a new rule *r*, she/he chooses the conditions of *r* from the so-called 'difference list' (Compton and Jansen, 1990).

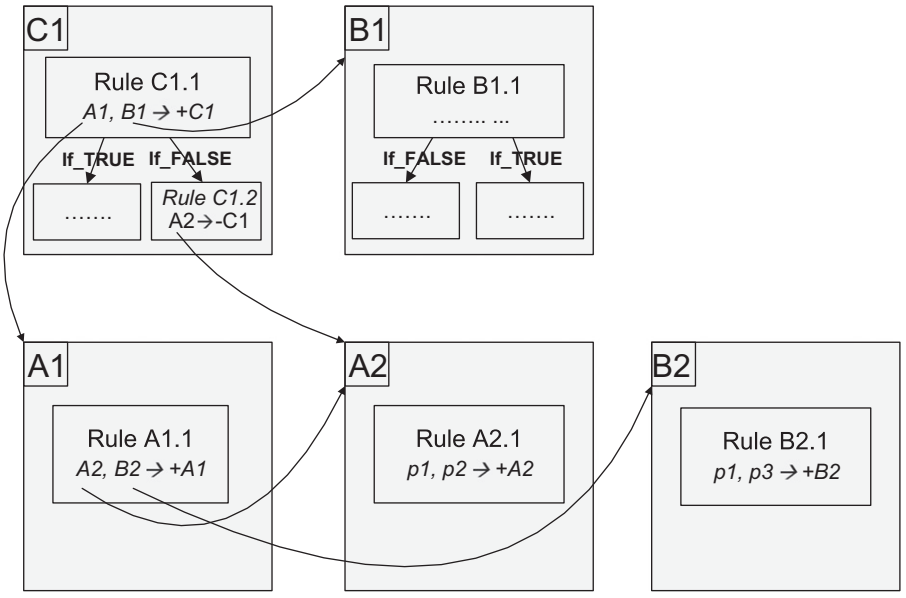

Fig. 1. An example of nested rules. An update in concept A2 can cause changes in the meaning of rules C1.1, C1.2, and A1.1 of the knowledge base (Beydoun and Hoffmann, 1997).

This list contains attributes satisfied by the case which triggered addition or *r*, and it excludes all attributes satisfied by any of the cases covered by the parent of *r*.

4. System architecture of the *DesignAssistant* software

Fig. 2 illustrates a scheme of the complete KA *DesignAssistant* system. A brief discussion follows which explains the function of each subsystem.

1. 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.
2. Knowledge acquisition module: gets the expert input through the user interface. It maintains the knowledge base as well as the knowledge (case) database.
3. (Search control) knowledge database: It stores what the expert expresses as search control knowledge. Using this knowledge, given the possible next states from the previous module, this module passes through only those states seen as worth pursuing deeper during the search. This module contains the larger part of the domain knowledge. This knowledge base is built during the actual knowledge acquisition process.
4. Knowledge acquisition 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 database to give these hints.
5. Knowledge (case) database: contains all cases classified by the expert. It allows retrieval of these cases according to their classifications time stamped. Thus, this database 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.
6. Search engine: It controls the generation of the search tree through interactions with the knowledge base. It saves local decisions about search tree nodes in the working memory. It also examines the pruned search tree and chooses an answer according to one of several evaluation criteria set by the user.
7. Domain specific search operators module: contains a set of search operators forming an instance generator. Given a particular search state, this module can generate all immediate next possible states. This module also allows the knowledge base to interpret any domain specific primitives used by the expert while describing his/her knowledge to the system. Also is where mathematical functions are declared.
8. Working memory: stores the progress of the search, which is often used by the expert to explain his/her decisions. In electronic circuit design, for example, and solving a component placement problem, a circuit designer chooses his/her next step based on a rough plan; this plan prevails in the progress

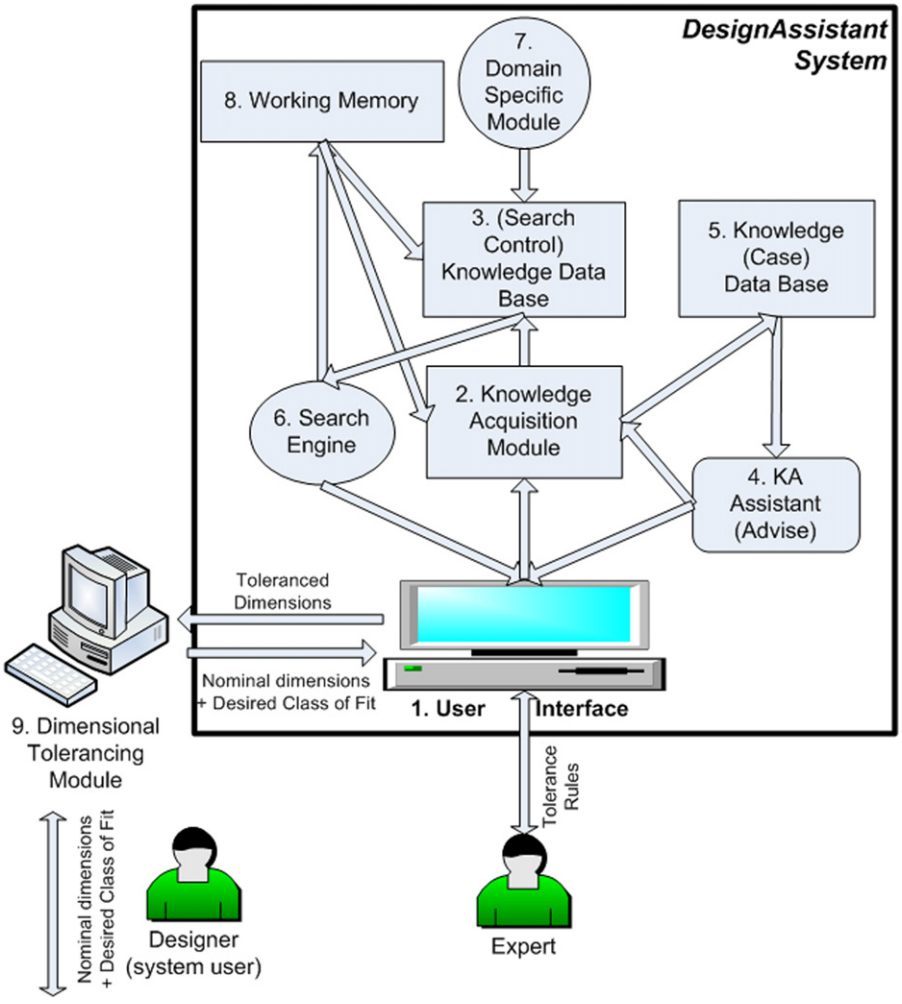

Fig. 2. A schematic representation of the complete *DesignAssistant* system.

Table 1
Tolerancing knowledge: lower (L) and upper limit (U) values for both the hole and shaft.

		Hole limit		Shaft limits	
Class of fit	Explanation of fit	Lower	Upper	Lower	Upper
RC1	Close sliding	0	0.392	−0.588	−0.308
RC2	Sliding	0.392	0.571	−0.7	−0.308
RC3	Precision running	0.571	0.907	−1.542	−0.971
RC4	Close running	0.907	1.413	−1.879	−0.971
RC5	Medium running	0.907	1.413	−2.84	−1.932
RC6	Medium running	1.413	2.278	−3.345	−1.932
RC7	Free running	1.413	2.278	−4.631	−3.218
RC8	Loose running	2.278	3.57	−7.531	−5.253
LC1	Locational clearance	0	0.571	−0.392	0
LC2	Locational clearance	0.571	0.907	−0.571	0
LC3	Locational clearance	0.907	1.413	−0.907	0
LC4	Locational clearance	1.413	3.57	−2.278	0
LC5	Locational clearance	0	0.907	−0.879	−0.308
LC6	Locational clearance	0.907	2.278	−2.384	−0.971
LC7	Locational clearance	2.278	3.57	−4.211	−1.933
LC8	Locational clearance	2.278	3.57	−5.496	−3.218
LC9	Locational clearance	3.57	5.697	−8.823	−5.253
LT1	Locational transitional	0	0.907	−0.281	0.29
LT2	Locational transitional	0.907	1.413	−0.442	0.465
LT3	Locational transitional	0	0.907	0.083	0.654
LT4	Locational transitional	0.907	1.413	0.083	0.99
LT5	Locational transitional	0	0.907	0.656	1.227
LT6	Locational transitional	0	0.907	0.656	1.563
LN1	Locational interference	0	0.571	0.656	1.048
LN2	Locational interference	0.571	0.907	0.994	1.565
LN3	Locational interference	0.571	0.907	1.582	2.153
FN1	Light drive	0	0.571	1.66	2.052
FN2	Medium drive	0.571	0.907	2.717	3.288
FN3	Heavy drive	0.571	0.907	3.739	4.31
FN4	Force	0.571	0.907	5.44	6.011
FN5	Force	0.907	1.413	7.701	8.608

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 (i.e. search states) of the evolving search plan. This reduces computational requirements as these features can get used again at a later stage of the search.

9. Dimensional tolerancing module: this constitutes the 'back end' of the *DesignAssistant* system and may be a standalone module or a module that interfaces with an existing CAD tool. It contains the part's engineering description: shape, size (nominal dimensions), dimensional tolerances, and factors of fit, form, and function. 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. Equally feasible is the inverse problem where the input is a pair of existing dimensions one for the shaft and the other for the hole with this module retrieving an applicable class of fit (if present).

5. The classical fit problem

A classical example common to the discipline of mechanical engineering by which the relative fit of a shaft to a hole is studied (Shigley and Mischke, 1986). Depending on the desired mating functionality between a shaft and a hole, Table 1 lists 31 different classes of interference/clearance fits. 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). Table 1 lists limit values—designated L for the lower tolerance limit and U for the upper tolerance limit. 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:

$$\begin{aligned}\text{Lower bound tolerance} &= LD^{0.333}\\ \text{Upper bound tolerance} &= UD^{0.333}\end{aligned} \tag{1}$$

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:

$$\text{Toleranced dimension} = \text{Nominal dimension}^{+\text{Upper bound tolerance}}_{-\text{Lower bound tolerance}} \tag{2}$$

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 the application in *tolerancing* is described.

6. Example: NRDR application to dimensional tolerancing

The above stated approach is applied to the tolerancing problem in mechanical design. The classical example of the shaft-in-a-hole mechanical fit problem as introduced above is

considered. 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. Two functions are needed for the proper evaluation of this hole–shaft fit problem. One 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 Eq. (1) such that

$$LD^{1/3} < (R-D)1000 < UD^{1/3} \tag{3}$$

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. The formula returns true if the inequality is true, otherwise false. The application of Eq. (3) necessitated that RDR is extended to allow for the utilization of mathematical functions and the corresponding 'quantitative' values. This is a major extension over the existing applications where the focus of NRDR development has been on 'qualitative' rules (e.g. chess playing (Beydoun and Hoffmann, 1997)). The second function needed to check whether or not the case may be classified as *locational* is an equality check.

$$A = B \tag{4}$$

This function returns true if string A is equal to string B where A or B may have any value of one of the attributes. Defining a function includes both naming the function and its attributes, as well as specifying the type of each attribute.

6.1. The fits case generator

Specific to this application, the *Fits case generator* module generates cases of hole, shaft, and diameter to be fed to the NRDR program. These cases are such that each one corresponds to a 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 has a separate rule for each class. (For the excepted case with common limits, the conclusion of either of the two classes LN1 or LT5 was given).

Fits case generator reads from two *Text (MS-DOS)* files, which are created with the desired values in the described format in order to generate the cases. The first input file ⟨Limits.txt⟩ is a file created by Microsoft Excel. It contains the limits of the classes for which cases are to be generated. It also contains the nominal diameter of the case. Fig. 3 is an example of cases generated where the nominal diameter=1 in (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. ⟨Locational.txt⟩ is the second file which is also created by Microsoft Excel. It contains only one column. The first row is the name of the cases attribute "Locational", the second row is empty, the rest of the rows, in the order of the limits in the ⟨Limits.txt⟩ file, are the values of the "Locational" attribute, either 'Yes' or 'No'.

	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

Fig. 3. Example of the input file ⟨Limits.txt⟩ 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).

	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

Fig. 4. Example of the ⟨fits_cases.txt⟩ file (showing only the first 17 fit cases for illustration purposes).

Fits case generator outputs the file ⟨fits_cases.txt⟩ the format of which is similar to that of the cases prepared by Microsoft Excel to be loaded by NRDR. The file contains a random case for each class of fit introduced by the file of the limits. Fig. 4 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.

6.2. Fits domain construction

Having generated fit cases, a domain (here called *Fits Domain*) will have to be constructed. Defining the domain includes case preparation, and function compilation including specifying the names and types of the relevant attributes. For example, the attribute name *Nominal Diameter*, which represents the nominal diameter of the hole–shaft system the type of which is defined as 'NUMBER'. Other attributes are *Hole* (actual hole diameter) and *Shaft* (actual shaft diameter), which are both of the 'NUMBER' type. Cases generated from the *Fits generator* module are then loaded. Defining functions such as (3) involves naming the function and its attributes, as well as specifying the type of each attribute. Function attributes 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, all of which being of the "Number" type. For the function in (4), both attributes A and B are defined to be of the 'String' type. Utilizing Workspace in Microsoft visual C++, declaring these functions involves an algorithm, which results in passing all the attributes to the function. While declaring the function in (3) and in order to declare the condition for the function, two comparisons (joined with the logical AND operator) are made as shown in Fig. 5. Note that in the figure 'Defined' represents a function that returns true if all the attributes passed to it are pre-defined. Also note that 'OK' means the function should return true. If the condition is not satisfied, false is returned. Next in the algorithm, function in (4) is declared in a similar fashion, thus, completing the task of function declaration.

```
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
    }
}
```

Fig. 5. The steps required in compiling the function in (3) as seen in a Microsoft Visual C++ window.

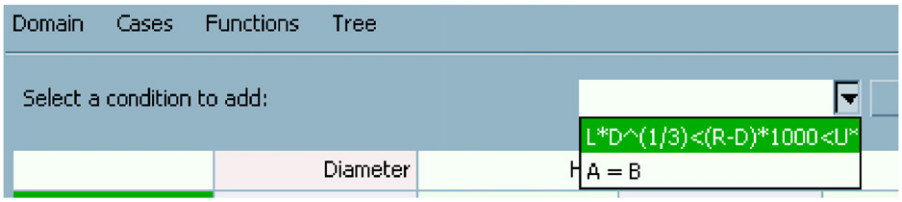

Fig. 6. Functions as they appear in *DesignAssistant* user interface.

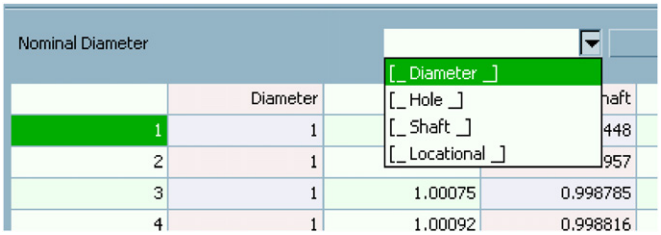

Fig. 7. Case attributes as they appear in *DesignAssistant* user interface.

Fig. 8. Adding conditions to functions in *DesignAssistant* user interface.

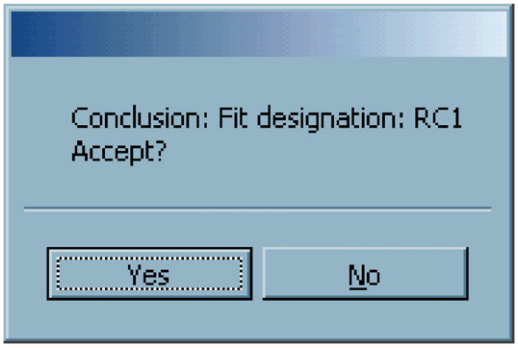

Fig. 9. Conclusion for RC1 as shown in *DesignAssistant* user interface.

6.3. The case validation module

Next, the *Case validation* module performs case validation sequentially starting from the first case of fit designated RC1. The expert is asked whether he accepts the conclusion of the tree. A 'DEFAULT' conclusion corresponds to the first rule of the tree that is always true. This default conclusion only appears if NRDR 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.

6.4. Adding rules

A function would have to be selected in order to add a new rule. Fig. 6 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 Fig. 7.

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 by selecting the second entry '_ Hole _' from the list in Fig. 7. The first condition for the hole has been added to the rule.

The condition for the shaft needs to be added for the rule to be complete (Fig. 8). The lower limit of the shaft for RC1 is -0.588. The upper limit is -0.308. Adding the 'Non-locational' condition for the function '*A=B*' requires that the attribute 'Locational' is assigned the value 'No'. To finish creating the rule, the condition: "*Fit designation: RC1*" is designated. 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*" as shown in Fig. 9. The rest of the rules are added in the same fashion.

6.5. Using the NRDR tree

When the expert has defined all the rules, the fully populated tree is, therefore, saved and will become available for later loading

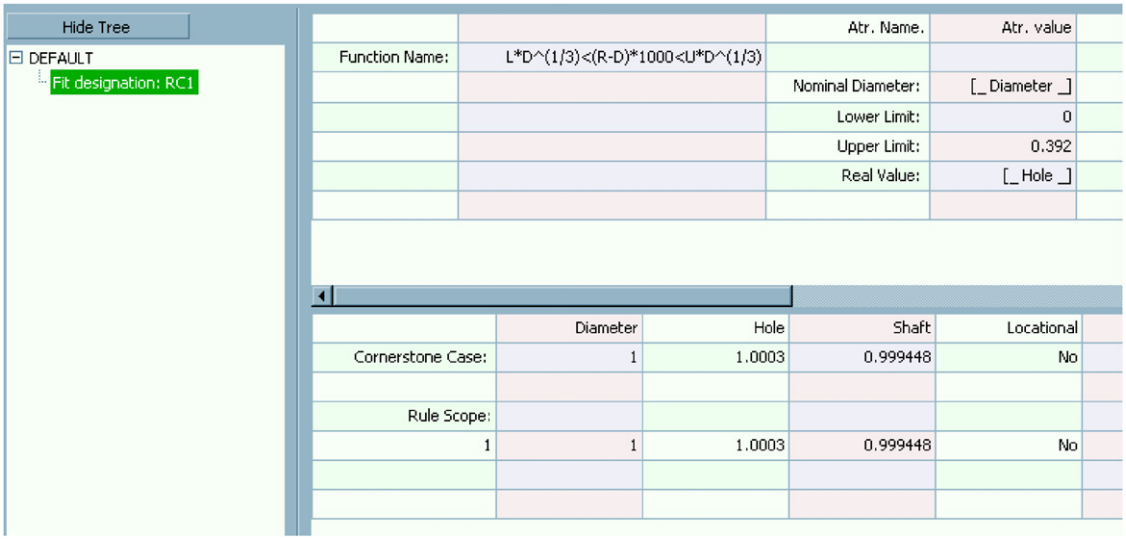

Fig. 10. A populated tree as seen on the *User interface* for the cornerstone case RC1with the condition(s) of each rule, as well as the scope of the rule.

and viewing. Upon viewing, the tree will appear as shown in Fig. 10 complete with the condition(s) of each rule, the cornerstone case, and the scope of the rule.

7. Discussion and summary

Given the proliferation in digital transfer of files representing mechanical databases (geometries and drawings), the requirement that dimensions must be properly toleranced to reflect the design intent has been captured in such standards as ISO 10303 Product Data Representation and Exchange (STEP) Part 47 "Shape variation tolerances" (ISO, 1997). The implications of such requirements should be reflected via an increase in the AI content of mechanical computer-aided design and drafting (MCADD) to where tolerancing of dimensions may have to become fully automated in schemes executed *during* the actual design process and not at a later stage.

In this work, we demonstrated that one can efficiently build an effective intelligent system to incrementally capture expert designer's prescription in dimensional tolerancing. This was accomplished by utilizing a knowledge base system built on the Nested Ripple Down Rules (NRDR) method. This intelligent system was successfully demonstrated in this paper by automating the process of tolerancing nominal dimensions based on the classical mechanical fit problem between a shaft and a hole. The system is able to perform both forward and backward fit schemes for cases like the one presented above. A forward scheme means that given a class of fit, i.e. LC1, LC2, etc., the software will return the upper and lower tolerance bounds for the nominal shaft/hole diameter of interest. A backward scheme means that given actual hole/shaft diameters the software would correctly identify the relevant class of fit. Such a dimensional tolerancing knowledge management system may be integrated into smart CAD platform (see Fig. 2) to help mechanical designers become more effective by automating the task of dimensional tolerancing of their designs in the future. Such a system would help mechanical designers become more effective in the time-consuming dimensioning and tolerancing process of their designs given the relative complexity of some tolerancing schemes. Implicit benefits of utilizing such a smart system include:

1) Shortened product development process cycle when compared with a traditional dimension-and-tolerance-by-hand approach;
2) 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.

Acknowledgements

This work was financially supported by the University Research Board (URB) of the American University of Beirut (AUB). This support is gratefully acknowledged.

References

American National Standards Institute ANSI-Y14.5.1M, 1994M. In: Mathematical Definition of Dimensioning & Tolerancing. The American Society of Mechanical Engineers, ASME Press.
American National Standards Institute ANSI Y14.5M, 1994M. Dimensioning & tolerancing, The American Society of Mechanical Engineers. ASME Press.
Beydoun, G., Hoffmann, A., 1997. NRDR for the acquisition of search knowledge. Lecture Notes in Computer Science, Advanced Topics in Artificial Intelligence, 1342. Springer, Berlin/Heidelberg 177–186.
Beydoun, G., Hoffmann, A., 2000. Incremental acquisition of search knowledge. International Journal of Human Computer Interactions 52 (3), 493–530.
Beydoun, G., Hoffmann, A., 2001. Theoretical framework of incremental hierarchical knowledge acquisition. International Journal of Human Computer Studies, Academic Press 54 (3), 407–452.
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.
Chapman, M.P., 1999. Design engineering—a need to rethink the solution using knowledge based engineering. Knowledge-Based Systems 12, 257–267.
Chiesi, F., Governi, L., 2003. Tolerance analysis with eM-TolMate. Journal of Computing and Information Science in Engineering 3 (1), 100–105.
Compton, P., Jansen., R., 1990. A philosophical basis for knowledge acquisition. Knowledge Acquisition 2, 241–257.
Compton, P., Preston, P., Yip, T., 1994. Local patching produces compact knowledge bases. In: The European Knowledge Acquisition Workshop (EKAW94).
Desrochers, A., 2003. A CAD/CAM representation model applied to tolerance transfer methods. Journal of Mechanical Design, Transactions of the ASME 125 (1), 14–22.
Finger, S., Tomiyama, T., Mantyla, M. (Eds.), 2000. IFIP TC5 WG5.2 Third Workshop on Knowledge Intensive CAD. Kluwer Academic Publishers.
Gaines, B.R., 1991. Induction and visualisation of rules with exceptions. In: Sixth Banff Knowledge Acquisition for Knowledge Base System Workshop (KAW91).
Hoffmann, A., Thakar, S., 1991. Acquiring knowledge by efficient query learning. In: 12th International Conference on Artificial Intelligence (IJCAI91).
Hu, X., Pang, J., Pang, Y., Atwood, M., Sun, W., Regli W.C., 2000. A survey on design rationale: representation, capture and retrieval. In: Proceedings of DETC'00,

2000 ASME Design Engineering Technical Conferences September 10–13, 2000, Baltimore, Maryland.

Hu, J., 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.

ISO 10303-42, 1999. Integrated generic resource: geometric and topological representation. International Organization for Standardization (ISO), second ed., 1999-06-30, Case postale 56, CH-1211 Geneva 20, Switzerland.

ISO 10303-47, 1997. Integrated generic resource: shape variation tolerances. 1997-07-29, ISO, Case postale 56, CH-1211 Geneva 20, Switzerland.

King, D.A., 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.

Kang, B., Compton, P., Preston, P., 1998. Multiple classification ripple down rules: evaluation and possibilities. In: Ninth AAAI-sponsored Banff Knowledge Acquisition for Knowledge Based Systems Workshop.

Linster, M., 1993. Explicit and operational models as a basis for second generation knowledge-acquisition tools. In: David, J.-M., Krivine, J.-P., Simmons, R. (Eds.), Second Generation Expert Systems. Springer-Verlag, pp. 477–506.

Meadows, J.D., 1995. Geometric dimensioning and tolerancing-applications and techniques for use in design, Manufacturing, and Inspection. Marcel Dekker, Inc..

Menzies, T., Compton, P., 1995. The (Extensive) implications of evaluation on the development of knowledge-based systems. In: Proceedings of the Ninth Banff Knowledge Acquisition for Knowledge Based Systems Workshop. pp 18.1–18.20.

Roy, U., 1994. Intelligent CAD system in concurrent engineering environment: a knowledge-based approach. Cybernetics and Systems 25 (4), 611–628.

Rezayat, M., 2000. The enterprise-web portal for life-cycle support. Computer-Aided Design 32 (2), 94–106.

Radack, G.M., Sterling, L.S., 1994. Reasoning about symbolic descriptions of mechanical parts. In: Dagli, C.H., Kusiak, A. (Eds.), ASME Series on International Advances in Design Productivity—Intelligent Systems in Design and Manufacturing. ASME Press, New York.

Shen, Z., 2003. Tolerance analysis with EDS/VisVSA. Journal of Computing and Information Science in Engineering 3 (1), 95–99.

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

Shigley, J.E., Mischke, C.R. (Eds.), 1986. Standard Handbook of Machine Design. McGraw-Hill Book Company.

Wang, N., 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.

Wu, W., Rao, S.S., 2004. Interval approach for the modeling of tolerances and clearances in mechanism analysis. Journal of Mechanical Design, Transactions of the ASME 126 (4), 581–592.