Proceedings of the 10th IASTED International Conference
INTELLIGENT SYSTEMS AND CONTROL
November 19–21, 2007, Cambridge, MA, USA
ISBN Hardcopy: 978-0-88986-707-9 / CD: 978-0-88986-708-6

MAICE – A TOOL FOR KNOWLEDGE REPRESENTATION

Fabricio M. Oliveira[†], Leizer Schnitman[†], Herman A. Lepkison[†], J. F. Corrêa[††], Almir L. Néri Jr.[††], and J. A. M. Felippe de Souza[†††]

[†]UFBA - Universidade Federal da Bahia, Brazil, fmota@ufba.br, leizer@ufba.br, herman@ufba.br,
[††]PETROBRÁS – Unidade de Exploração e Produção da Bahia, Brazil, jfcorrea@petrobras.com.br, almir.neri@petrobras.com.br
[†††]UBI - Universidade da Beira Interior, Covilhã, Portugal, felippe@ubi.pt

ABSTRACT
In the present work the software MAICE is shown as a solution for knowledge representation. This software is a visual modeling tool which allows to represent the rules that define a given knowledge universe, independently of specific application area in order to make this knowledge available to be integrated with other systems afterwards. Using this tool, it is possible to separate from the knowledge of others systems which need use it. This software also brings resources that allow the expert modeling and validating the knowledge and, moreover, the possibility of to diminish the responsibility from the developers with respect to producing the knowledge rules inside the source code of their systems. This software has been successfully integrated with a petroleum wells automation systems. One of MAICE's assets is the possibility of modeling using *fuzzy logic*, which could be a powerful tool for modeling knowledge using simple rules. Once finished, this knowledge could be easily integrated with other control and real-time systems, which could need these diagnoses for managing maintenance and emergency actions. This paper also shows some of this software's applications.

KEY WORDS
Knowledge Representation, Knowledge Modeling, Fuzzy Logic, Electrical Systems Management, Electrical Transformer Diagnosis.

1. Introduction

To create systems based on expert knowledge is not a simple task. System designers and developers do not always have all the knowledge that is needed to introduce it into their systems, which make them strongly depend on the original specifications.
On the other hand, many experts from many areas do not need to know about designing and programming Information Systems. Faults or gaps within the communication between an expert and a programmer during the specification could cause problems such as late deadlines, inadequate system designs, or even serious errors in these software's operations.
Introducing the rules that works in a given knowledge universe directly in the source code, by the analysts, designers and developers, might not to be a good option. Problems that vary since the lack of speed while developing the solutions up to creating modules and routines that do not satisfy the actual customer's necessity, could appear due to the missing mutual knowledge between developers and experts, joined to communication problems when the solution is specified.
In addition, it is common that the same knowledge needs to be applied in multiple systems inside the same corporation. If this knowledge is introduced directly in these systems as code lines, redundancies will be created. Hence, not only the developing cost would be greater, but also the maintenance use will become hard and dangerous. Updating or alterations in the knowledge model could either lead to a team overwork or to take the systems into inconsistent procedures.
The present paper shows a software solution called MAICE – *Metodologia para Armazenamento e Integração do Conhecimento Especialista* (Methodology for Storing and Integrating the Expert Knowledge), which has the aim to separate the knowledge rules from the customer's systems. On the other hand, MAICE allows the expert to model his knowledge quickly and in a simple way. Resources such as for example fuzzy logic are offered to model, test and validate the knowledge, in a clear and unambiguous way. Once the knowledge is validated, it could be easily integrated with other systems in order to reduce both their costs and their risks from altering the system's source code. In addition, the MAICE tool allows discharging system's designers and programmers from the responsibility to thoroughly know the rules from that specific area of knowledge.
The MAICE tool was developed as a research project from the Research and Technological Qualification Center in Industrial Automation (CTAI), from Federal University of Bahia (UFBA) in Brazil, by demand of Petróleo Brasileiro S.A. Company (PETROBRAS). MAICE as created is bound to be applied to the knowledge of Petroleum Elevation Methods, and has

been integrated with a system called SGPA (Automated Wells Management System) and other corporative systems within PETROBRAS Company.
Here it is shows the proposed methodology for modeling the knowledge, illustrating in this way some of the functionalities that this tool is capable to bring about. The paper is divided into five sections. Here in the introduction it is also presented some notions about the proposed solution. In the second section, a brief approach about the methodology for knowledge representation is shown. It also includes a fuzzy logic overview. The third section presents the MAICE software, its purposes, user's interface and interactions with other systems. In fourth section, an application example for electrical transformer diagnosis is presented. The last section presents some conclusions and also some propositions and perspectives for future work.

2. Knowledge Representation Methodology

This section presents the MAICE methodology for representation of the knowledge. The methodology is based on block diagrams and connections. Here it is considered that diagrams could be an interesting language to allow the experts to model their knowledge and provide an automatic processing in a moment subsequent. The blocks in the diagrams, here called *operational blocks*, refer to functional elements which are able to perform some operation and produce a result on its output pin, as a response of its input pins values.
The way to perform interactions between the blocks is using connections. A connection is represented by the lines that connect two blocks. For that, it is mandatory always to link the output pin of a source block to an input pin of a destination block, so it can perform an information supply chaining. The connections define the way where the flow of information will pass by. An example of knowledge representation using block diagrams is shown at figure 1. Notice that the blocks are triangular shaped and the right block depends on information produced by the left block.

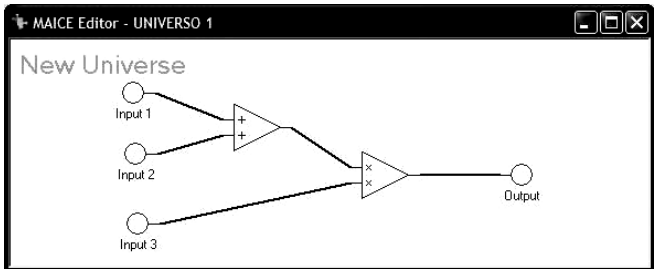

Figure 1 Example of knowledge representation using blocks diagrams methodology.

So, we consider that the main objective is to solve a problem that requires the expert knowledge for its solution. The answer for this problem is typically modeled as an output variable. This variable, as any other, can assume one of many different types offered: *analog*, *fuzzy*, *digital* and *multi-state*, as explained below:

- ***analog variables*:** refers to numeric variables, which can assume continuous values contained in a established real domain. Example: *height*, *weight*, *voltage*, etc;
- ***digital variables*:** refers to variables which can assume a truth value (*yes* or *no*). For that, there is a large set of digital operation blocks available, able to processing them. Examples: *valve opened*, *alarm on*, *condition reached*, etc;
- ***fuzzy variables*:** a specific type of analog, which contains fuzzy sets assigned, and that is able for fuzzy processing, respecting to the Mamdani model (see the session II-A) [1];
- ***multi-state variables*:** a type of variable created for allowing the modeling variables that can assume any amount of non-numeric discrete values. Examples: *surface material (iron, brass, zinc, aluminum)*, *type of power supply (DC, AC-one-phase, AC-three-phases)*, *sex (male, female)*, etc.

In a second step of the modeling, other variables – the *input variables* – are also required for the processing. These variables hold all the required information, determined in the model, which will be necessary to perform any functional processing and to produce the final result. The number, the types and the purpose of these variables should be defined by the expert.
Once both the output and the input variables are defined, the third step is to perform a functional relationship between these data by means of blocks and connections. For that, many different types of blocks are offered, as mentioned below:

- ***for analog processing:*** sum (numeric sum or subtraction), product (numeric product or division), module (arithmetic module), gain (multiplication by a constant factor), etc;
- ***for digital processing:*** (switchers, *and*, *or*, and *not* gates, with possibilities of inverting their inputs or outputs), etc;
- ***for multi-state processing:*** multi-switchers, etc;
- ***for general purpose:*** (with any type of variables) inputs, outputs, switchers, data converters (analog to digital, digital to analog, analog to multi-state, etc), constant sources, subsystems (blocks that contains other systems internally), etc;
- ***for fuzzy operations:*** fuzzy blocks.

There is a special emphasis here to fuzzy block, because it was the first operation block offered by MAICE. There are some advantages in using fuzzy logic for numeric processing and knowledge representation, as it is shown in next paragraphs.

2.1 Fuzzy Logic

Fuzzy logic is a powerful tool for building mathematical analysis methods by using linguistic rules. From the basic concepts of fuzzy logic, it is possible to establish fuzzy

set classes of values for chosen variables, with qualifiers, known as *membership functions*. Unlike the classic logic where there are only two possible states for a variable, fuzzy logic allows for a large range of different levels in which qualifiers can be related [2, 3, 4]. For example, by using fuzzy logic it is possible to assign to a variable the qualifiers "high", "medium" and "low". This variable will assume different levels or, possibilities of being associated with each one, depending on its value and its constructed model.

So, the knowledge that relates the variables of the fuzzy system is then built by using simple rules that deals with the membership functions. In this way all the formal calculation that would be necessary to solve the problem by an expert in the subject, independently of being complex or not, becomes unnecessary, once he knows the rules that relate these variables.

The necessary computation is then achieved by means of a resolution method called *defuzzyfication*. There are two classic approaches to the computational procedures in fuzzy logic: Mamdani [1] and Takagi-Sugeno (TS) [2]. MAICE software presently carries out only the computational structures of Mamdani approach, although it is meant to be extended to the computational structures of TS in the future.

For the sake of simplicity, let us consider a Mamdani model with only two inputs '*x*' and '*y*', and one output '*z*'. These variables are qualified by their membership function which represents its analysis in a given universe of discussion. Figure 2 illustrates the fuzzy sets "Low" "Medium" and "High" for variable '*x*'.

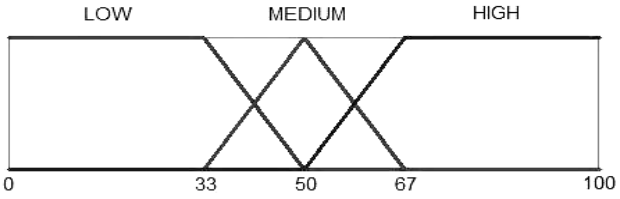

Figure 2 Example of variable with its fuzzy sets: *Low*, *Medium* and *High*.

The line that illustrates the behaviour of each qualifier in the domain of the fuzzy variable '*x*' represents, for each possible value of '*x*', the membership level $\mu(x)$ of '*x*' with each qualifier. For example, for $x = 0$, $\mu_{LOW}(x)$ presents the maximum level of membership and means that $\mu_{LOW}(x)=1$. Alternatively, for the same value of $x = 0$ it is clear that both $\mu_{MEDIUM}(x)$ and $\mu_{HIGH}(x)$ present the minimum level of membership, that is the value 0.

Similarly, if we consider that $x = 50$, them both $\mu_{LOW}(x)$ and $\mu_{HIGH}(x)$ will be 0 and $\mu_{MEDIUM}(x) = 1$. Now considering '*y*' as another fuzzy variable which is defined by its fuzzy sets "Weak", "Normal" and "Strong". So, the variable '*y*' will also have three membership functions $\mu(y)$ to represents its level of membership: $\mu_{WEAK}(y)$, $\mu_{NORMAL}(y)$ and $\mu_{STRONG}(y)$.

Finally, let us consider the fuzzy variable '*z*' defined by its fuzzy sets "Mild", "Normal" and "Severe", each with its membership functions . This *fuzzy* system will be defined now in such way that *z* is a function $f(x, y)$ of the variables '*x*' and '*y*'. The aim of this *fuzzy* system is to produce the function $f(x, y)$ in order to calculate *z* without necessity to know the mathematical relations that governs these variables.

One wishes that the *fuzzy* element is able to get and process the rules that relate the input and the output variables. Each rule will fulfil a standard format of logic implication of the kind: Antecedents → Consequent, such as, for example, **if '*x*' is *high* and '*y*' is *weak*, then '*z*' will be *mild***. For each rule and for each input variable of the system an *antecedent* is needed in order to be associated to the membership functions of this variable. Each rule will also have a *consequent* that will be associated to one of the membership functions of output variable. The generic model of the rules for this system is:

If 'x' is A_i and 'y' is B_j, then 'z' is C_k.

where:

- 'A_i' is the i^{th} fuzzy set for the variable '*x*';
- 'B_j' is the j^{th} fuzzy set for the variable '*y*'; and
- 'C_k' is the k^{th} fuzzy set for the variable '*z*'.

The numerical value is then computed based on the rules of the Mamdani model.

3. The Software MAICE Overview

MAICE is a system that implements a Methodology for Storing and Integrating the Expert Knowledge. Its purpose is to allow experts to edit knowledge, and afterwards applying it, by means of integration, in real time, into other systems which use this knowledge for specific purposes. MAICE is based on the idea that the rules and relations composed by a given knowledge universe can be quickly modeled by the experts, in order to be easily integrated into the client systems. Thus, the knowledge is stored into a repository (UKM files – Universal Knowledge Model) that could be accessed by other systems.

Using MAICE, problems related to specification or communication faults, excessive spending of time or maintenance works, and other problems provided by the updating of the knowledge, either are reduced or disappear.

The software allows experts to model their own knowledge without being needed intervention from the persons who will develop or support these systems. The expert could also test and store his knowledge into a unique and shared repository, so it could be effectively used by the whole company and partners. Another important advantage is that new versions of the knowledge do not require maintenance of the client system's source code to work.

MAICE is composed by a set of modules, programs and tools that help the experts since the modeling,

maintaining, validating and integrating the knowledge with the other systems. When the software is installed in a given computational environment, its operation becomes simple and hence no more interventions by experts in computers or system developers are required.
MAICE's operation could be brake in two phases: the edition and integration of the knowledge, which are explained as follow:

3.1 The Edition Phase

At the first moment, or the edition phase, the expert could use the tool called *MAICE Editor* for modeling the knowledge using diagrams, with lines and shape entities, using visual components. At this stage the emphasis is pointed to the edition tool and to the data repository which contains the knowledge. Yet, using this tool it is possible to test and validate if the knowledge rules recently applied satisfy to the expected behavior. Once if the knowledge is validated, it will be stored into a shared repository (UKM file), to be passed to the *application phase*.

3.2 The Application Phase

In this phase, it is not necessary to use the MAICE Editor tool. The knowledge stored in the repository is viewed and processed by some modules, developed specifically for integration with other systems. It is important to notice that it is possible to integrate these modules with software written in many different types of programming languages or environments, by means of DLL files or other alternatives.
Hence, the client system can require an automatic diagnosis or other information from MAICE, simply informing the current context from the analysis universe, based on variables or conditions, modeled into the knowledge universe. Thus, MAICE will consult their data bases, will perform the procedures according to the defined rules, and will show immediately the result or diagnosis of the system.
The MAICE tool had its first integration with the SGPA system – a system used by PETROBRAS Company in Brazil, through its Bahia Business Unit [5, 6, 7]. Since then, it has been used for modeling and integration in various Petroleum Elevation Methods, for example. Research is being carried out in the following subjects: SCP (Submerse Centrifuge Pump), PCP (Progressing Cavity Pump), MP (Mechanic Pump) and IGL (Intermittent Gas Lift) methods.

4. Application in Electrical Transformer Diagnosis

Here, it is presented an application case of using MAICE for modeling how an expert could evaluate the adequate functioning of an electrical transformer.
This aim is to perform a diagnosis for preventive maintenance, substitution or repair of the transformer, based on expert knowledge, modeled and automated using MAICE. These maintenance actions can be used to improve the transformer lifetime or reduce the costs [8]. Firstly, a knowledge universe was created and named *Transformer Analysis*. Hence, every entity modeled is put inside this model, which can be called *universe*.
The final evaluation result of the universe was modeled as a fuzzy variable called *Transformer Condition* – with domain defined in the range 0% to 100% – that shows how good or how bad the transformer is working, and may require some maintenance intervention. Four fuzzy sets are assigned to this variable: *emergency*, *critical*, *alert* and *normal*. Their functional behavior is showed in figure 3.

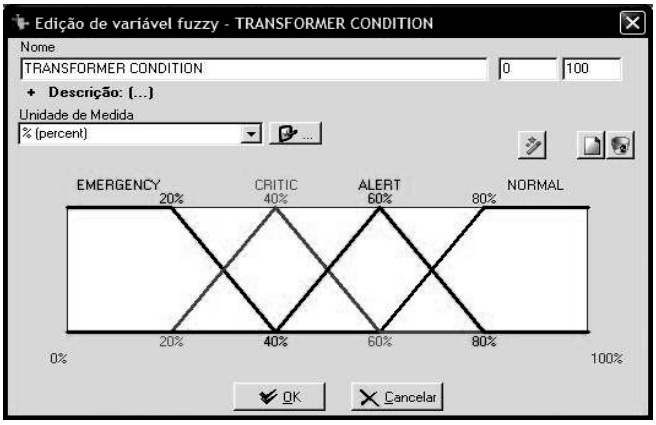

Figure 3: Fuzzy variable *Transformer Condition*, with its four fuzzy sets: *Emergency*, *Critic*, *Alert* and *Normal*.

The *Transformer Condition* is based on some input variables. The input are typically the information needed, used by the expert to deduce final or partial diagnosis. In this case, the main variables are: *gas production* (the amount of gas produced); *measured current* (the actual current measured in the primary winding); *nominal current* (expected current for the primary winding during project); *core temperature*; *oil temperature*; *winding temperature*, and *protection acted* (information on whether or not the system overload protection has acted or not).

Hence, the knowledge modeling consists in to insert operational blocks linked is such way to perform a functional relationship between the input variables – mentioned above – and the output variable *Transformer Condition*. The main diagram that represents this model can be seen in the figure 4.
In this diagram: the inputs are represented by the circles at left; the output is represented as the foremost circle on the right; the squares represent the operational blocks, which process information; and the lines represent the connection between these blocks, where the information will flow, from the inputs to the output block. The legend that appears beside each block pin represents test data values, which can be used for validating the model in a step later on.

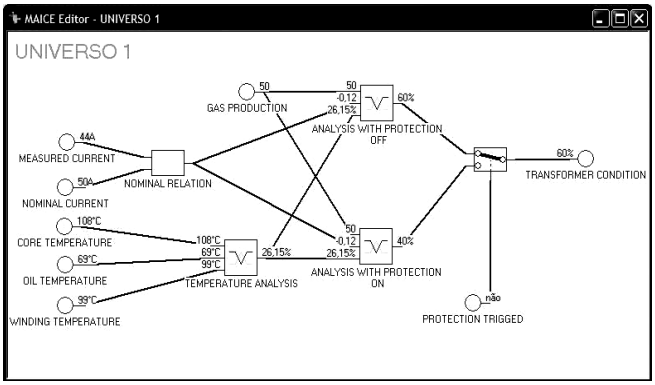

Figure 4: Main diagram, representing the knowledge rules about the *Transformer Analysis* universe.

3.3 Modeling the Knowledge

Once modeling a preliminary part of the knowledge, a fuzzy block was inserted at the left side of the diagram, called *Temperature Analysis*. A fuzzy block is an operational block which is able to perform fuzzy operations based on Mamdani model [1]. In this case, the aim is to allow the performance of a fuzzy-relationship between three input variables – the "temperatures measured on these transformer parts" (*oil*, *core* and *winding*) – and compress them into a single variable, called *"Preliminary Temperature Analysis"*, as a preliminary conclusion. Three fuzzy sets are assigned to each input variable: *"cold"*, *"medium"* and *"hot"*, as illustrated in figure 5 for the input variable "Core Temperature". On the other hand, the output variable will have the fuzzy values *"normal"*, *"alert"* and *"critical"* assigned.

Once a fuzzy block is configured with all input and output variables, the way to define its functional behavior is to insert fuzzy rules that assign the input fuzzy values – as "causes" – to a corresponding output fuzzy value – as a "consequence". In this case, an example of a fuzzy rule for *Analysis of Temperatures* could be semantically defined as follow:

> *if* **Core Temperature** *is* **hot** *and* **Oil Temperature** *is* **medium** *and* **Winding Temperature** *is* **hot** *then* **Preliminary Analysis of Temperature** *is* **critical**.

The actual need for modeling a *Preliminary Analysis of Temperatures* is to simplify the number of rules that will fulfill the fuzzy analysis afterwards (*Analysis with Protection On* and *Analysis with Protection Off*). In this way, instead of three temperature variables, it is necessary only one to perform the fuzzy rules. Considering that, while modeling a fuzzy block, the maximum number of different rules is the combination of all fuzzy sets assigned to each input variable. In the present case, the number of rules in each fuzzy block was reduced from 243 to 27.

In MAICE, the way to define fuzzy rules is filling a grid, as illustrated in figure 6 (left side). In the example shown, the first three columns represent the input variables, with their possible fuzzy values. The fourth and last column represents the output variable, where the user must fill with the correspondent output fuzzy values. The number of rows in the grid corresponds to the number of rules of the input fuzzy sets for all variables and combinations.

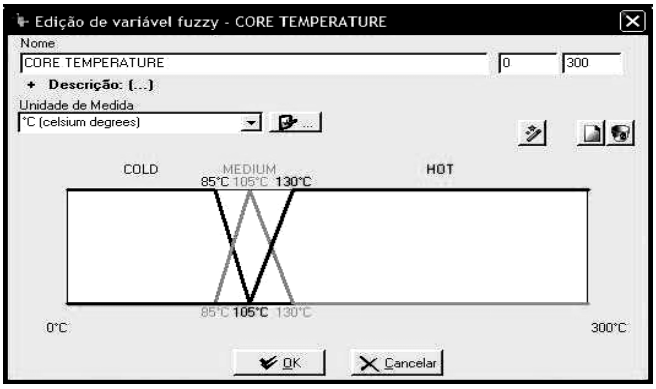

Figure 5 Fuzzy variable *Core Temperature*, assigned to three fuzzy sets: *Cold*, *Medium* and *Hot*.

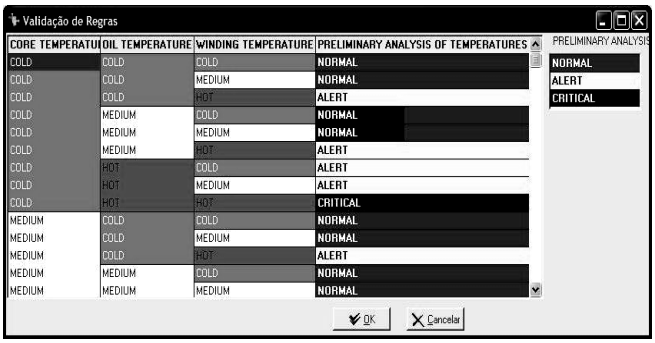

Figure 6 Fuzzy Rules Properties Editor, for the fuzzy block *Preliminary Analysis of Temperatures*.

Some tools are available to make the rules configuration user friendly, such as *drag and drop* options, that allows catching an output value from an option list (at the right), and releasing it at any position of the grid (at the left).

Looking at the left side of the diagram presented in figure 4, another interesting ability can be seen. That is a *subsystem* called *Nominal Relation*. In MAICE a subsystem can be understood as a configurable operational block, which is able to perform any functional relation based in internal blocks and connections (that includes the use of other subsystems). Internally, a subsystem behaves as an isolated universe, but it can be used recursively as many times as it is necessary.

The subsystem *Nominal Relation* has as its purpose to produce a numeric comparison between two analog variables for the same context – the nominal (or projected) and the actual (or measured) variables. This comparison aims to point how far it is, (proportionally), the actual variable from the nominal one. In this case, this subsystem was used for comparing the nominal and the measured currents for the transformer. Nominal Relation (*R*) produces: a real positive number if the actual (*A*) variable is greater than the nominal (*N*); a real negative number if the actual variable is lower than the nominal; and zero (0) if both are equals, according with the

following formula:

$$R = \frac{A - N}{N} \quad (1)$$

The internal view of the subsystem *Nominal Relation* is shown in figure 7. In this case, two arithmetic blocks were used, in order to perform both the subtraction and the division.

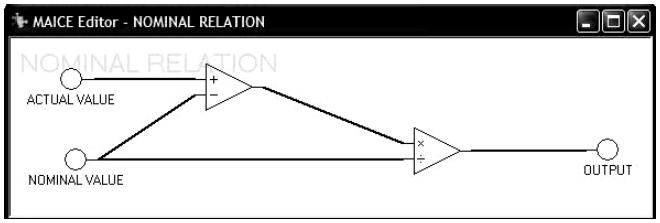

Figure 7 Internal view of the subsystem *Nominal Relation*.

Returning to the main diagram (figure 4), after processing the *nominal relation*, a fuzzy variable called *Relation of Current* is produced, with the fuzzy values: "*below*", "*normal*" and "*above*". This variable will be useful for the next step of processing. Further to the right, in this diagram (figure 4), two more fuzzy blocks can be seen: "*Analysis with Protection Off*" and "*Analysis with Protection On*". They are very similar to each other, and intend to deduce a preliminary condition on the transformer. However, they are based on the hypotheses of whether protection is triggered or not.

The data listed until now are modeled as analog or fuzzy variables. However, there is a digital input variable *Protection Trigged*, which indicates if the protection device has acted, due to some overload in the system. For each possible value this variable could assume – "*yes*" or "*no*" – a different analysis will be necessary, once they will yield different conclusions. So, a *switcher* block is used, as seen in the right part of figure 4.

The switcher in this diagram has the aim to select one of two sub-analysis, based on the information of the digital variable *Protection Trigged* as a controller variable. This means that if *Protection Trigged* is *yes*, the upper input of the block is enabled; if not, then the lower input will be enabled instead. So, it is possible to select the way where the information comes from, based on the value of a digital or a multi-state variable.

Despite the fact that the two fuzzy blocks which supply both alternatives in the switch are similar, the rules for *Analysis with Protection On* will be different from the rules for *Analysis with Protection Off*. One considers that the protection alarm indicates that something abnormal happened to the system. However, note that the input variables for these two fuzzy blocks are the same: *Preliminary Analysis of Temperatures*, *Relation of Current* and *Gas Production*.

3.4 Validating the Model

Considering that the knowledge model is constructed based on expert knowledge, it is possible to evaluate if the function relation between the system variables are working correctly. For that, MAICE has some tools available in order to allow the expert to test his model.

The first one is a *real-time test tool*, as could be seen still in figure 4. Each pin, on each block in diagram will contain a default value – shown above the corresponding pin – that can be changed by the user. When a new block or a new connection is inserted to the model, all the consequent values will be recalculated, in order to allow the user to evaluate his model while constructing it.

Another way to validate the functional behavior of the system is using the diagram test window, as shown in figure 8. Here it is possible adjust the values from the input variables (left side) and see the corresponding behavior of the output one (right side). For fuzzy variables, it is possible to see the level or possibilities of each fuzzy set related.

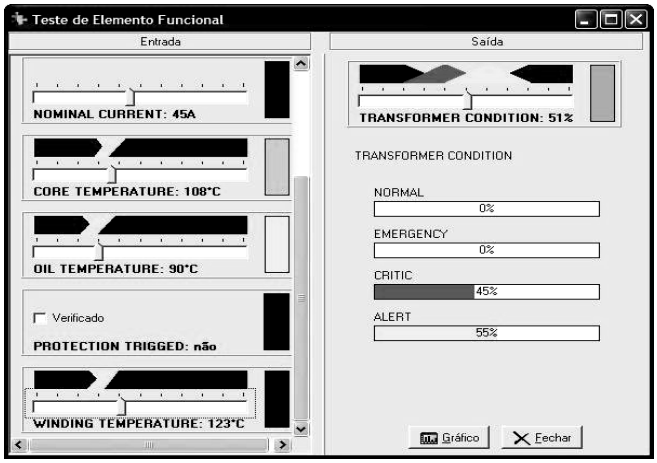

Figure 8 Tool available for diagram test.

Similarly to diagrams, this tool can also be used for testing the various parts of the diagram separately, such as subsystems, fuzzy blocks or switchers. Notice that different variables types also have different input controls assigned. In figure 8, the digital variable *Protection Trigged* has a check box, while the analog variable *Oil Temperature* is changed by a slider control.

There is another interesting tool available which allows for plotting three-dimensional graphics of the output in function from pairs of input values, as shown in figure 9. Here it is shown a surface representing the Transformer Condition in function of two variables, *Gas Production* and *Measured Current*. For this condition, all other input variables but the domain will be assumed to be constant, according to the values defined previously in the test window (figure 8). Any amount of 3-D graphic windows can be opened simultaneously, each one with a different combination of input variables, chosen by the user.

The straight line that can be seen in the center of the diagram marks the *operation point* for the input values, that is the position chosen for the variables represented by *x* and *y* in the graphic. Once these values are changed in

the test window, the operation point also changes in diagram.

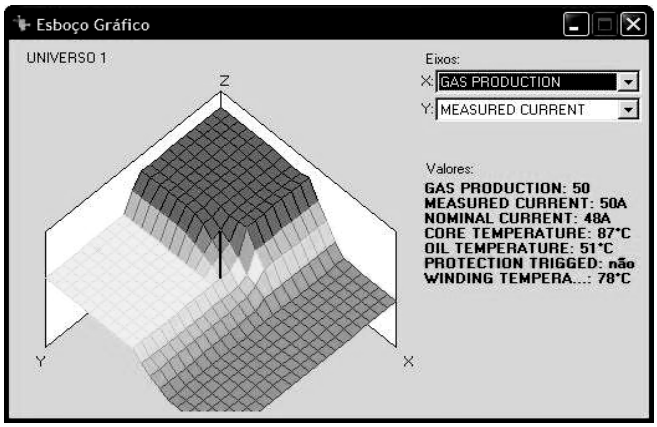

Figure 9 Graphic plotted for the output variable in function of two input variables.

5. Conclusion

Many problems are assigned to fit the expert knowledge directly into the source code of the customers' systems which needs it. The software MAICE provides a relatively easy way to separate this knowledge from the source code, and therefore to allow for its modeling and validation, as well as integrating it into the systems which needs it.

MAICE is not a finished project, since it has been progressing constantly in order to supply different demands, with new abilities. Some of these abilities, expected in a short time, include: Artificial Neural Networks with training algorithms, variables containing arrays values, variables containing data structures, tools for optimization, and so on.

The MAICE tool is applicable to the context of electrical transformer diagnosis, as well as it was used for Petroleum artificial elevation methods. The software has been improved constantly, so that it could be well applied for different knowledge areas. The knowledge representation model presented in this paper was constructed, tested and validated by experts, but its actual purpose was not to be integrated to a customer system. However, the knowledge models on SCP and other Petroleum Elevation Methods are in fact integrated and operational.

References

[1] E. H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, *Int. Journal of Man-Machine Studies*, Vol.7, N.1, 1975, 1-13.

[2] T. Takagi, M. Sugeno, Fuzzy identification on systems and its applications to modeling and control, *IEEE Trans. on Systems, Man, and Cybern.*. Vol. 15 pp.116-132, 1985.

[3] R. Babuska, H. Verbruggen, An overview of fuzzy modeling for control, *Control Engineering Practice*, 4, 1996.

[4] P. Dash, et al., Fuzzy and neural controllers for dynamic systems: an overview, *Proc. Int. Conf. on Power Electronics and Drive Systems*, 2, 1997.

[5] J. F. Corrêa, et al., Fuzzy analysis for progressing cavity pump, *SPE – ATC&E*, 2004.

[6] J. F. Corrêa, et al., Development of an Intelligent Distributed Management System for Automated Wells (SGPA), *SPE – ATC&E*, 2002.

[7] J. F. Corrêa, et al., Intelligent Distributed Management System for Automated Wells (SGPA): experience and results, *SPE – ATC&E*, 2003.

[8] P. Jirutitijaroen, S. Singh, The effect of transformer maintenance parameters on reliability and cost: a probabilistic model, *EPSR*, 2004.