2011 IEEE International Conference on Fuzzy Systems
June 27-30, 2011, Taipei, Taiwan

An Intelligent Decision Support Tool Based on Belief Rule-Based Inference Methodology

Alberto Calzada, Jun Liu and Hui Wang
School of Computing and Mathematics
Faculty of Computing and Engineering
University of Ulster, Northern Ireland, UK
albertocalsa@gmail.com; {j.liu, h.wang}@ulster.ac.uk

Luis Martinez
School of Computing
University of Jaén
Jaén, Spain
luis.martinez@ujaen.es

Anil Kashyap
School of the Built Environment
University of Ulster
Northern Ireland, UK
a.kashyap@ulster.ac.uk

***Abstract* — Taking into account the need of handling hybrid information with uncertainty in human decision making, a new belief rule-base inference methodology (RIMER) has been recently proposed. RIMER approach and its relevant extensions have proved to be highly positive solving decision problems. However, for an end user it is difficult to implement the methods and algorithms from the raw equations in order to solve a specific problem. This paper presents a decision support tool based on the RIMER approach that facilitates its implementation and use to end-users. The overall structure and main functionalities of the tool are outlined, followed by an example to illustrate the use of this tool for applications.**

Keywords* — *knowledge-based system, belief rule base, decision making, uncertainty, decision support system.

I. INTRODUCTION

Nowadays, large amount and heterogeneous information provided by different sources is common for decision making problems, this fact implies a high complexity that is rather difficult of managing for human beings without any support. Consequently, more and more computer based decision-making support has been accepted and developed, to provide a stress-free view of the situation, joining the individuals intellectual resources with the capabilities of computers for improving the quality of decisions [1].

The use of rules is one of the most common ways in many tools for knowledge representation and inference in decision problems. Knowledge-based systems, usually constructed from human knowledge by means of if-then rules, have been one of the most visible and fastest growing branches of artificial intelligence [2]. In the design and implementation of knowledge-based systems for supporting human decision making, it is necessary and inevitable to use a scheme for representing and processing imprecise, incomplete and uncertain information in conjunction with precise data. Because of the need of handling hybrid information with uncertainty in human decision making, the belief Rule-base Inference Methodology using the Evidential Reasoning approach – RIMER was introduced in [3]. This methodology uses a belief structure for modelling hybrid rule-bases and an evidential reasoning (ER) approach [4] for making inference in the belief rule-based system by combining decision science [5], rule-based systems, and Dempster-Shafer (D-S) theory of evidence [6-7]. Therefore in RIMER a rule-base is designed with belief degrees embedded in the entire consequent terms of a rule, called belief rule-base (BRB), to capture nonlinear causal relationships as well as uncertainty. RIMER approach has been further investigated and extended in [8], denoted as RIMER$^+$, where an extended rule base is designed with belief degrees embedded in the entire consequent terms as well as in the entire antecedent terms of each rule. In addition, a simple but efficient method for automatically generating such extended belief rule-base from numerical data is proposed involving low time cost iterative learning procedure and simple rule generation mechanisms [8]. RIMER results and its relevant extensions have proved to be highly positive solving decision problems applied to different areas, such as, among others, safety and risk analysis, oil pipe leak detection and some other application in engineering systems [9-13].

This paper presents a decision support software tool which aims to facilitate the implementation and use of RIMER and RIMER$^+$ to solve complex decision making problems. This tool presents a friendly end-user interface that allows the implementation of RIMER and RIMER$^+$ in a transparent, informative and integrated way. Concretely, it offers the following functionalities: 1) A data base to store the input-output data; 2) a friendly user interface for problem structuring, information collection, and data presentation in both graphical and text formats; 3) The necessary modules for the rule-base which would be automatically generated based on the input-output sample data or assigned by domain experts and can be then represented in the interface; 4) management of heterogeneous input information in an integrated way; 5) inference algorithms to be applied once the rule-base is generated and the inputs are obtained; 6) a module to save and load solutions of a problem in both textual and graphical formats. Each functionality will be further detailed in the following sections.

This paper is organized as follows: RIMER and RIMER$^+$ are briefly overviewed in Section II. The architecture of the decision support tool and its functionalities are presented in Section III. Section IV introduces an illustrative example. This paper is concluded in Section V.

978-1-4244-7316-8/11/$26.00 ©2011 IEEE

II. OUTLINING RIMER APPROACH

RIMER [3] presents a new belief rule representation scheme to extend traditional IF–THEN rules using a belief structure. The belief rule base (BRB) introduced in RIMER approach is designed with belief degrees embedded in the consequent terms. It has been then extended into a new belief rule base with belief degrees embedded in the entire consequent terms as well as in the entire antecedent terms of each rule [8].

Suppose a BRB is given by $\boldsymbol{R}=\{R_1, R_2,\ldots, R_L\}$ with the k^{th} rule represented as follows [3]:

R_k: IF $\boldsymbol{U}$ is $\boldsymbol{A^k}$ THEN D with belief degrees $\boldsymbol{\beta^k}$, with a rule weight θ_k and attribute weights $\delta_1, \delta_2,\ldots, \delta_T$ (1)

where $\boldsymbol{U}$ represents the antecedent attribute vector ($U_1,\ldots, U_T$), $\boldsymbol{A^k}$ the packet antecedents $\{A_1^k,\ldots, A_T^k\}$, and A_i^k (i=1,…, T) the referential value of the antecedent attribute U_i in the k^{th} rule; T the total number of antecedent attributes used in the rule base; $\boldsymbol{D}$ the consequent vector ($D_1,\ldots, D_N$), and $\boldsymbol{\beta^k}$ the vector of the belief degrees ($\beta_{1k},\ldots, \beta_{Nk}$) for $k\in\{1,\ldots, L\}$, and β_{sk} ($s\in\{1,\ldots, N\}$ the belief degree to which D_i is believed to be the consequent if in the k^{th} packet rule the input satisfies the packet antecedents $\boldsymbol{A^k}$. θ_k is the relative weight of the k^{th} rule and $\delta_1,\ldots, \delta_T$ are the relative weights of the T antecedent attributes. L is the number of all the packet rules in the rule-base. $\sum_{s=1}^{N}\beta_{sk}\le 1$. If $\sum_{s=1}^{N}\beta_{sk}=1$, the k^{th} packet rule is said to be complete; otherwise, it is incomplete.

Take for example the belief rule in safety analysis [10-11]:

R_k: IF the **failure rate** is *frequent* and the **consequence severity** is *critical* and the **failure consequence probability** is *unlikely,* THEN the **safety estimate** is {(*good,* 0), (*average,* 0), (*fair,* 0.7), (*poor,* 0.3)}, (2)

where the consequent is a belief distribution representation for safety consequent, stating that it is 70% sure that safety level is fair and 30% sure is poor. This kind of rule reflects another kind of uncertainty caused because an expert is unable to establish a strong correlation between premise and conclusion.

In [8], to facilitate the more general application cases and more flexible and simpler rule-base generation scheme, this belief rule is extended with belief degrees embedded in the entire possible antecedent terms of each rule as well, for example, belief rule (2) can be extended as follows:

R_k: IF the **failure rate** is {(*very low,* 0), (*low,* 0), (*reasonably low,* 0), (*average,* 0), (*reasonably frequent,* 0), (*frequent,* 0), (*highly frequent,* 1) AND the **consequence severity** is {(*negligible,* 0), (*marginal ,* 0), (*moderate ,* 0.3), (*critical,* 0.7), (*catastrophic,* 0)} AND the **failure consequence probability** is {(*highly unlikely,* 0.1), (*unlikely ,* 0.7), (*reasonably unlikely,* 0.2), (*likely ,* 0), (*reasonably likely,* 0), (*highly likely,* 0), (*definite,* 0)} THEN the **safety estimate** is {(*Good*, 0), (*Average*, 0.1), (*Fair*, 0.6), (*Poor*, 0.3)} (3)

This kind of extended belief rule is able to not only capture vagueness (with linguistic terms), uncertainty (with beliefs), incompleteness (partially known beliefs in antecedents and/or consequents) and nonlinear relationships (IF-THEN rules) between these three parameters and the safety level but also provide the flexible way to incorporate the hybrid input information as well as more efficient rule generation scheme.

Generally, rule (1) can be extended in the following form:

R_k^*: IF $\boldsymbol{U}$ is $(\boldsymbol{A}, \boldsymbol{\alpha^k})$ THEN $\boldsymbol{D}$ with belief degrees $\boldsymbol{\beta^k}$, with a rule weight θ_k and attribute weights $\delta_1, \delta_2,\ldots, \delta_T$ (4)

where $(\boldsymbol{A}, \boldsymbol{\alpha^k})$ is the packet antecedents $\{\{(A_{ij}, \alpha_{ij}^k), j=1,\ldots, J_i\}; i=1,\ldots, T\}$ (as illustrated in (3)) representing antecedent with belief structure; here A_{ij} ($j\in\{1,\ldots, J_i\}$) is the jth referential value of the i^{th} antecedent attribute U_i (i=1, …, T), α_{ij}^k the likelihood to which U_i is evaluated to be the referential value A_{ij} in the kth rule with $\alpha_{ij}^k\ge 0$ and $\sum_j^{J_i}\alpha_{ij}^k\le 1$ (i=1,… T), k=1,…, L^*, L^* is the total number of all the packet rules in the rule-base. A rule base with rules in the form of (4) is called an extended belief rule base (EBRB) compared with (1). Note that all the rules in an EBRB share the same antecedent terms and consequent terms, but have different belief distributions for each antecedent attribute and the consequent. The use of the superscript k in (4) reflects this feature.

Once given an input, the activation weight w_k which measures the degree to which the k^{th} rule is weighted and activated, will be calculated using various ways depending on the nature of an antecedent attribute as detailed in [3, 8]. Having determined the activation weight of each rule in the rule base, the ER approach can be directly applied to combine the rules and generate final conclusions. The final conclusion generated by aggregating all activated rules by the current input vector $\boldsymbol{x}=\{x_i, i=1,\ldots, T\}$ can be represented as follows:

$$f(\boldsymbol{x})=\{(D_s, \beta_s), s=1,\ldots, N\}. \quad (5)$$

The final result is still estimated a belief distribution, which gives a panoramic view about estimate for a given input [3, 8]. The inference methodology based on (1) is referred to RIMER, and the one based on (4) as RIMER$^+$. The following introduced software tool includes both methodologies, for the convenience, we use only RIMER as a representative.

III. DECISION SUPPORT SYSTEM BASED ON RIMER

This section introduces a decision support software tool that implements the RIMER method.

A. Architecture

This system has been developed under the object-oriented programming paradigm. So abstract concepts can be defined and grouped into packages. The general structure of the packages follows the classic MVC architecture [14]. Each main package – Model, View and Controller – communicate with the rest through interface classes, i.e. with non-implemented methods. The main feature of this design pattern is that permits the encapsulation of each part of the system with a minimal decoupling with the rest. The MVC architecture allows us to build the system iteratively, from the first basic window with the basic features, until the final result, which is improved due to the versioning system until all the functionalities required are achieved.

B. *Features of RIMER support system*

This section introduces the main features of the system. Firstly it presents the features related to the *problem definition* and secondly related to the *solving process.*

1) Problem definition

The system provides the flexibility to define the hierarchical decision structure which is composed of several attributes. Attributes (also referred to as parameters) are variables that represent decision sub-problems. They are organized hierarchically so the attributes in higher levels of the hierarchy depend on lower-level attributes. The system presents a clear-cut interface to define the hierarchical structure and its attributes. The interface has been designed in a way that each attribute in the structure can be directly modified. Therefore, all the elements of the problem definition can be managed directly in an integrated way, just by clicking on them. These elements are subsequently described.

The system interface is based on two main windows that show the details of every parameter and keep them visible in the window to facilitate their management. Hence, an accessible and fast way to navigate between both windows is provided. Figure 1 (a) illustrates the layout of attributes management and Figure 1 (b) shows a general hierarchical structure window (also taken as the main window).

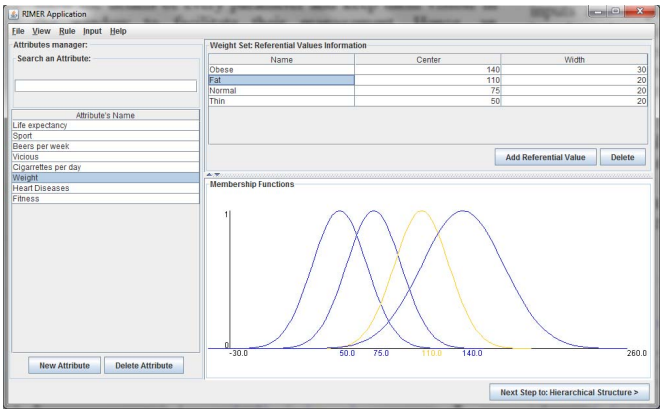

Figure 1 (a) Attribute window

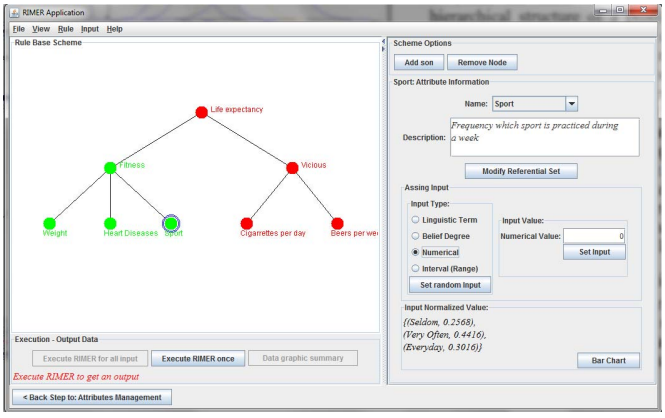

Figure 1 (b) Hierarchical structure window as main window

a) Attributes management

The attributes management window manages every aspect regarding the attributes and their referential values (a set of meaningful and distinctive evaluation standards for describing an attribute, which is commonly described by subjective linguistic terms). The definition of each attribute is essential to obtain proper results when executing RIMER, since both inputs and outputs are transformed into belief distribution based on the referential values of each attribute (see Section II). The attributes management window is divided into three parts – attributes manager (on the left), referential values manager (top-centre) and functions graph (bottom-centre).

It is possible to choose among different types of functions for the referential values when creating a new attribute in the attributes manager. Figure 2 shows examples about different functions can be graphically represented in the system– from top-left to bottom: Triangular, Trapezoidal and Gaussian fuzzy membership functions, qualitative linguistic terms and utility functions.

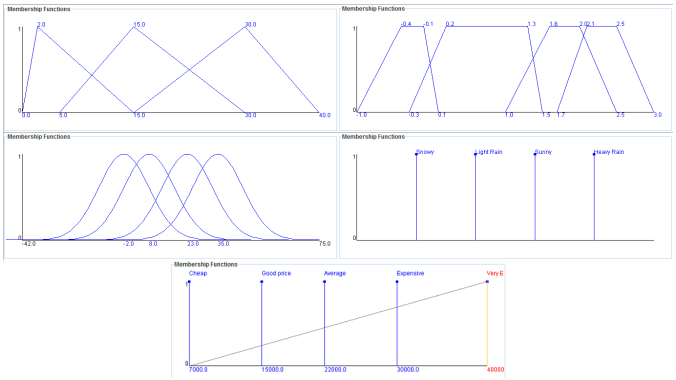

Figure 2. Possible representations in the functions graph

b) Generate hierarchal structure of the problem

From the main window, it is possible to generate a hierarchical structure of a problem by adding and removing nodes. In this case, each node of the hierarchy is associated with one of the attributes defined previously in the attributes window. By selecting a node included in the hierarchy, users will be able to modify its properties. Note that the leaf nodes have different properties from the non-leaf nodes: for a leaf node, it is possible to assign an input and for the non-leaf nodes, users can create and manage rules which form a sub-rule base in the hierarchical structure. Therefore, the content of the panel on the right side of the main window will change depending on the type of the current selected node. In Figure 1, this panel corresponds to a leaf node, since the selected node – surrounded by blue circles – is a leaf node.

c) Belief rule-bases generation and representation

Once the attributes are defined and organized into a hierarchical structure, the next step is to generate the belief rule base to model the causal relationship among the attributes. Rule based can be generated automatically from a data set, or can be manually defined and adjusted by an expert. Rules can be manually assigned in a sub rule-base by selecting one of the non-leaf nodes in the hierarchical structure. The right-side panel will shows the information related to the management of the sub rule-base correspondent to this non-leaf node. Figure 3 illustrates the appearance of the main window when a non-leaf

node (i.e. sub rule-base) is selected. Rules can be assigned automatically by selecting the option "Load Rules File" in the "Rule" menu, in the menu bar.

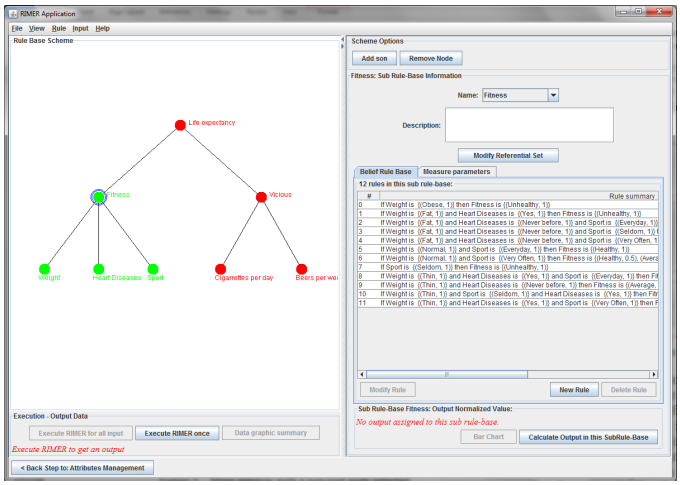

Figure 3. Main window with a non-leaf node selected

There are several options to modify the type of rules in the hierarchical structure are included. For the comparison purposes in the RIMER algorithm, the system can adopt three different types of rules:

- RIMER original rules: Rules with belief degree distributions in the antecedents of each rule. This is RIMER algorithm originally developed and executed in [3].
- RIMER^{+} extended rules: Rules with belief degree distributions not only in the antecedents of the rules, but also in each consequent. This is the extended RIMER algorithm in [8]. In the system, the rules can be automatically generated from the input-output sample data based on the simple but efficient generation proposed in [8].
- Wang-Mendel rules: Rules which do not include belief degree distributions, but linguistic terms in both antecedents and consequents. The Wang-Mendel algorithm [15] is executed when these rules are selected in the system.

In order to obtain an improved analysis of the results, the system includes data to measure the inconsistence [16] of each sub-rule base. This is an important parameter to explain the results returned by the algorithm, since it indicates the quality of the data set the system is working with.

Repeated rules are not necessary when executing the decision algorithm. Options to remove similar rules – based on a threshold – are therefore included in the system.

d) Facilate different types of inputs

Inputs can be assigned manual or automatically. To assign an input manually, click one of the leaf nodes in the hierarchical structure and select one of the options that appear in the right-side panel. Therefore, the system will ask for the required information to generate the input. Figure 4 illustrates the different types of the input format which the system could offer:

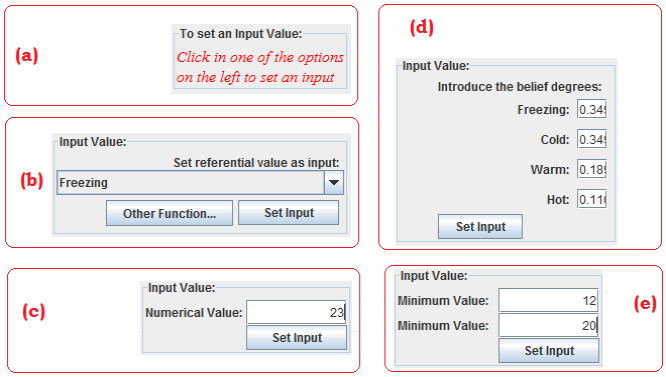

Figure 4. Different inputs that can be assigned

Fig. 4 (a) shows the initial panel, when there is no option selected; Fig. 4 (b) sets a referential value (linguistic terms) or another function that can be defined by the user as an input; Fig. 4 (c) indicates a numerical input; Fig. 4 (d) sets a belief degree distribution and Fig. 4 (e) creates an interval (range) input. Once the button "Set Input" is pressed, the input will be transformed automatically into a belief degree distribution and assigned to the current leaf node.

However, to assign inputs automatically from a properly defined input CSV file the option *Load Input File*, in the *Input* menu, included in the menu bar must be selected. In a CSV file, it is possible to specify any type of input (i.e. numerical, range, etc.). The system will automatically recognize each one of these types when loading a CSV file. Once the inputs are loaded, the system will be ready to execute the algorithm for all of them (in multi-execution mode).

Sometimes it is necessary to test a problem with random data. Thus, the system includes options to define a single random input or directly assign random values to every leaf node in the hierarchical structure.

2) Solving process

Once all the elements required for a problem definition are assigned, it is then ready to generate results by executing the RIMER approach algorithm embedded into the tool.

Depending on how the inputs have been assigned, the system could be running in different execution modes:

- If the inputs have been assigned manually, the system will be ready for single execution (with the inputs currently assigned in every leaf node). To carry out a single execution, it is just necessary to press the button so-called *Execute RIMER once* in the main window. The system will report the results of the execution through a label at the bottom part of the window. Errors during the execution process will be also specified in this label, if applicable. Users can obtain graphical representations of every input and output regarding the last single execution. The *Bar Chart* button is included in the right-side panel in the main window. It is also possible to obtain an entire graphical summary of the execution selecting the *Data Graphic Summary* included in the main window. Figure 5 shows the window where users can observe different graphical data.

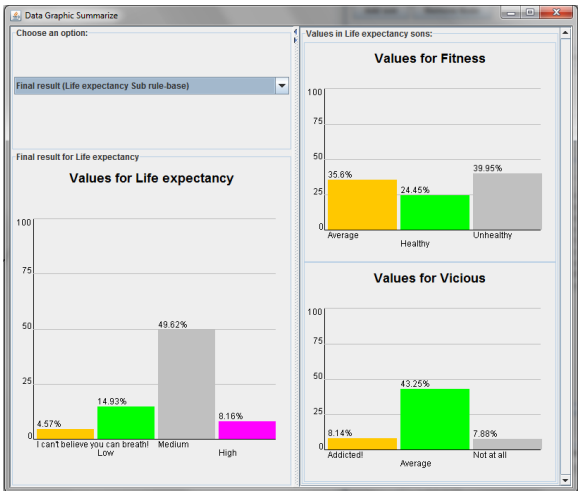

Figure 5. Data graphic summary window

- If the inputs have been loaded from a CSV file, the system will be ready for a multi-execution (one execution for each input loaded in the CSV file). For a multi-execution, the button titled "Execute RIMER for all Input" in the main window should be enabled. After this multi-execution, a log window with the results will automatically appear.

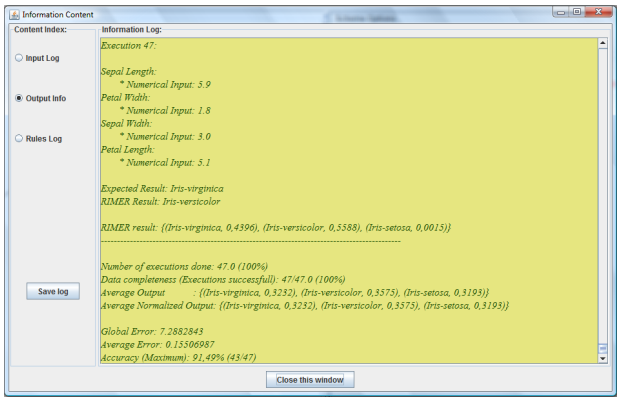

Figure 6. Statistics of the multi-execution mode for the Iris data set

Figure 6 illustrates the information log retrieved after a multi-execution process. As it can be observed, specific details are shown in this figure for the last execution (Execution 47th):

- List of original inputs used in this execution, before its transformation into belief degree distributions
- Expected result
- Result of the RIMER approach
- Result of the RIMER approach, in belief degree distribution

As aforementioned, in the multi-execution mode, general statistics about all the execution performed will be generated and showed in a log window. The statistics calculated by the system (See Figure 5) are summarized as follows:

- *Executions done*: This percentage tells us how many executions were done above the total that was supposed to be done (In case that users stop the execution, if not, this percentage always will be 100%).
- *Executions successfully done*: This percentage data acts as an indicator to show us the completeness of our rule bases. If an execution is not successfully done it is because the input data does not activate any rule. Therefore, if no rule is activated, means that a rule is not defined for this input. Thus, there is a lack of completeness in the rule-base.
- *Global and average error*: This data will show us the total and average error in RIMER predictions or classifications.
- *Accuracy*: This percentage statistic displays in how many cases RIMER predict the correct classification.
- *Average Output*: This statistic in a belief degree shape shows us which referential values of our output are more likely to be predicted. The data in the Average Normalized Output eliminates the uncertain degree and shows the average output with its values normalized.
- *T parameter*: In case that the root attribute in the hierarchical scheme has Utility functions defined within it, the system will calculate a special figure based on the distance between the output calculated and the expected one, assessing the T statistic for T=0.25, T=0.5 and T=1.

Another important feature in the system is to provide methods to load and save in external files all the information related with the application, which includes loading and saving the rules and inputs included in the current belief rule-bases and the information logs that keep information about it.

IV. Case Study

A numerical example is provided in this section to demonstrate the implementation and potential applications of the proposed system. The main focus is given on the illustration of the functionality of the developed system.

A. Problem Description

For this illustrative simulation, Iris data set for classification was chosen. Its characteristics are: 1) 150 instances (50 in each of three classes); 2) 4 attributes - Sepal length in cm, Sepal width in cm, Petal length in cm and Petal width in cm - and three possible output values - Iris Setosa, Iris Virginica and Iris Versicolor; 3) Does not contain missing values. In this example, a K-fold cross validation method by be applied, with K=5. In each test, 120 samples will be used for training and the rest 30 samples will be used for testing.

B. Implementation

Firstly, the attributes should be defined in the attributes window. In this simulation, each attribute has been assigned three linguistic terms as referential values (high, medium and low) which are defined with utility functions, so those three terms correspond the value of the maximum, average and minimum value for this attribute in the data set respectively). Figure 7 shows the attributes manager window with the definition of the five attributes for this problem (four for the input attributes and one output attribute, so called Iris).

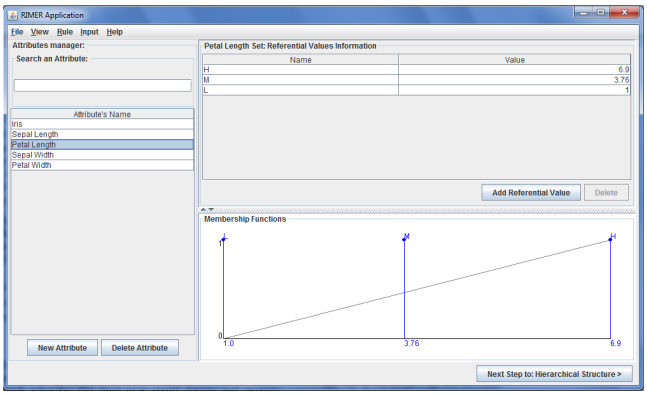

Figure 7. Attributes manager window with the Iris problem attributes

After defining all the attributes, it is needed to design the hierarchical structure for the problem. In the main window, the structure is built just by adding four son nodes to the root node, which will be the output attribute Iris.

Afterwards, the data set was downloaded and adapted into 10 CSV files, five for training and another five for testing each of the training files to apply the K-fold cross validation method. After loading one of the training files, the right-panel of the main window is updated and the rules table is filled with the rules loaded. Figure 8 shows the main window after defining the problem structure and loading one of the training files. The last element remaining to complete the definition of the problem is the input one. It is possible to assign manually inputs to the leaf nodes in the hierarchical structure. However, in this simulation one of the test CSV files that has been created before will be loaded.

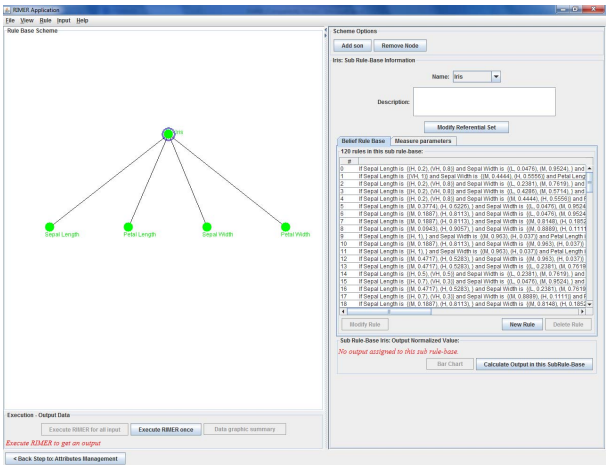

Figure 8. Main window with Iris problem structure defined

Finally, the algorithm is executed in multi-execution mode and after that the results log appears in the window, showing the results of the test executed (See Figure 6). The results for this test are 0.966 accuracy, with an average error of 0.141. For the 5 tests carried out in this simulation, the accuracy is 0.96, and the average error of 0.136.

V. CONCLUSIONS

This paper presented a decision support tool which facilitates the implementation of the recently presented RIMER approach whereby the final user could design and execute problems to obtain the decision result. The architecture and features of the system are outlined and illustrated. The system is expected to be applied for different application and we are working on further improvement about the overall presentation and functionality of the system.

ACKNOWLEDGMENT

This paper has been partially supported by the UU strategy fund, and the research projects TIN2009-08286, P08-TIC-3548 and Feder Fonds.

REFERENCES

[1] E. Turban, and J.E. Aronson, *Decision Support Systems and Intelligent Systems*, Fifth Edition, Prentice Hall, 1997.

[2] R. Sun (1995), "Robust reasoning: Integrating rule-based and similarity-based reasoning," *Artif. Intell.,* vol. 75, no. 2, pp. 241–295.

[3] J. B. Yang, J. Liu, J. Wang, H. S. Sii, and H. W. Wang, "A generic rule-base inference methodology using the evidential reasoning approach—RIMER," *IEEE Transactions on Systems, Man, and Cybernetics-Part A*: *System and Humans*, vol. 36, no. 2, pp. 266–285, Mar. 2006.

[4] J. B. Yang and D. L. Xu, "On the evidential reasoning algorithm for multiple attribute decision analysis under uncertainty," *IEEE Trans. Syst. Man Cybern., A, Syst. Humans*, vol. 32, no. 3, pp. 289–304, May 2002.

[5] J. S. Dyker, P. C. Fishburn, R. E. Steuer, J. Wallenius, and S. Zionts, "Multiple criteria decision making, multiattribute utility theory: The next ten years," *Manag. Sci.*, vol. 38, no. 5, pp. 645–654, May 1992.

[6] G. Shafer (1976), *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press.

[7] A. P. Dempster, "A generalization of Bayesian inference," *J. R. Stat. Soc., Ser. B*, vol. 30, no. 2, pp. 205–247, 1968.

[8] J. Liu, L. Martínez, A. Calzada, and Hui Wang, "An extended belief rule based inference methodology and its rule-base generation by learning from examples", submitted to *IEEE Transactions on Knowledge and Data Engineering*.

[9] J.B. Yang, J. Liu, D.L. Xu, J. Wang, H.W. Wang, "Optimization models for training belief-rule-based systems", *IEEE Transactions on Systems, Man, and Cybernetics-Part A*, 37(4)(2007) pp. 569-585.

[10] J. Liu, J.B. Yang, D. Ruan, L. Martinez, and J. Wang. Self-tuning of fuzzy belief rule bases for engineering system safety analysis, Annals of Operations Research, 163(1): 143-168, 2008.

[11] J. Liu, J.B. Yang, J. Wang, H.S. Sii and Y.M. Wang. "Fuzzy rule-based evidential reasoning approach for safety analysis", *International Journal of General Systems*, 33(2-3): 183-204, 2004.

[12] D. L. Xu, J. Liu, J. B. Yang, G. P. Liu, J. Wang, I. Jenkinson and J. Ren (2007), "Inference and learning methodology of belief-rule-based expert system for pipeline leak detection", *Expert Systems with Applications*, Volume 32, Issue 1, pages 103-113, ISSN 0957-4174.

[13] Z.J. Zhou, C.H. Hu, J.B. Yang, D.L. Xu, and D.H. Zhou, "Online updating belief-rule-based system for pipeline leak detection under expert intervention". *Expert Systems with Applications*, 36(3): 7700-7709, 2008.

[14] Reenskaug T. *Models - views – controllers*, Technical report, *Xerox PARC*, Dec. 1979.

[15] L. X. Wang, J.M. Mendel, "Generating fuzzy rules by learning from examples", *IEEE Transactions on Systems, Man, and Cybernetics*, 22:6 (1992) 1414-1427.

[16] J. Liu, L. Martínez, D. Ruan, R.M. Rodriquez, and A. Calzada (2010), "Optimization algorithm for learning consistent belief rule-base from examples", *Journal of Global Optimization*, in press: 10.1007/s10898-010-9605-x (22-09-2010).