**Abstract**—A Fuzzy Inference System Tool (FISTA) is presented. It is a shell that allows the design of fuzzy knowledge base systems from scratch. The unified modeling language (UML) offers the possibility of an easily-maintainable, shareable and continuously upgradable software system. Java™ programming language is used in the development for portability reasons, enhanced by a friendly graphical user interface (GUI). FISTA can be used as a blueprint for fuzzy knowledge base systems shell development since it describes the whole software design process from UML diagrams to testing stage. Differing from most available similar systems, FISTA offers an open-source framework easy to use and to eventually upgrade by ordinary users and experts alike, based on the knowledge they wish to process.

**Keywords**—component; Fuzzy knowledge base system shell, Fuzzy inference engine, Knowledge base, Parser, UML design.

I. INTRODUCTION

Fuzziness is the capability of a computational system to process approximate information. Fuzziness manages degrees of truth instead of the usual Boolean truth (“true or false”) offered by traditional logic. The key concept of fuzziness is that it allows a gradual and continuous transition between two binary values (0-1), rather than a crisp change between them. Fuzzy logic formalism is the basis of fuzziness and is capable of dealing with imprecision and uncertainty, based on vagueness. This approach is sometimes needed in contrast with Boolean logic, based on sharpness and sometimes unnatural and unneeded precision [1].

One main issue in the field of artificial intelligence (AI) is the design and development of algorithms and tools that allow computers to reason in the same way as human experts do when drawing conclusions from built-up knowledge and pre-programmed rules. A fuzzy knowledge base system (FKBS) is one particular type of such programs. An FKBS, or interchangeably, expert system, is a program that mimics human knowledge in order to solve problems that ordinarily require human expertise, expressed in approximate terms. A standard FKBS is designed to emulate some thinking patterns of human experts in a specific domain and allows codified human skills to be made available to non-experts. FKBS is typically composed of two main modules: a knowledge base (KB) and a fuzzy inference engine [2].

The KB itself contains two components; the rule base and the fact base.

- The rule base is where the knowledge about a particular problem domain is codified. It is usually a set of static IF-THEN production rules, obtained by eliciting human knowledge from experts or other media such as written material from encyclopedia or ad-hoc specialized references. The KB relies on linguistic variables, used to express degrees of truth in the construction of membership functions.

- The fact base is the other component of the KB. The fact base stores data at a given stage of the reasoning process. Contrary to the rules in the rule base, the facts are dynamically updated.

The second module in the FKBS is the inference engine, responsible for logically solving problems stated by the user. The inference engine applies fuzzy logic procedures using rules and facts of the KB. Eventually, explanations of solutions are also provided to the user by tracing the reasoning from initial to final stages [3].

FKBSs are applied in diversified domains. A quick search for “fuzzy knowledge base system” in the ISI Thomson© database gives 7,633 results. In [4] an KFBS is used for evaluating course difficulty and proposing proper learning strategies for students. In [5] a decision support system is offered to farmers to improve the agricultural productivity. In [6], an FKBS is used to predict earthquakes. In [7], cotton knitted fabrics are predicted using an FKBS.

Because of their pervasiveness, thoroughly designed, open, user-friendly, reusable, interoperable, and flexible software systems are essential to bridge the gap between academic settings and practical applications. Nonetheless, due to the diversity of the application domains and the large number of fuzzy methods, the development of such systems...
is both complex and time consuming. Researchers have to write \textit{ad hoc} independent and usually non-portable specialized software often developed from scratch or use a commercial product to solve a specific problem. The development based on the concentration on a specific research question does not adhere to state of the art software engineering practices and drastically reduces the chance of reusability and/or upgradability. On the other hand, when available, the developed systems are offered as black boxes. Our proposed system attempts to circumvent these issues.

In the next section, we describe the background related to the proposed system. In Section 3, we summarize the design and implementation steps followed in \textit{FISTA}. Section 4 is devoted to \textit{FISTA} testing, showing the results in its GUI. The paper ends with a conclusion summarizing the main results and designating some possible future enhancements.

### II. \textit{FISTA} Background

FKBSs are generally designed to be used by experts as knowledgeable assistants and by non-experts to improve their problem solving capabilities. Such systems could function better than any single human expert in making judgment in a specific, usually narrow area of expertise. At present, fuzzy systems are used in a wide range of industrial and scientific applications such as fuzzy control, data analysis and knowledge-based systems, among so many others as exemplified by the applications cited in the previous Section. The scope of the present work comprises the processing of fuzzy IF-THEN rules and performs a chain of inferences on complex fuzzy propositions.

#### A. Related works

The latest available fuzzy tools are the MATLAB\textsuperscript{TM} Fuzzy-Toolbox\textsuperscript{1} and FuzzyCLIPS. Unfortunately, this latter is no more maintained and the official website is down\textsuperscript{2}. Other systems such as FESS, FINEST, FPS, FLINS and FIDE\textsuperscript{TM} are now outdated (at least two decades old) as they did not stand the test of time for reasons stated above.

1) \textbf{MATLAB\textsuperscript{TM} Fuzzy Toolbox}

One of the rare widely-used systems is MATLAB\textsuperscript{TM} Fuzzy Toolbox\textsuperscript{3}. This toolbox offers a complete set of built-in functions for manipulating fuzzy systems, both for model building and for simulation. The construction of fuzzy models can be accomplished in three ways: the first being through commands, the second through a graphical user interface (GUI), and the third through block diagrams. The Fuzzy Toolbox\textsuperscript{TM} features comprises:

- Offers a Fuzzy Logic Design app for building fuzzy inference systems and viewing and analyzing results.
- Offers membership functions for creating fuzzy inference systems.
- Supports for AND, OR, and NOT logic in user-defined rules.
- Provides both Mamdani and Sugeno-type fuzzy inference methods.
- Ability to embed a fuzzy inference system (FIS) in a Simulink\textsuperscript{TM} model.
- Ability to generate embeddable C code or stand-alone executable fuzzy inference engines.

However, MATLAB\textsuperscript{TM} Fuzzy Toolbox\textsuperscript{TM} requires the MATLAB\textsuperscript{TM} platform; not free. Moreover, all functions are proprietary and therefore completely closed. The software engineering steps are not indicated in the help menu.

2) \textbf{FuzzyTECH\textsuperscript{TM} Toolbox}

FuzzyTECH\textsuperscript{3} is a modeling tool of fuzzy systems developed by INFORM GmbH. FuzzyTECH is one of the lobar leading family of software development tools for fuzzy logic and neural–fuzzy solutions. An item that differentiates the system from other FuzzyTECH are the characteristics of ergonomics and usability. Among the FuzzyTECH features we can cite:

- Allowing the creation of rules through a grid previously completed.
- Providing tools for analysis and simulation with 2D and 3D graphics resources.
- Allowing peripherals integration using RS232 serial port or TCP / IP connections.
- Generating documentation on the models of fuzzy systems implemented.
- Providing three different rule block editors, Spreadsheet Rule editor, Matrix Rule editor and FTL (fuzzy technology language) editor.

However, as for MATLAB\textsuperscript{TM} Fuzzy Toolbox, FuzzyTECH requires the \textit{ad-hoc} platform; not free. Once again, all internal functions are completely closed, for commercial reasons.

3) \textbf{FIDE\textsuperscript{TM}}

FIDE\textsuperscript{TM} (Fuzzy Inference Development Environment)\textsuperscript{4} is a fuzzy systems tool developed by Aptronix, company founded by a group of mathematicians and engineers. The FIDE\textsuperscript{TM} is a complete environment for developing a fuzzy system. It supports all phases of the development process, from the concept to the implementation. This tool has two ways of constructing models, one being through commands and the other through a GUI. Among FIDE\textsuperscript{TM} features:

- Generating fuzzy inference algorithms in Java\textsuperscript{TM} code, ANSI C code, and MATLAB\textsuperscript{TM} M-file.
- Allowing finding errors in the rules in a visual way.
- Providing the trace functionality. This feature allows the system to display all operations that are performed during the simulation in order to find and solve problems.

---

\textsuperscript{1} https://www.mathworks.com/help/fuzzy/
\textsuperscript{2} http://awesom.eu/~cygal/archives/2010/04/22/fuzzyclips_downloads/index.html
\textsuperscript{3} http://www.fuzzytech.com/
\textsuperscript{4} http://www.zaptron.com/imageproc/fide.htm
However, FIDE™ is not free and is a closed environment.

B. FISTA motivation and characteristics

FISTA is an effort to address the actual systems issues expanded above such as readability, openness, upgradability, maintainability, ease of use. We believe that FISTA will save valuable time for researchers by providing them with an open framework for further development.

1) Open-source and freeware trend

Although the fuzzy approach is still used in KBS development, there is no freely available and portable software that addresses FKBSs design. Thus, there is a pressing urge for a system that offers in a structured framework, well-known off-the-shelf algorithms for fuzzy inference. In order to contribute toward this end, our work proposes the design and development of a FKBS shell for processing approximate reasoning that can be used as a template or blueprint. It implements a powerful declarative language which allows users to express their knowledge in a natural and convenient way. On the requirements side, some similar efforts have been undertaken in FAME [8].

The available fuzzy systems software tools cited above such as MATLAB™ Fuzzy Toolbox™ are completely closed in terms of source code. As a result, the open-source software with the detailed description of the software process may be appealing to researchers, practitioners, and educators.

2) Objectives

Based on the shortcomings of actual FKBSs, the present work involves the design and development of an FKBS shell. This tool allows users to store their knowledge in a KB using a friendly graphical user interface (GUI). The tool allows the creation of knowledge bases by insertion of IF-THEN rules in the rule base and insertion of facts in the fact base. The fuzzy inference engine uses appropriate rules stored in the knowledge base and corresponding facts contained in the fact base using the Mamdani or Sugeno inference techniques to reach a certain result.

3) FISTA characteristics

FISTA offers the following characteristics:

- Software Standards: FISTA is based on state-of-art software engineering standard methods represented by the unified modeling language (UML) which plays an important role in the readability of the software, its continuous improvement and its future maintenance.

- Inference methods: Implement both Mamdani and Sugeno fuzzy inference methods are offered.

- Intuitive input GUI: Availability of a friendly GUI for writing and / or storing rules and initial facts. FISTA is GUI-supported and allows input to be easily and intuitively introduced to the system in addition to various general utilities such as help and other basic definitions.

- Readability: FISTA offers outputs that are naturally offered to the user in numerous forms such as graphical forms and texts.

- Upgradability: Users are able to add the implementation of their own algorithms. This is so, because FISTA is based on state-of-the-art software engineering techniques. Therefore FISTA is easily upgradable.

- Parser: A parser allows the indication of errors in the inputs.

- FISTA is free: no license is required to use it.

- Limitations: At present, FISTA offers a nucleus of an FKBS that can handle knowledge bases of the order of 100 rules, say. The expandability is possible since it is open-source. This task can be accomplished by the fuzzy community, if adopted.

4) Characteristics of FKBSs

TABLE 1 summarizes the characteristics of some FKBSs.

<table>
<thead>
<tr>
<th>TABLE 1 – FUZZY INFERENCE SYSTEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference Method</td>
</tr>
<tr>
<td>Interface</td>
</tr>
<tr>
<td>Source Code Export</td>
</tr>
<tr>
<td>License</td>
</tr>
<tr>
<td>Documentation</td>
</tr>
</tbody>
</table>

III. FISTA DESIGN AND IMPLEMENTATION

A. Overall architecture

1) Generic architecture

Figure 1 describes the generic architecture of an FKBS used in FISTA. Rules and facts are used by the inference engine to generate new knowledge, stored in the KB. Explanations facilities and GUI are additional tools for better use of the program by users and experts.

Figure 1 – Generic FKBS architecture
1) **Overall UML class diagram**

For interactive systems, such as *FISTA*, the object-oriented approach to the software development process is widely used. The UML is used as a modeling tool because it has become a standard for object-oriented language modeling. In the UML, a class diagram is a static structural picture of the system presenting classes, attributes, and the operations they can perform with the relationships between objects [9].

B. **Knowledge Base**

The knowledge base (KB) is a set of sentences that describe some specific domain. Each sentence is stated in a language called a knowledge representation language and describes some affirmation about the domain. As shown in Figure 1 above, the KB is composed of two modules, namely the fact base and the rule base. Facts are manipulated as data and rules represents human expertise codification. The rules are used to infer new valid knowledge to be incrementally added to the KB as facts.

1) **UML knowledge base class diagram**

Figure 2 shows the UML Class Diagram of the KB.

![Figure 2 – Knowledge base class diagram](image)

2) **Knowledge base parser**

Parsing refers to the way knowledge is handled by *FISTA* when data are introduced by the non-expert user. As far as compilation is concerned, *FISTA* offers two solutions, a command line and a Java™-based GUI.

3) **Parser grammar**

The parser obeys the grammar described below. The symbols used are the standard ones used in grammars, viz. $<\text{non terminal}>$; $[\text{optional}]$; $\dagger$ at least one time; $\ast$ 0 time or more; $\rightarrow$ derivation; $|$ or $//\text{ comments}$

```
// Grammar of the parser //
<Fuzzy-KB> → <rule>+
.rule → if <premises> then <conclusions>|IF <premises> THEN <conclusions>
<premises> → (<premise>) [and <premises>]*
<premise> → <LingVar> is <LingVal>
<LingVar> → <id>  // universal def of identifiers
<LingVal> → ([<hedge>]<Val>
<Val> → <string>
<string> → any sequence of characters
{id} → letter [letter|digits|_]*
$conclusions$ →{<conclusion> [and <conclusions>]}
$conclusion$ → <LingVar> is <LingVal>
<hedge> → very | very very | a little | extremely | more or less | slightly | somewhat | indeed
```

The parser automaton is shown in Figure 3.

![Figure 3 – Parser Automaton](image)

4) **Hedges in FISTA**

The hedges used in *FISTA* are described in Figure 4. The hedge is a modulation of a given membership function expression. The modulation is mathematically expressed as an exponent on the equation representing the original straight lines of the triangular shape. When the exponent is unity, there is no hedge; we find the unmodulated membership function $\mu_A(x)$ indicating the 2 triangle sides describing the original membership function [10].
C. Fuzzy Inference Engine

1) Fuzzy inference process steps

The process of drawing valid conclusions from existing fuzzy data is called inference, in other words deriving new sentences from previous knowledge. Fuzzy inference can be defined as a process of mapping from a given crisp input to a crisp output, using the theory of fuzzy sets and fuzzy logic [2]. The fuzzy inference engine works with linguistic sentences from previous knowledge. Fuzzy inference can infer new knowledge. Our fuzzy inference engine supports fuzzy data is called inference, in other words deriving new knowledge. Our fuzzy inference engine supports eventually including hedges. IF-THEN rules are used to infer new knowledge. Our fuzzy inference engine supports forward chaining procedure and both Mamdani and Sugeno methods. The Fuzzy inference engine follows the steps described below.

2) Generic knowledge-based agent

Because the main result of inference is to add new knowledge to the KB, there must be a way to update the KB with new valid sentences and a way to query what is known. The standard names for these tasks are TELL and ASK, respectively. Both tasks may involve inference. In logical agents, inference must obey the fundamental requirement that when one ASKS a question from the KB, the answer should be entailed from what has been told (or rather, TELLs) to the KB previously. A Generic knowledge-based agent is described below.

IV. FISTA TEST

A. Interaction with FISTA

In this section, we show how the two types of users can interact with FISTA via the interface. There are basically two types of users. The user can be an expert or non-expert; each of them seeking different queries from FISTA. The user as human expert, is responsible for building the fuzzy inference system (FIS) from personal knowledge. The responsibility of the expert is to construct the membership function, choose their ranges, their hedges along with the rules codifying the knowledge. The non-expert user task is to rely on the FIS as a black box to solve a specific problem by providing external data as facts.

1) Expert user interaction with FISTA

Using the GUI, the expert can build an FIS using two editors, the variable editor and the rule editor.

a) Variable Editor

Variable Editor: the expert can define the FIS through this window. The Variable Editor offers the following characteristics:

- Add/Delete membership functions of the input and output variables.
- Choose the Name, Range and Type of the corresponding membership function (MF) used (trimf or trapmf) and its parameters (Params).
- Allows display the shape of the MF.

b) Rule Editor

Rule editor: FISTA supports two types of rule editors that allow the expert to construct rule statements automatically. The expert can Add/Delete rules by using the following two methods:

- GUI.
- Command line supported by syntactic analyzer.

2) Non-expert user interaction with FISTA

The non-expert user can rely on the FIS to solve specific problems by entering corresponding inputs and obtaining outputs. The non-expert user does not need the introduction of variables, rules and MFs; these have been
supplied by the expert user. The non-expert user has access to the Input / Output and KB View.

B. Major screens in FISTA

1) Variable Editor

Figure 5 shows the Variable Editor. It is composed of the following items:
1: Input/Output representation
2&3: Add/Delete Input/Output
4&5: Variable Name / Variable Range
6: List of membership functions (MF)
7&8: Add/Delete MF
9&10: MF Name / MF Params (parameters)
11: MF Type: triangular / trapezoidal (trimf/trapmf)
12: Graphical representation of actual MFs.
13: Overall window
14: Toolbar indicating Problem Name/Inference Method used and Current State (whether some MF need to be introduced).

2) Rule Editor

Figures 6a-6b below show the Rule Editor. The Rule Editor is composed of the following items:
1: The File/Edit bar allows to open/create/save a rule.
2: Choose this option to Add antecedents (IF) and/or conclusions (Then).
3: Choose this option to create one of the following:
   a premise (Figure 6a) / a conclusion (Figure 6b).
4: Click on this button to add next premise or conclusion.
5: Click on this button to clear a rule.
6: Click on this button to edit/delete rules.
7. This status toolbar describing the most recent operations.

3) View Editor

This is the only Editor offered to the non-expert user. It contains inputs and corresponding results to a given query. Figure 8 shows the View Editor.

C. Example

We test the behaviour of FISTA as perceived by the expert user and the non-expert user. The variables of a decision problem are given in Table 2. The expert addresses the following issue: “How to model the risk of undertaking a project defined by some KB?”

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>project funding</td>
<td>Adequate (adqt)</td>
<td>Marginal (margl)</td>
<td>Inadequate (inad)</td>
</tr>
<tr>
<td>project staffing</td>
<td>Large (large)</td>
<td>Small (small)</td>
<td>--</td>
</tr>
<tr>
<td>Risk</td>
<td>High (high)</td>
<td>Medium (medium)</td>
<td>Low (low)</td>
</tr>
</tbody>
</table>

Based on Table 2, the expert constructs the following KB.
Rule 1: If prjct_fund is adqt then risk is low
Rule 2: If prjct_staff is small then risk is low
Rule 3: If prjct_fund is mrgl and prjct_staff is small then risk is normal
Rule 4: If prjct_fund is inad then risk is high

1) Variable Editor Contents (Expert User)
The above KB is then introduced by the expert through the Variable Editor as in Figure 6c.

![Figure 6c – Rule Editor showing the Output Plots - (For Expert User)](image)

2) View Editor Contents (Expert User)
Figure 7 shows the View Editor. The non-expert user introduces the input values (e.g. project_s = 60 people) with other inputs and obtains the output value (Risk is normal). Note that the risk value is crisp; 86.2% risk is considered normal after defuzzification.

![Figure 7 – View Editor (for non-expert user)](image)

V. CONCLUSION
We have designed and developed FISTA; a novel fuzzy inference system tool used as knowledge base system shell. We have described the significant methodological software steps needed to design and implement an FKBS from first principles. In this respect, the FISTA offers off-the-shelf fuzzy methods for solving practical problems in the same manner as MATLAB™ Fuzzy Toolbox™. The main contribution is to make FISTA a representative of an open-source, readable, upgradable FKBS. As it stands now, FISTA offers helpful assistance to both experts and users in addressing any fuzzy inference-related issues, provided that the knowledge is duly codified. The proposed shell is ready for practical use to solve diversified FKBS problems. It is suggested to use FISTA and develop it further by adding other relevant options such as machine learning methods to make it more powerful.

REFERENCES