

Monitoring and Diagnostics with Intelligent Agents Using Fuzzy Logic

Karim Ramirez, Arnulfo Alanis, Oscar Castillo, Hector Arias, Patricia Melin
Tijuana Institute of Technology, Tijuana, Mexico

Abstract. Intelligent Agents have originated a lot of discussion about what they are, and how they are different from general programs. We describe in this paper a new paradigm for intelligent agents. This paradigm helped us deal with failures in an independent and efficient way. We proposed three types of agents to treat the system in a hierarchic way. A new way to visualize fault tolerant systems (FTS) is proposed, with the incorporation of intelligent agents, as they grow and specialize create the Multi-Agent System (MAS). The MAS contains a diversified range of agents, which depending on the perspective will be specialized or evolutionary (from our initially proposal) they will be specialized for the detection and possible solution of errors that appear in an FTS). The initial structure of the agent is proposed in [1] and named reflected agent with an internal state and in the Method McCSMA [2]. The present work is based on the idea that with the help of the paradigm of intelligent agents, we may be able to handle fault tolerant systems, in the modality of embedded systems. The idea is to detect errors and to try to correct the failures that could happen in industrial control by monitoring and diagnosis.

Keywords: Intelligent Agents, Fuzzy Logic, Monitoring, Diagnostics

1 Introduction

At the present time the approach by means of agents for real applications, has worked with movable agents, which work at the level of the client-server architecture. However, in systems where the requirements are higher, as in the field of the architecture of embedded industrial systems, the idea is to innovate in this area by working with the paradigm of intelligent agents. Also, it is a good idea in embedded fault tolerant systems, where it is a new and good strategy for the detection and resolution of errors.

The main goals of the present research work were the following:

1. To create a new visualization tool of the application of the intelligent agents, in the fault tolerant systems in embedded systems.

2. To create a model, that helps the programmers to create profiles in the embedded circuits, according to utility, by means of, Intelligent Agents

3. The reflected agent with an internal state, of [3] Fig 1, sets out the general structure of the recovery Intelligent Agent for Fault tolerant Systems in Distributed Systems, whit three types of intention agents.

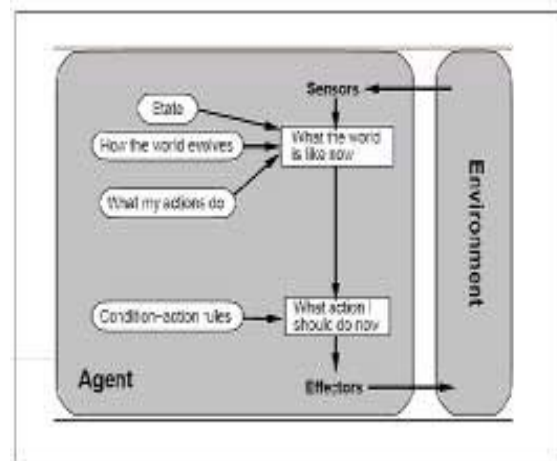


Fig 1 Agent with an internal state.

1.1 Agents

Let's first deal with the notion of intelligent agents. These are generally defined as "software entities" which assist their users and act on their behalf. Agents make your life easier, save you time, and simplify the growing complexity of the world, acting like a personal secretary, assistant, or personal advisor, who learns what you like and can anticipate what you want or need. The principle of such intelligence is practically the same of human intelligence. Through a relation of collaboration- interaction with its user, the agent is able to learn from himself, from the external world and even from other agents, and consequently act autonomously from the user, adapt itself to the multiplicity of

experiences and change its behavior according to them. The possibilities offered for humans, in a world whose complexity is growing exponentially, are enormous [1], [4], [5], [6].

2 Distributed Artificial Intelligence

Distributed Artificial Intelligence (DAI) systems can be defined as cooperative systems where a set of agents act together to solve a given problem. These agents are often heterogeneous (e.g., in Decision Support System, the interaction takes place between a human and an artificial problem solver).

Its metaphor of intelligence is based upon social behavior (as opposed to the metaphor of individual human behavior in classical AI) and its emphasis is on actions and interactions, complementing knowledge representation and inference methods in classical AI.

This approach is well suited to face and solve large and complex problems, characterized by physically distributed reasoning, knowledge and data managing. In DAI, there is no universal definition of agent, but Ferber's definition is quite appropriate for drawing a clear image of an agent: "An agent is a real or virtual entity which is emerged in an environment where it can take some actions, which is able to perceive and represent partially this environment, which is able to communicate with the other agents and which possesses an autonomous behavior that is a consequence of its observations, its knowledge and its interactions with the other agents".

DAI systems are based on different technologies like, e.g., distributed expert systems, planning systems or blackboard systems. What is now new in the DAI community is the need for methodology for helping in the development and the maintenance of DAI systems. Part of the solution relies on the use of more abstract formalisms for representing essential DAI properties (in fact, in the software engineering community, the same problem led to the definition of specification languages) [7] [8].

3 Proposed Approach

Let DS denote a distributed system made up of a set of Nodes $N = \{N_i\}$, where each N_i can be formed by several Devices (De) $[D_i, z]$. On the other hand, a DS also contains a set of Tasks to execute, $T = \{T_j\}$.

Definition 1: $N = \{N_i\}$, where i is the number of nodes of the distributed system.

Definition 2: $T = \{T_j\}$, where j is the number of tasks that are executed in the system.

Definition 3: $De = [D_i, z]$, where z is the number of devices that will be monitored by N_i . From these definitions, it can be made the following one

Definition 4: Let a distributed system DS be pair $\langle N, T \rangle$

This is where we equipped this DS with certain characteristics of failure tolerance.

This is where the use of the DAI paradigm, applied to the Fault Tolerant System (FTS) as a DS can represent a new approach with the implementation of Intelligent Agents.

IAFT = $\{AN_i, AT_j, AS\}$ will now define the Fault tolerant Agents, that work a DS.

The Node Agent (AN_i) N_i , whose mission is related to the tolerance to failures at node level (What works and what not within the node).

The Task Agent (AT_j) AT_j , whose mission is related to the tolerance to failures at task level (like recovering the tasks of the possible errors that can suffer)

System Agent (AS) DS, whose mission is the related to the tolerance to failures at the system level (what tasks must be executed in the system and on what nodes)

With it a fault tolerant DS is defined as:

Definition 5: A Distributed Fault Tolerant System DFTS is the pair $\langle DS, IAFT \rangle$, DSTF is defined as $\{DS, IAFT\}$

4 Schematic Descriptions

The scheme of the reflected agent with internal state is show below. Node Agent (AN_i): She will be that agent who is in charge to monitor the work of $[D_i, z]$ in node N_i its work will be unique and independent for each node, and its work is to activate the necessary mechanisms which can be of two types: Hw (Hardware) and SW (Software). Next we show the states in which could be, Fig 2:

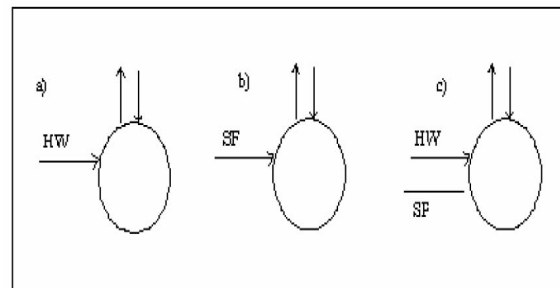


Fig 2. Node Agent (AN_i)

a) Node Agent (AN_i) with communication with hardware, in addition to the Task Agent b) Node Agent (AN_i) with communication with software, in addition to the Task Agent c) Node Agent (AN_i) with communication with software as much as with hardware, in addition to with the Task Agent

Task Agent (AT_j): She will be that agent who is in charge of monitoring and to redistribute the T in each [D_i, z], will have communication with (AN_i) and (AS), Fig 3:

Next we show its state

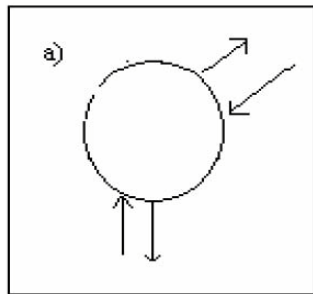


Fig 3. Task Agent (AT_j)

a) Task Agent (AT_j) with connections of communication with (AN_i) and (AS) Agent system (AS): She will be the agent who is in charge to monitor all the operation of S, having communication with (AN_i) and (AT_i), and interacting with the user.

System Agent (AS) Having communication with (AT_j), is show in Fig 4:

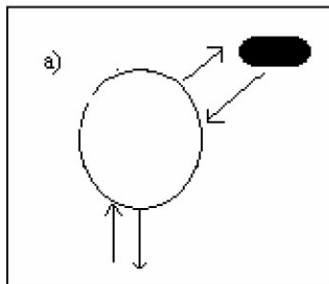


Fig 4 System Agent (AS)

We have a distributed system DS that is made up of a set of nodes N where each N_i has set of Devices D. In this DS a set of Tasks must be executed, Each node can execute a set of Tasks K.

As an example let a distributed system DS

be formed by the pair: DS = {N,T}

Where

N = {N1, N2, N3, N4, N5, N6}

T = {a, b, c, d, e, f, g, h, i, h, k}

And each node can execute:

KN1 = {a,b} KN2 = {c,f} KN3 = {a,f}

KN4 = {g,d} KN15= {g,d} KN6 = {h,k}

KN7 = {b,f} KN8= {k,a}

And for simplicity we assume each node has only one device. We assign one Node Agent to each node in N, and a single Task Node AT1 for T, and then

We have a single System Agent AS1 for the system DS. With this configuration, the AS1 coordinates all the work at application level, the Task Node AT, assigns the tasks to each node fig 5, if one node fails then the AN tries to repair it, if he can't send a warning to the AT witch in turn tries to reassign that task to other node that can execute it restoring the operation of the system. If there is no other Node that can take over the task it passes a signal to the AS so he can alert the user.

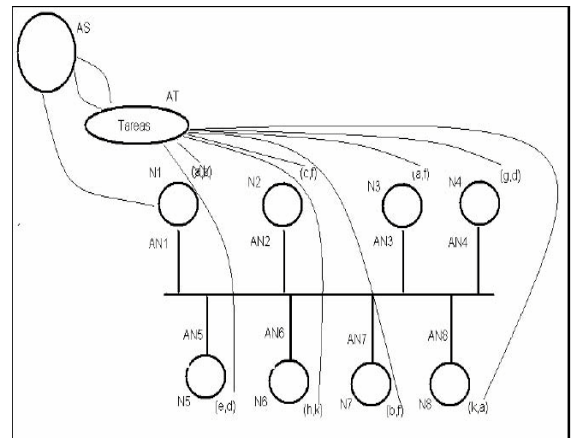


Fig. 5 Example of a DS

5 Implementation of the Approach

We have implemented the intelligent agents using the MATLAB programming language. The reason for using this programming language is that it is very easy to develop prototypes. At the beginning the implementation was done with traditional Boolean logic. In other words, the if-then rules of the agent were considered as categorical (as in an ideal situation with no uncertainty). We show in Figure 6 the implementation in MATLAB of one of the intelligent agents with categorical if-then rules. In a second phase, we consider using fuzzy logic to model uncertainty in the decision process. In this case, the knowledge base consists of fuzzy rules. We show in Figures 7 and 8 the implementation in MATLAB of the fuzzy rules of an intelligent agent.

```

% AGENTE SISTEMA

%Deteccion

    if AS.Fase.Deteccion && AN(i).S.Baja
        AS.Fase.Localizacion
    end

    if AS.Fase.Deteccion && AS.Test-
        Nodo(i).Error
        AS.Fase.Localizacion
    end

%Localizacion

    if AS.Fase.Localizacion
        AS.Fase.Aislamiento
    end

%Aislamiento

    if AS.Fase.Aislamiento
        AS.Fase.Reconfiguracion
    end

%Reconfiguracion

    if AS.Fase.Reconfiguracion &&
        AN(i).S.Baja
        AT(j).S.A-Recuperar
        AS.Fase.Recuperacion
    end

%Reconfiguracion

    if AS.Fase.Reconfiguracion &&
        AT(j).S.Recuperado
        AS.Fase.Deteccion
    end
end

```

Fig. 6 Knowledge base of the intelligent agent.

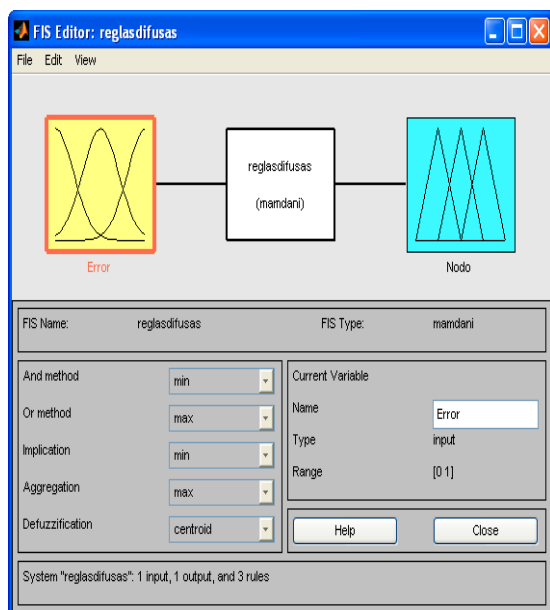


Fig. 7 Fuzzy system structure of the intelligent agent.

```

a=newfis('reglasdifusas');
a.input(1).name='error';
a.input(1).range=[0 1];
a.input(1).mf(1).name='bajo';
a.input(1).mf(1).type='trapmf';
a.input(1).mf(1).params=[-0.25 0 0.2
0.5];
a.input(1).mf(2).name='medio';
a.input(1).mf(2).type='trimf';
a.input(1).mf(2).params=[0.2 0.5 0.8];
a.input(1).mf(3).name='alto';
a.input(1).mf(3).type='trapmf';
a.input(1).mf(3).params=[0.5 0.8 1
1.12];

a.output(1).name='nodo';
a.output(1).range=[0 1];
a.output(1).mf(1).name='Baja';
a.output(1).mf(1).type='trapmf';
a.output(1).mf(1).params=[-0.25 0 0.1
0.4];
a.output(1).mf(2).name='Degradado';
a.output(1).mf(2).type='trimf';
a.output(1).mf(2).params=[0.2 0.5 0.8];
a.output(1).mf(3).name='OK';
a.output(1).mf(3).type='trapmf';
a.output(1).mf(3).params=[0.6 0.9 1
1.25];

a.rule(1).antecedent=[1];
a.rule(1).consequent=[3];
a.rule(1).weight=1;
a.rule(1).connection=2;

a.rule(2).antecedent=[2];
a.rule(2).consequent=[2];
a.rule(2).weight=1;
a.rule(2).connection=1;

a.rule(3).antecedent=[3];
a.rule(3).consequent=[1];
a.rule(3).weight=1;
a.rule(3).connection=2

fuzzy(a)

```

Fig. 8 Fuzzy rule base implementation of the agent.

6 Simulation Results

We show in this section the simulation results of the intelligent agent that was implemented in MATLAB. We show results of two different fuzzy models used in the intelligent agents, one of Mamdani form and the other in Sugeno style. First, we show the results of the Mamdani fuzzy model in Figures 9, 10, 11, 12, and 13. In Figure 9 we show the use of the rule viewer of the MATLAB fuzzy logic toolbox to test the fuzzy rules. In Figures 10 and 11 we show the membership functions used in the fuzzy rules. In Figure 12 we show the non-linear surface of the fuzzy system. In Fig. 13 we show the fuzzy rules for the error in monitoring the process.

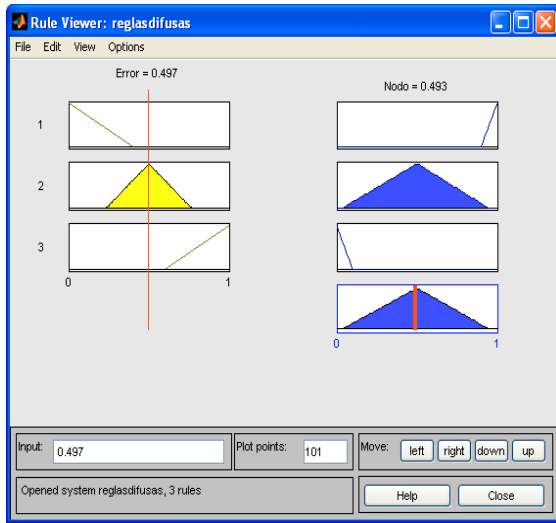


Fig. 9 Rule viewer to test the fuzzy system.

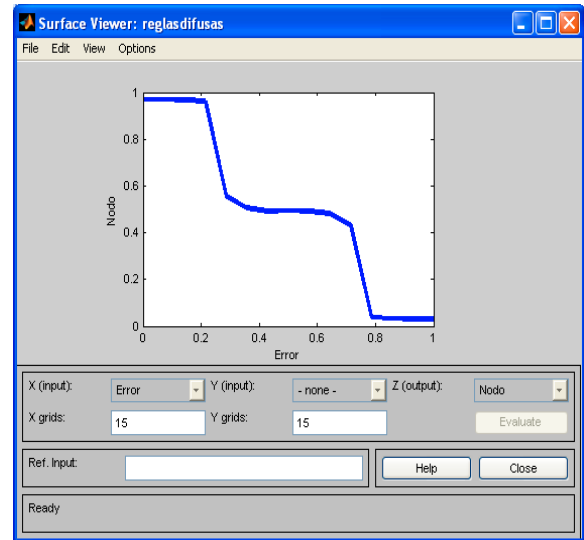


Fig. 12 Non-linear surface of the fuzzy system.

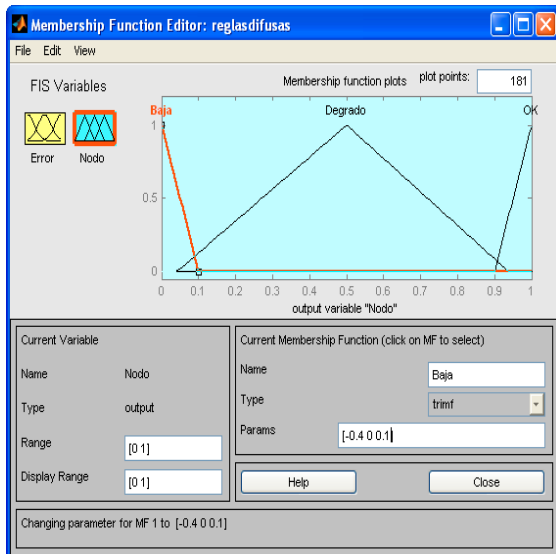


Fig. 10 Output membership functions.

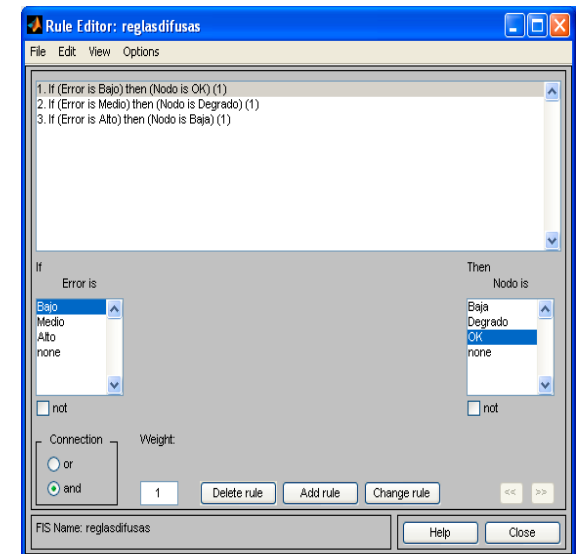


Fig. 13 Fuzzy rules of the system.

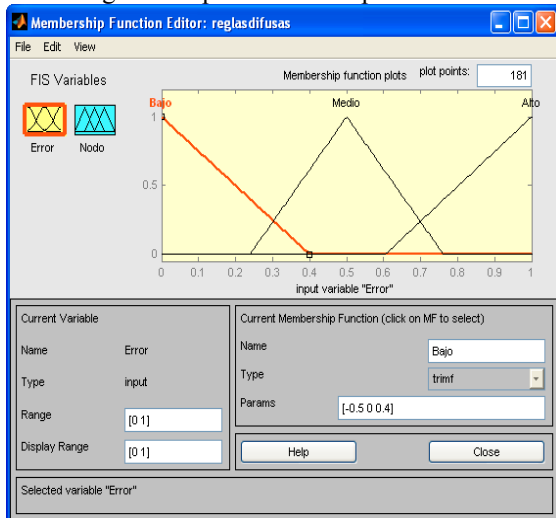


Fig. 11 Input membership functions.

Now we show the results of the Sugeno fuzzy model in Figures 14, 15, 16, 17, 18 and 19. In Figure 14 we show the structure of the Sugeno fuzzy model. In Figure 15 we show the use of the rule viewer of the MATLAB fuzzy logic toolbox to test the fuzzy rules. In Figure 16 we show the non-linear surface of the fuzzy system. In Figure 17 we show the membership functions used in the fuzzy rules. In Fig. 18 we show the constant output values of the Sugeno model. In Figure 19 we show the fuzzy rules for the error in monitoring the process. The simulation results of the Sugeno fuzzy model are better than the results of the Mamdani model and it is selected as the better option for this problem of monitoring and diagnostics.

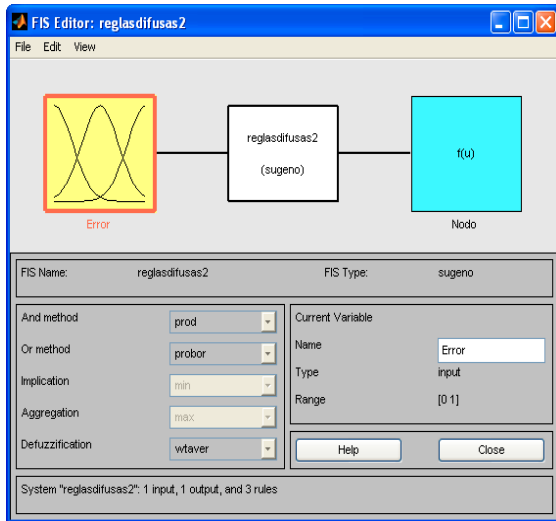


Fig. 14 General structure of the Sugeno model.

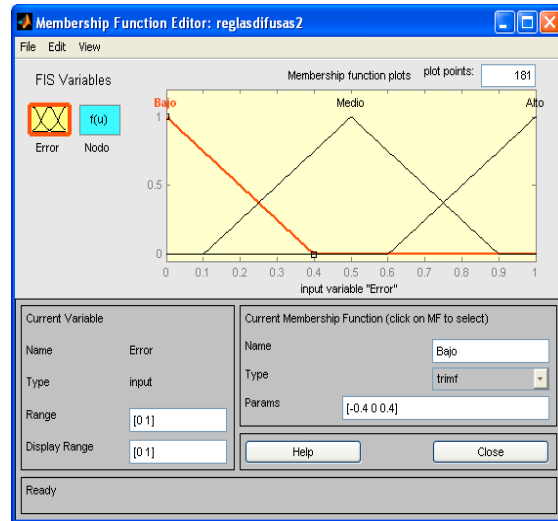


Fig. 17 Input membership functions of the system.

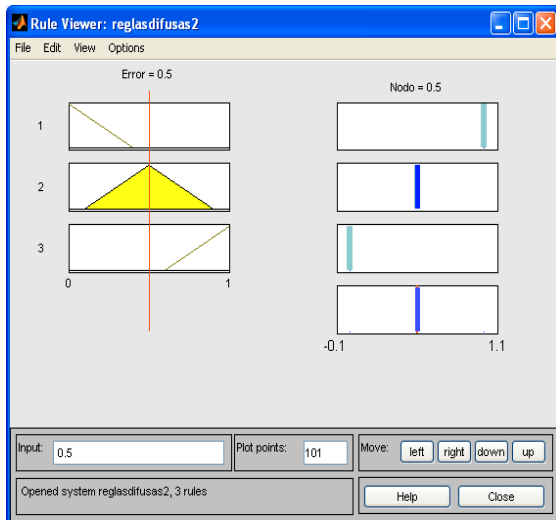


Fig. 15 Use of the rule viewer to test the system.

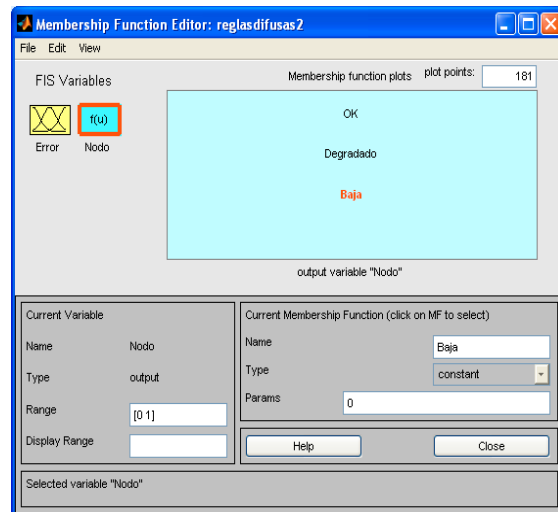


Fig. 18 Constant output values of the rules.

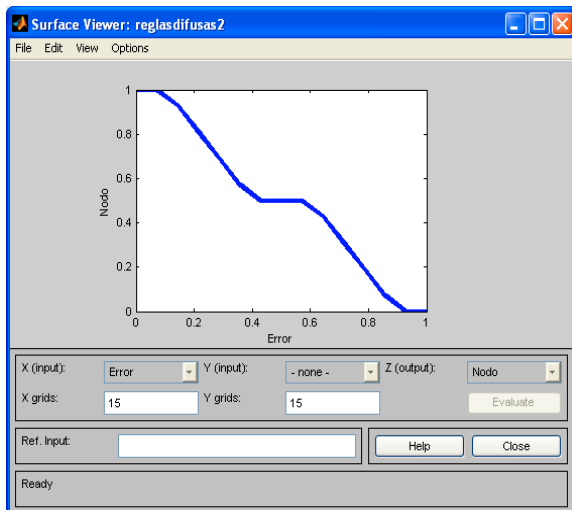


Fig. 16 Non-linear surface of the fuzzy system.

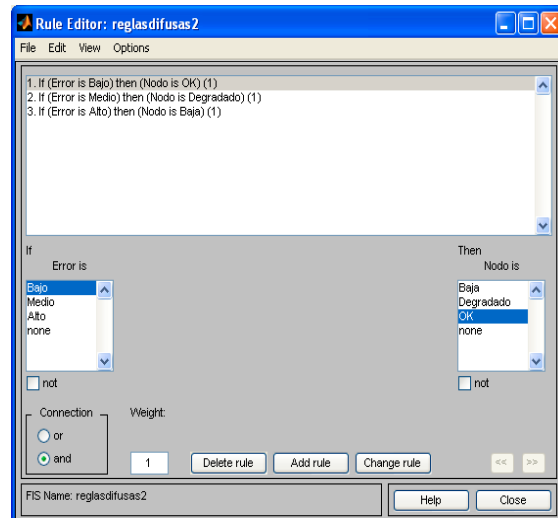


Fig. 19 Rules of the fuzzy Sugeno model.

7 Conclusions

We described in this paper our approach for building multi-agent systems for achieving fault tolerant control system in industry. The use of the paradigm of intelligent agents has enabled the profile generation of each of the possible failures in an embedded industrial system. In our approach, each of the intelligent agents is able to deal with a failure and stabilize the system in an independent way, so that the system has a behavior that is transparent for the application as well as for the user. An intelligent agent with fuzzy logic is able to monitor and make diagnosis of possible problems in the process.

Acknowledgements

The authors would like to thank DGEST for the financial support of this research project under grant 493.05-P. The student in this research project (Karim Ramirez) was also supported with a scholarship of CONACYT.

References

- [1]. Stuart Russell and Peter Norvig, Artificial Intelligence to Modern Approach, Prentice artificial Hall series in intelligence, Chapter Intelligent Agent, pages. 31-52.
- [2]. A. Alanis, Architectures for Systems Multi-Agentes, (Master Degree thesis in computer sciences), Tijuana Institute of Technology, November, 1996.
- [3]. Michael J. Woodrige, Nicholas R. Jennings. (Eds.), Intelligence Agents, Artificial Lecture Notes in 890 Subseries of Lectures Notes in Computer Science, Amsterdam, Ecai-94 Workshop on Agent Theories, Architectures, and languages, The Netherland, August 1994 Proceedings, ed. Springer-Verlag, págs. 2-21.
- [4]. P.R. Cohen ET ,An Open Agent Architecture, working Notes of the AAAI Spring symp.: Software Agent, AAAI Press, Cambridge, Mass., 1994 págs. 1-8.
- [5]. Bratko I. Prolog for Programming Artificial Intelligence, Reding, Ma. Addison-Wesley, 1986.
- [6]. Or Etzioni, N. Lesh, and R. Segal,Bulding for Softbots UNIX? (preliminary report). Tech. Report 93-09-01. Univ. of Washington, Seattle, 1993.
- [7]. Elaine Rich, Kevin Knight, Artificial intelligence, Second Edition, Ed. Mc Graw-Hill, págs. 476-478.
- [8]. Elaine Rich, Kevin Knight, Artificial intelligence, Second.Edition, Ed. Mc Graw-Hill, págs. 478-479.