Intelligent Fuzzy Controllers Laboratory

Janos L Grantner[1], Ramakrishna Gottipati[1], George A Fodor[2]

[1] Department of Electrical and Computer Engineering
Western Michigan University
Kalamazoo MI 49008-5329, USA
grantner@wmich.edu, r0gottip@wmich.edu
[2] ABB Automation Technology Products
AB, S-721 67 Vasteras, Sweden
george.a.fodor@se.abb.com

Abstract

The Intelligent Fuzzy Controllers Laboratory has been developed in the Department of Electrical and Computer Engineering of Western Michigan University with the help of a DURIP grant by the Department of Defense [6] and generous donations by ABB Automation Technology Products. This new lab is to support research, the development of advanced courses, and graduate projects in the area of intelligent controls for large, complex hybrid systems. One of the targeted applications is the supervisory level of flatness control for cold rolling mills.

Contemporary industrial control systems are designed using Object Oriented methods and software agents. It is required that the system should reach its goals even when unexpected events occur in an uncertain environment. The current research focus is on fault detection and automated recovery from faults. A fuzzy automaton has been proposed to model the system at critical junctions of the control algorithm.

A research project is currently underway to develop a Generic Encapsulated Fuzzy Automaton Software Agent for Object Oriented Control Systems. The aim of the project is twofold: 1) to develop and evaluate the performance of new methods and architectures for this area of research, and 2) to create a laboratory environment in which masters and doctoral students can work on independent projects as well as course assignments.

In the first part of the paper, we will summarize the key research ideas behind the lab. The second part of the paper will focus on creating a software environment for conducting research and supporting graduate courses in the intelligent supervisory controls area. Students will be developing solutions for plant control problems using skeleton codes provided and will be verifying the performance of their solutions with the help of the Fuzzy Automaton Software Agent and a plant simulator program. Both the Fuzzy Automaton Software Agent and the plant simulator are written in Java. These two programs are added to ABB's industrial software control system. The overall system will

have an interactive GUI interface as well as an electronic file input/output interface through the Internet.

Introduction

Among the problems that characterize industrial process control innovation, and which are not domain-related, the most difficult ones are as follows: (a) how can new knowledge be introduced into a system, (b) how can the system activate stored domain knowledge in an autonomous way, (c) how can the knowledge be validated (or otherwise detected as inappropriate) and (d) how can the system recover if the new, activated knowledge (or the currently active knowledge) is not suitable to handle the situation at hand.

The use of agent technology helps to answer question (a): in that paradigm an agent is defined as an architecture-neutral, mobile software entity that can act on behalf of a human and have decision making capabilities similar to a human [1]. The theory of software dynamical architectures describes the dynamics of the environment in which agents can act. The software architecture, by using an architecture broker, mediates the information flow among agents in order to achieve overall population-level goals, and also makes various resources (computational power, sensors, and actuators) available to the agent. The activation of the appropriate knowledge is accomplished via identification operations performed by agents that are capable of finding the right model among a set of available models [4]. Thus the answer to problem (b) is defined as the appropriate interaction among the software architecture mechanisms and the agents. Agents with model-based target seeking algorithms utilizing fuzzy logic, interacting in a distributed large system are strong candidates to replace traditional communications channels among units of a distributed system at a higher system level.

A fuzzy automaton [2] can implement new knowledge by means of the states of the goal path of an event-driven, sequential control algorithm while providing an effective approximation method to model continuous and discrete signals in a single theoretical framework. With respect to problem (c): knowledge validation is achieved by quantifying the degree of deviation from the nominal operating conditions due to unexpected events caused by either abrupt, or gradual changes in the system, or in the environment of the system. With respect to problem (d): these properties can facilitate the development of computationally inexpensive fault detection and identification (FDI) algorithms, and automated recovery from faults (subject to further research). Regarding to FDI, the evaluation of the state transitions between states of a large, complex system will be done by focusing only on clusters of relevant states along the goal path. A reconfigurable virtual fuzzy automaton will be used to model those clusters of states [3].

Software Environment for Research and Teaching

The development of a software environment for the Intelligent Fuzzy Controller Laboratory that will support research and teaching has been divided into three phases.

Development of the Software Agent GEFASA

The primary objective of Phase 1 is to create a program referred to as Generic Encapsulated Fuzzy Automaton Software Agent (GEFASA) aimed for Object Oriented Control Systems using Java. This program is then loaded as a reconfigurable agent in the FSA (Flatness Server Architecture) broker running on the Stressometer System by ABB. The Stressometer System is the key equipment in the Intelligent Fuzzy Controller Laboratory. It includes several industrial computers and PLCs (Programmable Logic Controllers) along with ABB's software development environment for complex control systems. The ABB Stressometer System and a section of the Intelligent Fuzzy Controllers Lab are as shown in Figures 1 – 2.

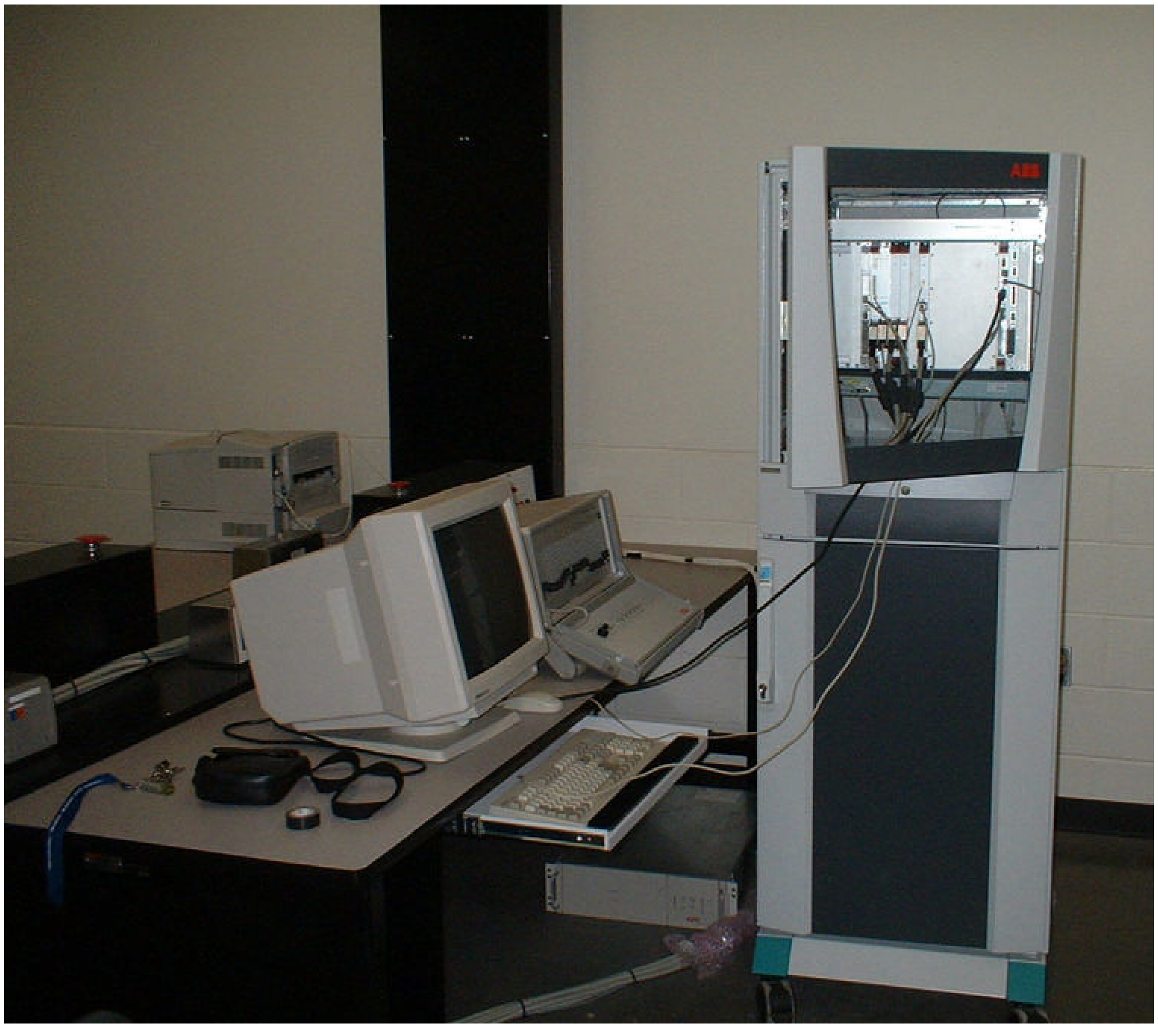


Figure 1. ABB Stressometer system in the Intelligent Fuzzy Controllers Lab

Figure 2. Intelligent Fuzzy Controllers Laboratory

The notion of fuzzy automaton in this research is based upon the premises as follows: it can stay in more than one crisp (i.e., Boolean) state at once, to a certain degree in each. Those degrees are defined by a state membership function. For each fuzzy state there is just one dominant (crisp) state, though, for which the state membership is a 1 (full membership). Each dominant state is associated with a linguistic model of the plant for inference. For each dominant state, a composite linguistic model is derived by composition of all contributing crisp states with a given strength matrix G. The transitions between fuzzy states are based upon the transitions defined between their dominant crisp states. There is an underlying Boolean finite state machine to implement the fuzzy automaton. The states of this Boolean automaton are the dominant (crisp) states of the fuzzy automaton. Fuzzy (i.e., continuous signal) inputs (and outputs) are mapped to sets of two-valued logic variables. In addition, other signal input types include two-valued ones and continuous signals with a threshold. Conditions for state transitions are defined by any combination of signals of these three types. Defuzzified (and fuzzy) outputs are obtained by computing the compositional rule of inference using the composite linguistic models. Two-valued (Boolean) outputs are devised in the usual manner [5].

GEFASA in the Flatness Server Architecture

The fuzzy automaton model implemented in the FSA broker along with an input stimuli generator is shown in Fig.3. The GEFASA model is made up of three fuzzy agents referred to as FSMSIM, FSM, and FSMPanel, respectively.

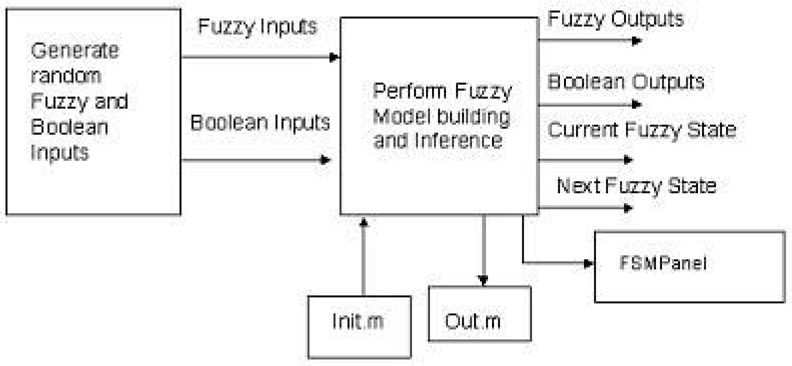


Figure 3. Functional block diagram of the GEFASA model

The FSMSIM module in the GEFASA generates random fuzzy inputs and Boolean inputs for the FSM module to perform fuzzy model building and inference operations. The block diagram outlining the FSMSIM agent in the FSA architecture is as shown in Fig. 4.

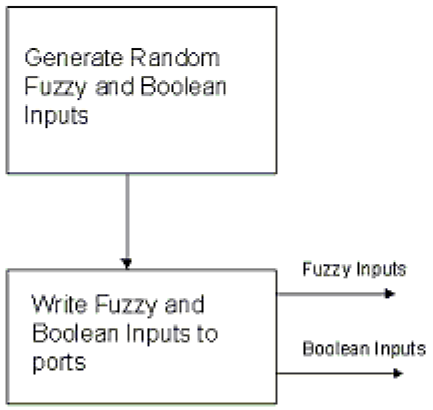

Generate Random Fuzzy and Boolean Inputs
Write Fuzzy and Boolean Inputs to ports
Fuzzy Inputs
Boolean Inputs

Figure 4. Block diagram outlining the FSM agent in the FSA broker

The FSM module generates the composite linguistic model $R^* = G^* R_s$ once for a given model and then performs the fuzzy model building and inference and the necessary state change computations. It also sends the results to the FSMPanel module for display purposes. The basic flowchart for the FSM agent is shown in Fig. 5. These three agents have been developed and loaded successfully into the FSA broker.

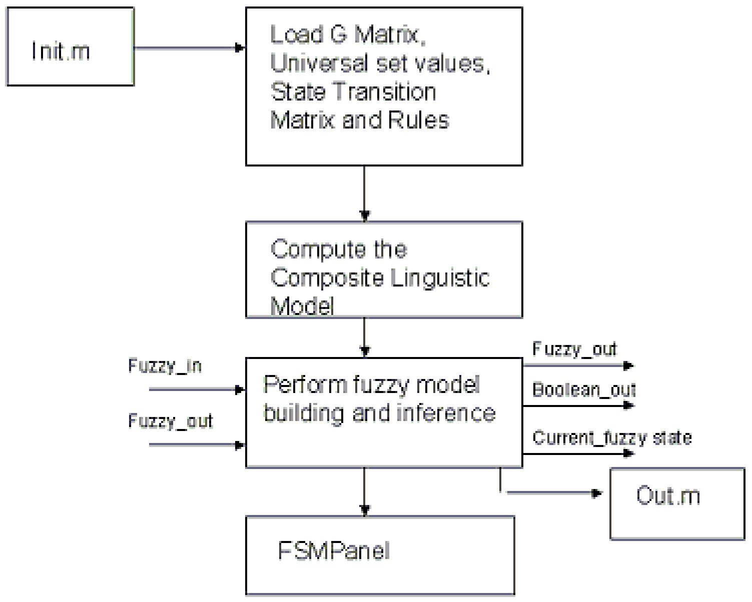

Init.m
Load G Matrix, Universal set values, State Transition Matrix and Rules
Compute the Composite Linguistic Model
Fuzzy_in
Fuzzy_out
Perform fuzzy model building and inference
Fuzzy_out
Boolean_out
Current_fuzzy state
Out.m
FSMPanel

Figure 5. Flowchart for the FSM agent

The Mill Model

Phase 2 involves the development of a Mill model by making use of the GEFASA agent created in the Phase 1. The idea is to provide Flatness Control in a closed loop using a PID controller to provide for a reference to fuzzy automaton–based control. The Mill model is shown in the Fig. 6.

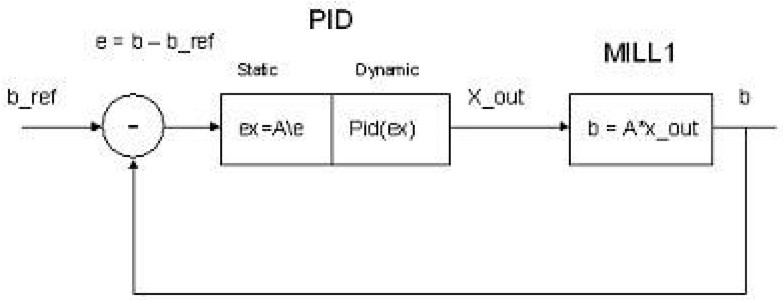


Figure 6. Mill model for Flatness Control in closed loop

Mill1 is a function that models a cold rolling mill with 3 actuators and 32 measurement zones. PID stands for an industry-standard Proportional-Integral-Derivative Controller modeled as a simple recursive function. It has two parts:

a. static computation that has an inverted model $ex = A\backslash e$ (pseudo-inverse function),
b. A dynamic section that includes a PID, or alike.

The objective is to replace the computation for the pseudo-inverse function (i.e., replace $ex = A\backslash e$ with a fuzzy approach) and leave the PID section alone. The flatness control program for the PID section is responsible for two tasks:

a. convert the error into an actuator setup, e.g., the least square of $A\backslash b$,

b. implement the PID algorithm that makes a smooth conversion between different actuator setup points.

But instead of using the least square method the program follows the following rule for converting the error into an actuator setup: the actuator that takes out more of the error is given more power to do it. The input to the algorithm is *e*, referred to as the flatness error. It is a vector of the size 32 x 1. The output from the algorithm is *ex*, of size 3x1. It is the set point for the actuators.

The algorithm has the steps as follows:

Step1.

Compute the actuator parameters as if each actuator compensated alone for the error defined as *x1*, *x2*, and *x3*, respectively.

Step2.

Compute the residue $r = e - Ax$. The actuator that works the best will have the smallest residue. Factor *A* that determines the percent of the error compensated by each actuator must be determined in this step. Let *A1*, *A2*, and *A3* be columns in matrix *A* that describes each actuator, *x1*, *x2*, *x3* be the parameters of each actuator, and the factors *k1*, *k2*, and *k3* are as follows:

k1 - part of the error compensated by Actuator1
k2 - part of the error are taken by Actuator2
k3 - part of the error are taken by Actuator3 and $K1+K2+K3 = 1$.

Then

$$A1 * x1 = k1 * b$$
$$A2 * x2 = k2 * b$$
$$A3 * x3 = k3 * b$$

Step3.

Compute factors *k1*, *k2*, *k3* as follows:

1. Compute the Euclidian norm of the residue $r = A1 * x1 - b$ which provides the error left if Actuator1 acted alone. Do the same for Actuators 2 and 3.
2. The 'best' actuator is the one with the lowest residue, but the actuator with the lowest residue must have the highest factor. That is obtained by computing $q1 = (r1+r2+r3)-r1$ and then $k1 = q1/(q1 + q2 + q3)$.

Step4.

Compute the effective actuator parameter by making use of the following rule: the actuator that can compensate more of the error will work alone until some other actuator becomes more prevalent.

Hence, the program is state-based with three states, each of which corresponds to one specific actuator working alone.

The fuzzy logic section of the program consists of the following steps:

Step1.

Define crisp states. Set up conditions for triggering a state change. Define fuzzy rules and their relationship with each crisp state

Step2.
It has three parts:

1. the initial part where the membership functions are computed and rules are given,
2. the dynamic part where the fuzzy inference is done, and
3. the interface to the program for actuator selection

Step3.
It is the implementation step:

1. Compute the flatness error norm: $||e|| = ||b\text{-}b\_ref||$ and fuzzify it.
2. Compute the inverse of the norm of the residue from each actuator (if it acted alone) $||r_i||$ $i = 1, 2, 3$ and fuzzify them.
3. Compute the parameter of each actuator $x1$, $x2$, $x3$ and fuzzify them.

This step results in 7 fuzzy inputs $||e||$, $||1/r1||$, $||1/r2||$, $||1/r3||$, $||x1||$, $||x2||$, $||x3||$
Some of these fuzzy inputs determine the states as shown in Table below.

State	Norm(e) – in(1)	Norm(1/r1) – in(2)	Norm(1/r2) – in(3)	Norm(1/r3) – in(4)
1	High	High	Any	Any
2	High	Any	High	Any
3	High	Any	Any	High
4	Low	Any	Any	Any

The state transitions are as shown in Fig. 7.

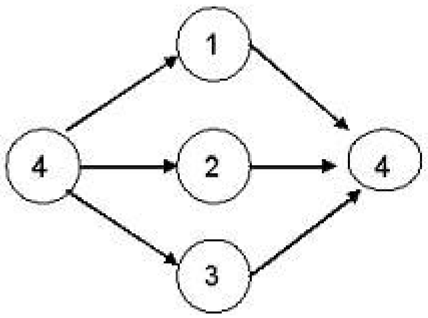


Figure 7. State Transition Diagram

Mill Model in the Flatness Server Architecture

The MillModel in the FSA architecture consists of two fuzzy agents referred to as MillFSMSIM and MillFSM. The whole model is implemented in a closed loop control in the Flatness Server Architecture as shown in Fig. 8.

The module MillFSMMSIM computes the MillModel that models the mill with 3 actuators and 32 measurement zones. The input to this module is the vector *x_out* that is obtained as a defuzzified value from the fuzzy inference process. Vector *x_out* is the command vector for some mechanical action. It is the input to the plant model that will again compute the error *e*, and the cycle will continue in this way equal to the number of cycles given by *T_MAX* (default is 100). The MillFSMSIM module consists of the following inputs and outputs:

1. input: *x_out* (3x1 matrix),
2. output: *e* (32x1 matrix), and
3. *b_ref* (32x1 matrix) that is the set point for the "reference profile" for Flatness. The mechanical process that is being controlled must achieve a result equal to this matrix. The current profile *b* is calculated and is compared with *b_ref* to determine the type of mechanical action required.

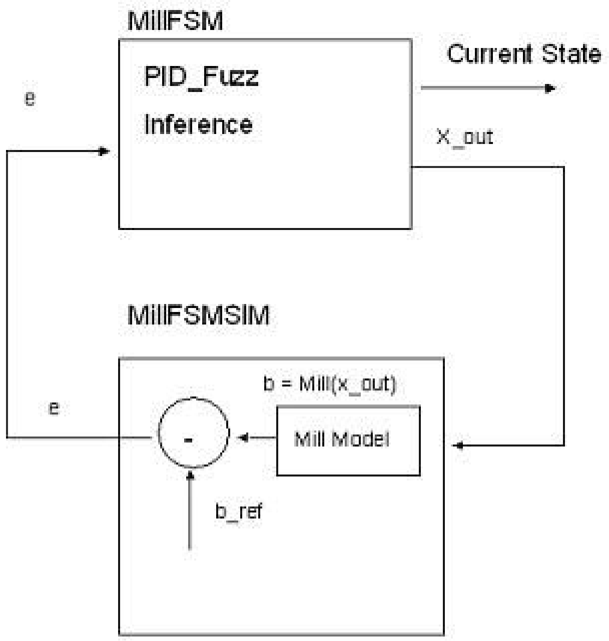


Figure 8. MillModel in the FSA architecture

The difference *e=b-b_ref* is an input to the control system implemented by the fuzzy automaton. The flowchart for the MillFSMSIM agent is shown in Fig. 9.

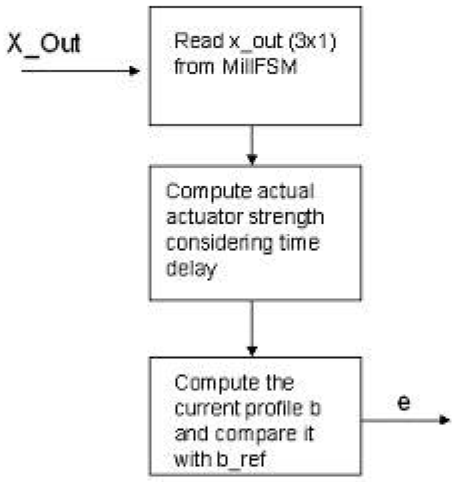

X_Out
Read x_out (3x1) from MillFSM
Compute actual actuator strength considering time delay
Compute the current profile b and compare it with b_ref
e

Figure 9. Flow chart for the MillFSMSIM agent

The MillFSM module computes the composite linguistic module $R^* = G^*R_s$ once for a model and then performs the fuzzy model building and inference computations as well as evaluates the necessary state changes. The module has one input vector *e* (32 x 1) which is the error calculated in the MillFSMSIM module, one output vector *x_out* (3x1) which is given as input to the MillFSMSIM module, and one output value *current_state* which depicts the necessary state changes. This module provides both fuzzy outputs as function of fuzzy inputs and Boolean outputs as function of Boolean inputs. The MillFSM module has the following inputs and outputs:

1. input *e* (32x1 vector), the error value computed in the MillFSMSIM module by making use of the above mentioned mill model,
2. output *x_out*(3x1 vector), obtained as a defuzzified value from the fuzzy inference process that acts as the command vector for some mechanical action.
3. output *current_state*, describes the necessary state changes.

The flowchart for the MillFSM agent is shown in Fig. 10.

Experimental System for Research

In Phase 3 the system will be connected to an ABB R&D Lab in Sweden through the Internet and will periodically receive input files that have been passed to the controllers of an actual cold rolling mill, as well as files describing the performance of the plant. This system will process the input files for the controllers using the Mill model developed in the Phase 2 and the fuzzy automaton model.

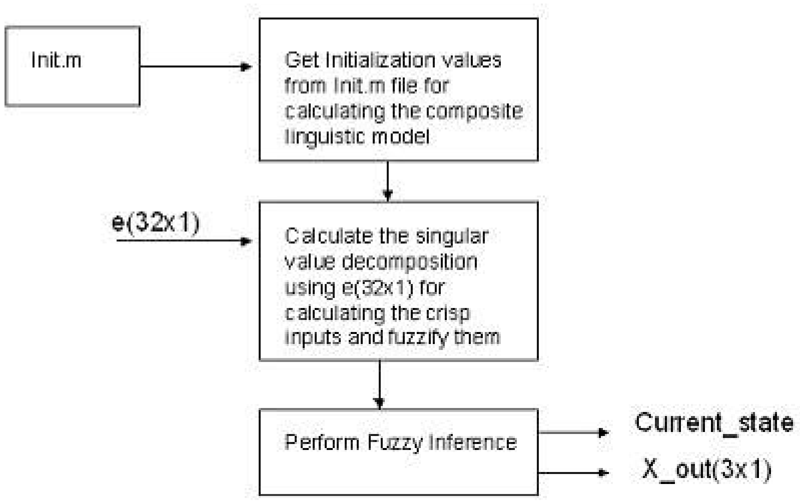

Init.m
Get Initialization values from Init.m file for calculating the composite linguistic model
e(32x1)
Calculate the singular value decomposition using e(32x1) for calculating the crisp inputs and fuzzify them
Perform Fuzzy Inference
Current_state
X_out(3x1)

Figure 10. Flowchart of the MillFSM agent

Its results will be compared with the output results produced by the controllers of the cold rolling mill to evaluate the performance of the method using the fuzzy automaton. Phase 3 will create an experimental system to conduct research on algorithms and architectures for supervisory control using fuzzy automata. The Flatness Server Architecture broker consisting of the GEFESA agent and the Mill model (under development) is as shown in Figures 11 – 15 below.

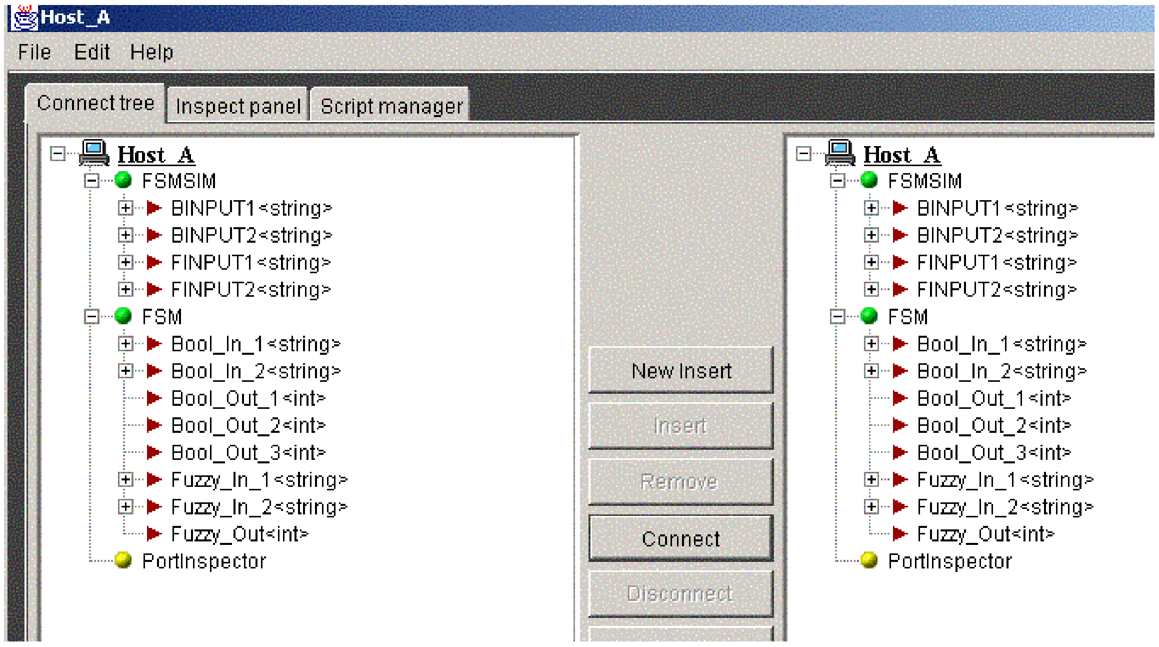
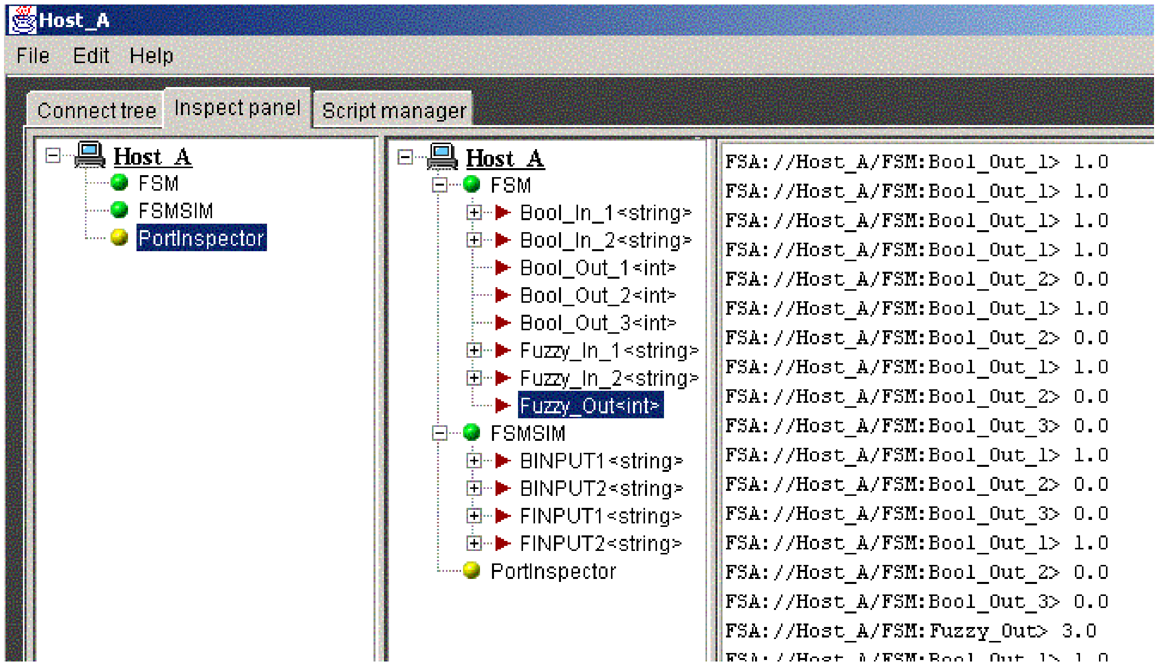
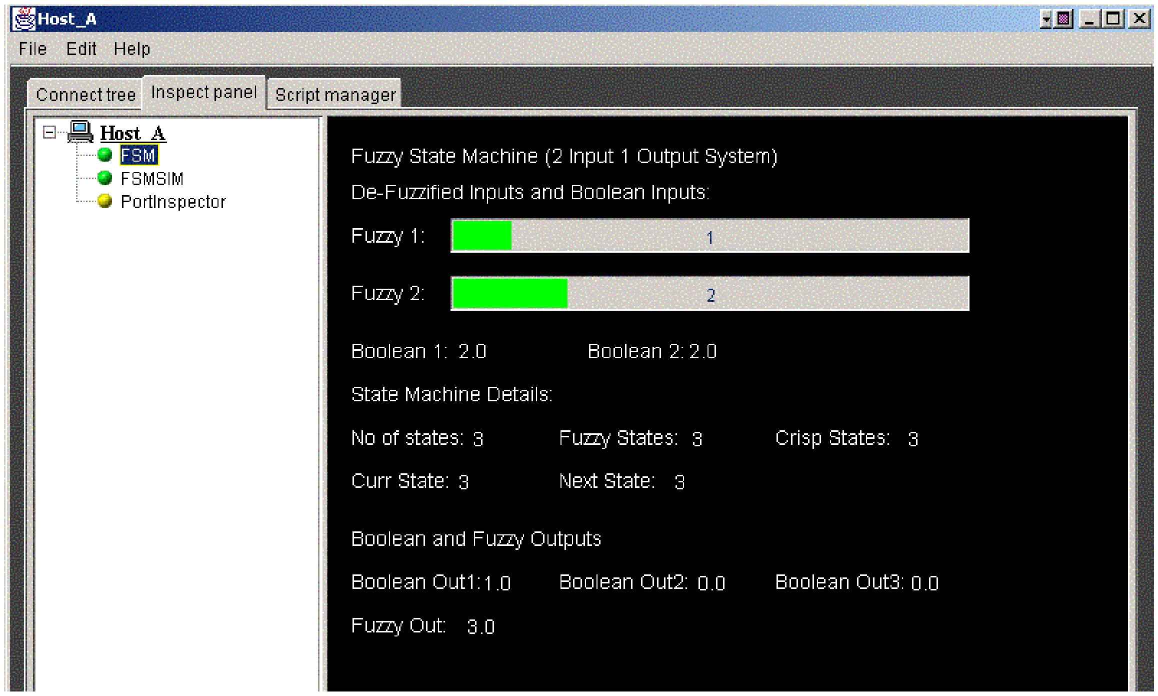


Figure 11. GEFASA in the FSA broker



```
FSA://Host_A/FSM:Bool_Out_1> 1.0
FSA://Host_A/FSM:Bool_Out_1> 1.0
FSA://Host_A/FSM:Bool_Out_1> 1.0
FSA://Host_A/FSM:Bool_Out_1> 1.0
FSA://Host_A/FSM:Bool_Out_2> 0.0
FSA://Host_A/FSM:Bool_Out_1> 1.0
FSA://Host_A/FSM:Bool_Out_2> 0.0
FSA://Host_A/FSM:Bool_Out_1> 1.0
FSA://Host_A/FSM:Bool_Out_2> 0.0
FSA://Host_A/FSM:Bool_Out_3> 0.0
FSA://Host_A/FSM:Bool_Out_1> 1.0
FSA://Host_A/FSM:Bool_Out_2> 0.0
FSA://Host_A/FSM:Bool_Out_3> 0.0
FSA://Host_A/FSM:Bool_Out_1> 1.0
FSA://Host_A/FSM:Bool_Out_2> 0.0
FSA://Host_A/FSM:Bool_Out_3> 0.0
FSA://Host_A/FSM:Fuzzy_Out> 3.0
```

Figure 12. PortInspector displaying data at the Output Ports



Fuzzy State Machine (2 Input 1 Output System)

De-Fuzzified Inputs and Boolean Inputs:

Fuzzy 1: 1

Fuzzy 2: 2

Boolean 1: 2.0 Boolean 2: 2.0

State Machine Details:

No of states: 3 Fuzzy States: 3 Crisp States: 3

Curr State: 3 Next State: 3

Boolean and Fuzzy Outputs

Boolean Out1: 1.0 Boolean Out2: 0.0 Boolean Out3: 0.0

Fuzzy Out: 3.0

Figure13. FSMpanel displaying the Outputs

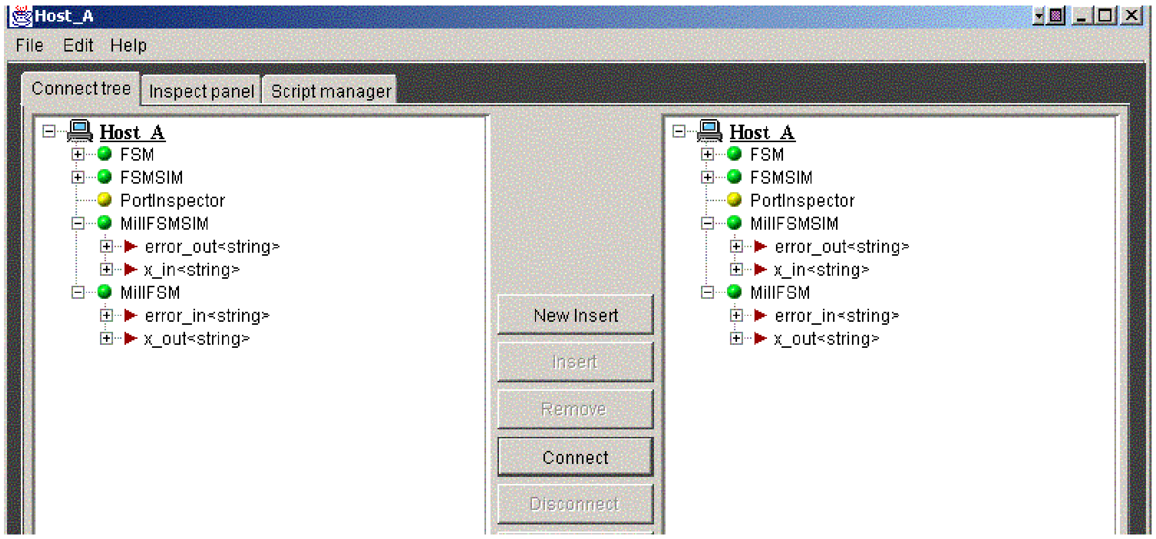
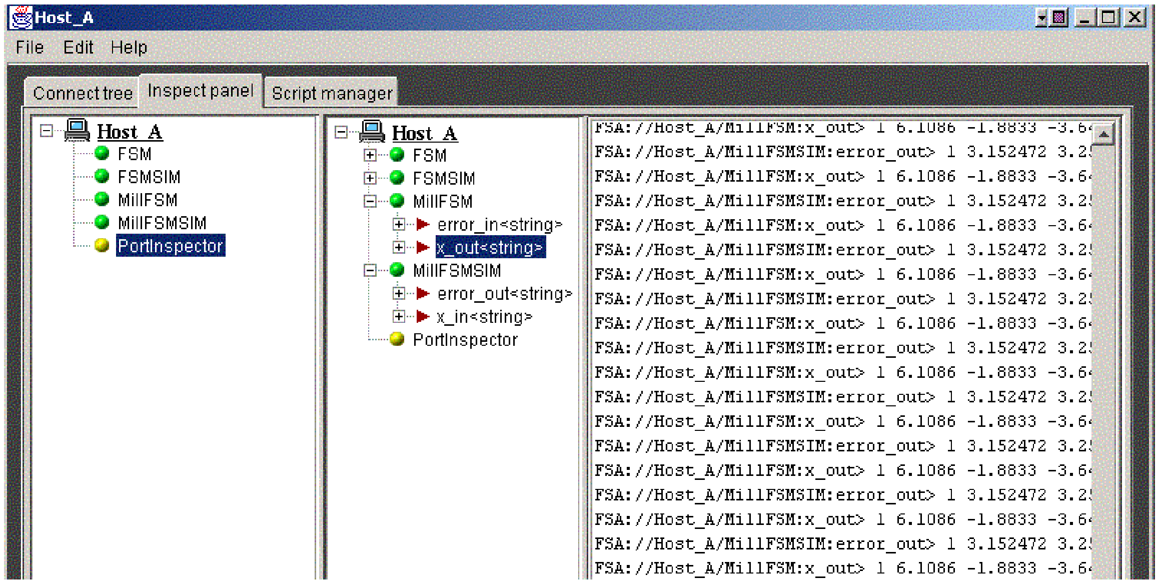


Figure14. Mill model under development in the FSA broker



```
FSA://Host_A/MillFSM:x_out> l 6.1086 -1.8833 -3.6
FSA://Host_A/MillFSMSIM:error_out> l 3.152472 3.2
FSA://Host_A/MillFSM:x_out> l 6.1086 -1.8833 -3.6
FSA://Host_A/MillFSMSIM:error_out> l 3.152472 3.2
FSA://Host_A/MillFSM:x_out> l 6.1086 -1.8833 -3.6
FSA://Host_A/MillFSMSIM:error_out> l 3.152472 3.2
FSA://Host_A/MillFSM:x_out> l 6.1086 -1.8833 -3.6
FSA://Host_A/MillFSMSIM:error_out> l 3.152472 3.2
FSA://Host_A/MillFSM:x_out> l 6.1086 -1.8833 -3.6
FSA://Host_A/MillFSMSIM:error_out> l 3.152472 3.2
FSA://Host_A/MillFSM:x_out> l 6.1086 -1.8833 -3.6
FSA://Host_A/MillFSMSIM:error_out> l 3.152472 3.2
FSA://Host_A/MillFSM:x_out> l 6.1086 -1.8833 -3.6
FSA://Host_A/MillFSMSIM:error_out> l 3.152472 3.2
FSA://Host_A/MillFSM:x_out> l 6.1086 -1.8833 -3.6
FSA://Host_A/MillFSMSIM:error_out> l 3.152472 3.2
FSA://Host_A/MillFSM:x_out> l 6.1086 -1.8833 -3.6
FSA://Host_A/MillFSMSIM:error_out> l 3.152472 3.2
FSA://Host_A/MillFSM:x_out> l 6.1086 -1.8833 -3.6
```

Figure15. PortInspector displaying data at the output ports

Course Development

The software development environment created for this laboratory will be used to support project assignments in the course ECE 672 Fuzzy Control Systems (4 cr) offered during the spring semester in the Electrical and Computer Engineering Department at

Western Michigan University. The lab consists of 6 Pentium4 machines connected to the Stressometer System through the Internet.

The Mill model (under development) will be used for class projects. Students will be provided with the necessary framework and Java code with respect to the computations for the Mill model. They will also be provided with the algorithm for the fuzzy automaton model that is to be configured for the project. Students are to configure the fuzzy automaton model for the given problem. That includes creating the linguistic models (IF THEN rules) assigned for the dominant states, setting up the state membership function (between 0 and 1), the state transition conditions, the Boolean output functions and selecting a suitable defuzzification algorithm. Then they are supposed to hand-tune them through experiments to achieve the required performance. Students will be working with a GUI interface developed in Java that produces an input file in Matlab format. The GEFASA agent and the Mill model in the FSA broker also create their output data in the form of a Matlab file. Input and output files are produced in the format of Matlab files because Matlab makes it easier for them to visualize and analyze the results and to generate reports.

Furthermore, this lab will also be used by masters and doctoral students to work on their independent research and Ph.D. dissertation projects in the area of Intelligent Control Systems.

Conclusions

The main software components of a new Intelligent Fuzzy Controllers Lab at Western Michigan University are presented. The lab is to support research work and graduate courses in the area of intelligent supervisory controls. The software architecture is based upon reconfigurable software agents. A fuzzy automaton model has been implemented to assist the decisions to be made at the supervisory level. A cold rolling mill model is under development to serve as a test-bench for research and course work. When completed the system will receive real-time plant data through the Internet to evaluate the performance of the algorithms and architectures under development.

REFERENCES

[1] M. Knapik, J. Johnson, Developing Intelligent Agents for Distributed Systems, Exploring Archtiecture, Technologies and Applications, McGraw-Hill, 1998.

[2] J.L. Grantner, G. Fodor, D. Driankov, Hybrid Fuzzy-Boolean Automata for Ontological Controllers, Proceedings of the 1998 IEEE World Congress on Computational Intelligence, FUZZ-IEEE'98, Vol. I, pp. 400-404, Anchorage, Alaska, May 4-9, 1998.

[3] J. L. Grantner, G. A. Fodor, D. Driankov, The Virtual Fuzzy State Machine Approach - A Domain-Independent Fault Detection and Recovery Method for Object-based Control Systems, 18th International Conference of the North American Fuzzy Information Processing Society - NAFIPS'99, June 10-12, 1999, New York, NY, Proceedings, pp. 158-162.

[4] MAS'99 Multi-Agent Systems in Production, Proc. of the First IFAC Workshop, Dec. 2-4, 1999, Vienna, Austria, Elsevier Science, 2000.

[5] J. L. Grantner, G. A. Fodor, Fuzzy Automaton for Intelligent Hybrid Control Systems, 2002 World Congress on Computational Intelligence, WCCI'02/FUZZ-IEEE'02, May 12-17, 2002, Hilton Hawaiian Village Hotel, Honolulu, HI, in the CD Proceedings, ISBN: 0-7803-7281-6

[6] J. L. Grantner, Intelligent Control Laboratory for Complex Hybrid System, DURIP'2001 Award (DAAD19-01-1-0431)

Biographical Information

JANOS L. GRANTNER

Janos L. Grantner is Associate Professor of Electrical and Computer Engineering. Dr. Grantner received the Ph.D. degree from the Technical University of Budapest, Hungary, in Computer Engineering, and the advanced doctoral degree Candidate of Technical Science from the Hungarian Academy of Sciences, in Computer Engineering, respectively.

GEORGE A. FODOR

George A. Fodor is Head of an R&D Department with ABB Automation Technology Products AB, Vesteras, Sweden. Dr. Fodor received the Ph.D. Degree from Linkoping University, Sweden, in Computer Science. He is Adjunct Professor at Western Michigan University, and Orebro University, Sweden, respectively.

RAMAKRISHNA GOTTIPATI

Ramakrishna Gottipati is Doctoral Student in the Department of Electrical and Computer Engineering at Western Michigan University. Mr. Gottipati received the MS degree from Western Michigan University, in Computer Engineering.