INTERFACE FOR MULTI-AGENT PLATFORM SYSTEMS

F. Ponci[(1)], A. Deshmukh [(1)], L. Cristaldi [(2)], R. Ottoboni [(2)]

[(1)] Department of Electrical Engineering - University of South Carolina
Swearingen Center – Columbia, SC 292082-0133 - USA
Phone: +1-803-7772722
Fax: +1-803-7778045
ponci@engr.sc.edu; deshmuka@engr.sc.edu

[(2)] Dipartimento di Elettrotecnica - Politecnico di Milano
Piazza Leonardo Da Vinci, 32 - 20133 Milano – Italy
Phone: 539 02 23993715
Fax: 539 02 23993703
loredana.cristaldi@polimi.it; roberto.ottoboni@polimi.it

Keywords: **Electric variables measurement, fault diagnosis, distributed computing.**

***Abstract*: an agency-agent based framework for monitoring industrial systems may comprise agents with a different tasks, (e.g. strictly measurement, computation, decision making, simulation), each implemented in a dedicated environment (e.g. LabVIEW, Matlab, custom simulators). The focus of this work is an heterogeneous agent-based monitoring system; aim of this work is to provide this complex environment with the interaction and proactiveness capabilities proper of agents. This goal is achieved by developing and setting up application program interfaces (APIs) that allow the agents to interact over a network, exchanging data, commands, requests and have each local agent react appropriately.**
In the application presented here, it will be shown how an agent-based architecture implementing APIs can be usefully employed to coordinate the roles and operations performed by the agents. In particular, it will be shown how a Control Agent, operating an induction motor drive, can interact with a Data Acquisition and Monitoring Agent, monitoring the status of the power network that supplies the drive.

1. INTRODUCTION

Multi-agent monitoring systems pose significant challenges in managing data and command exchanges and synchronizing the distributed systems. In previous papers, a possible architecture and algorithm have been presented for an advanced diagnostic and monitoring system to be applied in industrial plants mostly where electrical drives are involved. A monitoring and diagnostic platform based on agent technology has been described in [1] where the agent capabilities are not fully exploited, though. In that application in fact, the monitoring and diagnostic agents do not actively exchange information and commands but rather statically perform the same operation all the time. The agent based technology instead enables the system to be flexible and easily reconfigurable, and these characteristics are to be actively exploited in the monitoring and diagnostics environment.
In a flexible and reconfigurable system the individual agents have to be equipped with the code to handle communication with other agents along with the code to implement their own functionality. The agents in a multi-agent monitoring system though, may not be

developed in the same environment. For example, in the monitoring and diagnostics systems presented in [2] the simulation agent has been implemented within a Virtual Test Bed [3] environment, while the other agents are implemented within a LabVIEW environment. As a consequence it is necessary to develop an interface that allows agents or applications developed in different environments to communicate and work together. The APIs are suitable interfaces for the purpose. The APIs provide a way to share data with external programs or applications developed in other programming languages. It can be called a formalized set of software calls and routines that can be referenced by an application program in order to obtain services from other applications. There are various ways of developing APIs. Selecting a particular method to set up an API depends on the requirements of the users and the services offered by the applications to be interfaced. In [2] the authors have presented the advantages of using ActiveX technology over the convention approach of use of DLLs to set up APIs between applications.
In this work, the authors exploit for measurement purpose an application that builds ActiveX technology on the top of socket programming to enable users to easily configure the agents of a distributed system. ActiveX technology is a binary standard that is built on the concept of the Component Object Model. Many users require the services of external applications along with their own to perform diagnostics or simulation of complex systems. This requires them to write code for building APIs using ActiveX which is quite a complex task [4]. This application makes the implementation of ActiveX over the network transparent to the users. The code generation for the ActiveX API is automated for the user.
In this paper, the authors describe the implementation of this interface and its use to develop a multi agent system over a network using LabVIEW VIs as different agents in the distributed monitoring and computing system.

2. PREVIOUS RESULTS AND CURRENT IMPROVEMENTS

In [1] the impact of the introduction of a simulation agent, implemented in the Virtual Test Bed (VTB) as active part of an agent-based monitoring system, predominantly implemented in LabVIEW, has been analyzed. This is an example of agents with different tasks and different implementation environments.
The operations in the measurement section (data post-processing for monitoring and diagnostic purpose) are systematically performed by measurement agents according to a rigid scheme, the data are sent from the measurement agents to the simulation and management agent. The management agent receives data from the measurement agent and simulation agent and performs the visualization.
The measurement and management agents are individual LabVIEW VIs. Each of these VIs has to be manually set-up (e.g. with the destination address and port number where data are to be sent) and executed (in a specific sequence to start data acquisition and simulation computation appropriately). This system is not proactive and does not exploit the agents flexibility.
Starting from the previous observation, the development of the described system is that of creating interfaces, suitable for all the environments within which the agents are developed and capable of active exchange. In particular, the first capability that this system is provided with is that of sending requests to other agents (e.g. a command to

start data transmission to a given address), in particular in the application here introduced, between the Simulation, Control and Measurement and Manager Agents.
The interface is realized using the ActiveX technology. With the implementation of the proposed application, for example, the Manager section would be able to call and stop other agents in the system depending on its own decision (based on incoming data and not on the action of a human operator) and coordinate the flow of acquired and computed data between various agents. Also, agents other than the Manager section would be able to request and acquire data from other agents whenever required.
The interaction and proactiveness behaviors considered in this work are just examples, more sophisticated cases based on decision making capability are to be investigated in the near future.
For what concerns the software implementation, the ActiveX programming is transparent to the user in each of the applications proposed here.

3. CONCEPT OF THE APPLICATION

The proposed system in Figure 1 consists of five agents on a network, each responsible for completing a task of control, data acquisition or computation.

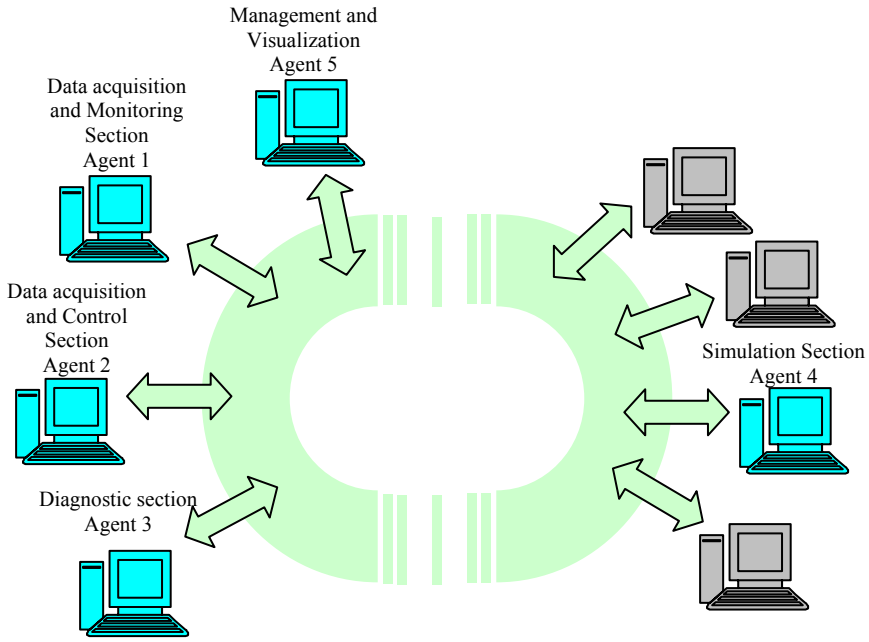

Figure 1: the scheme of the multi-agent network implemented with the interface here presented (in grey agents that are not currently interfaced yet)

The test-bench for the described agent-based monitoring system is an electrical induction motor drive, whose scheme is reported in **Figure 2**.
The agent devoted to the Drive Control (DC) performs its task through current and speed loops. In particular, the monitoring system is set-up so that this agent has to request information to the Data Acquisition and Monitoring (DAM) agent about the status of the network whenever a change in the drive operation is requested by the System Manager or by the normal operation of the control itself. The acquired data can be sent to other

agents, e.g. to the simulation agent or to the diagnostics agent, for network state evaluation.
To achieve the described system operation, DC Agent has to be able to communicate with the other agents in the same network and has to be able to perform an inference process.

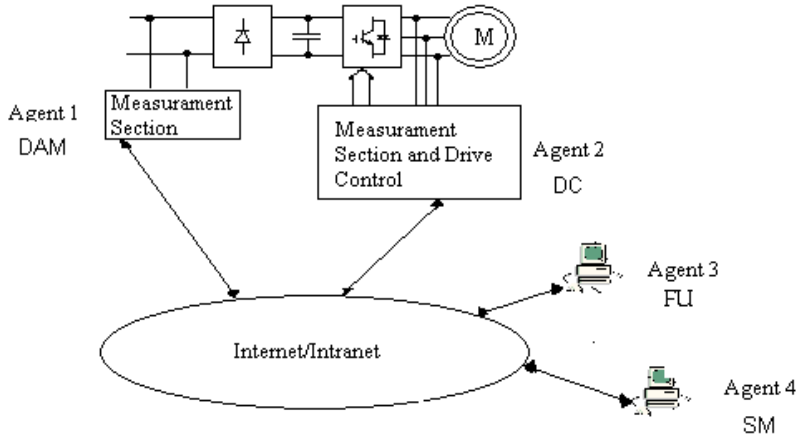

Figure 2: scheme of the monitored systems and monitoring agents

The role of the interface is to set up the connection between the agents on the network. The idea is to install the application implementing the interface on each computer carrying an agent. Each of these installed applications either poll or wait for an event on some network (TCP) port on the network. Each agent can send enable signals, data and retrieve data commands on these pre determined ports. The interface applications then react to these incoming commands by either calling the agents, making them receive or send data or by closing them.
Consider the following example, consistent with the application previously described:
The DC Agent needs to request the voltage signal at the power network side. It sends the appropriate command to a predetermined port. The interface application installed with the Measurement Agent reads these requests and appropriately uses ActiveX commands and functions to call for the voltage measurement program which in turn will start acquiring the voltage data. The DC Agent can choose to acquire these measured voltage data or/and have it sent to another agent based on its request.
In general, an intelligent multi-agent network based system is developed with the capacity of bidirectional data exchange and the ability to respond to commands.
The interface application is developed in C# language [6], [7], [8]. C# language enables easy development of windows applications with ActiveX technology built on the top of network programming. The LabVIEW VI agents can be replaced by applications from any environment that expose their services.

4. CONCLUSION AND FURTHER DEVELOPMENTS

A multi-agent, multi-environment monitoring system equipped with interaction, inference and proactive capabilities has been presented.
The proposed interfacing approach enables faster development of agents in a multi-agent system. The automatic development of the agent interface is object of the authors' future work.
From the application point of view, further development will equip the agents with stronger proactive behavior and the ability to deeply modify the monitoring agency, like for example requests to change the measurement set-up of one agent based on decisions taken by another agent and based on measurements performed in a different section of the system.

REFERENCES

[1] Cristaldi L.; Monti A.; Ponci F.; Ottoboni R.: Multi-agent based power systems monitoring platform: a prototype IEEE-Power Tech03; Pag: 528 – 532
[2] L. Cristaldi, M. Lazzaroni, A. Monti, F. Ponci "A Monitoring System Based on a Multi-Agent Platform", IEEE IMTC04, Como, Italy.
[3] T. Lovett, A. Monti, R.A. Dougal, "The new ar chitecture of the Virtual Test Bed", IEEE-COMPEL02, Mayaguez (Puerto Rico), June 2002
[4] F. Ponci, A. Monti, A. Deshmukh, "API Techniques using ActiveX for co-simulation purposes"
[5] Matlab External Interfaces, Mathworks.
[6] C++, COM and Beyond, Yashwant Kanetkar ,Sudesh Saoji.
[7] C# The Complete Reference, Herbert Schildt.
[8] Programming C#, O'Reilly.