A Multi-agent System for Integrated Control and Asset Management of Petroleum Production Facilities - Part 1: Prototype Design and Development

Atalla F. Sayda and James H. Taylor

Abstract— **This three-part paper thoroughly addresses the design and development of multi-agent system for asset management for the petroleum industry, which is crucial for profitable oil and gas facilities operations and maintenance. A research project was initiated to study the feasibility of an intelligent asset management system. Having proposed a conceptual model, architecture, and implementation plan for such a system in previous work [1], [2], [3], defined its autonomy, communications, and artificial intelligence (AI) requirements [4], [5], and initiated the preliminary design of a simple system prototype [6], we are extending the build of a system prototype and simulate it in real-time to validate its logical behavior in normal and abnormal process situations and analyze its performance.**

I. INTRODUCTION

Asset management and control of modern process plants involves many tasks of different time-scales and complexity, which makes the multi-agent system (MAS) approach a suitable framework to design a system that can manage such complex problem. Many research studies proposed different combinations of systems theoretic and artificial intelligence techniques to tackle the asset management problem, and delineated the requirements of such system [7], [8], [9]. Several research programs addressed the automation of asset management in large complex systems, namely the Pilots Associate (PA) program sponsored by the Defense Advanced Research Projects Agency (DARPA) [10], [11], the Rotorcraft Pilots Associate (RPA) program funded by the US army [12], MAGIC (Multi-Agent-Based Diagnostic Data Acquisition and Management in Complex Systems) developed by a joint venture of several European universities and companies [13], ISHM (Integrated System Health Management) project developed by NASA for space applications [14], AEGIS (Abnormal Event Guidance and Information System) developed by the Honeywell led Abnormal Situation Management (ASM) Consortium in the United States [15], and CHEM-CSS (Advanced Decision Support System for Chemical/Petrochemical Manufacturing Processes) developed by the European Community (EC) Intelligent Manufacturing Systems (IMS) consortium [16].

Among all projects, AEGIS is the most relevant. It proposes a comprehensive asset management framework from an industrial view point. AEGIS built on the experience of military aviation research projects, especially the Pilots Associate (PA) and the Rotorcraft Pilots Associate (RPA) [17]. Although the 12 year old research program has achieved several goals and developed a well established abnormal situation management awareness and culture, it did not address the automation of massive process data interpretation and process fault diagnosis and accommodation, which would be aimed to minimize the workload on process operators [18].

A new asset management research project, PAWS (Petroleum Applications of Wireless Systems), was initiated by a joint venture of Atlantic Canadian universities and the National Research Council of Canada (NRC) for oil and gas applications [1], [19], [2], [20], [21], [3], [22], [23], [24], [25], [4], [26], [5], [27], [6]. The PAWS project scope is to develop a control and information management system which consists of two subsystems. The first subsystem is a wireless sensor network which will alleviate the need for data cables in offshore oil rigs and onshore refineries, and improve flexibility for adding and reconfiguring sensors. Wireless sensors will be used where permitted by safety. The second subsystem intelligently manages the massive data flow from oil rigs and interprets it so as to help operators take more appropriate decisions during abnormal events and, through intelligent control, improve process economics. This effort is building now on the AEGIS project experiences.

As part of the PAWS project, our team is developing an *intelligent control and asset management system* (ICAM system) in which several milestones have been achieved. The conceptual model of an automated asset management system, its architecture, and its behavioral model have been defined [1], [2]. An implementation plan for such system has been prepared, and the appropriate development platforms have been chosen [3]. Furthermore, a general ICAM system agent-based structure and its communication and the artificial intelligence requirements were defined [4], [5]. A preliminary ICAM system prototype, in which few functionalities were embedded, was designed and validated [6]. This three-part paper builds on the previous work and extends the design and development of a real-time ICAM system prototype. Furthermore, a real time simulation experiment is conducted to verify the system design and validate its performance (refer to part 2 of this paper). A thorough performance analysis is also done in real time to the ICAM system prototype (refer to part 3 of this paper).

The first-part paper is organized as follows: First, we describe the ICAM system prototype in section 2. Then we discuss the design of the middleware layer, the supervisory agent layer, and the reactive agents layer of the ICAM system prototype in sections 3, 4, and 5 respectively. Finally we suggest a deployment scheme for the prototype, which

James H. Taylor is with the Department of Electrical & Computer Engineering, University of New Brunswick, PO Box 4400, Fredericton, NB CANADA E3B 5A3 jtaylor@unb.ca

Atalla F. Sayda is a PhD candidate with the Department of Electrical & Computer Engineering, University of New Brunswick, PO Box 4400, Fredericton, NB CANADA E3B 5A3 atalla.sayda@unb.ca

facilitates its validation in real time in section 6.

II. The ICAM System Prototype

A prototype has been developed in order to have the ICAM system requirements deployed in a real-world system. Figure 1 portrays the simplified ICAM system prototype. Real-time data from the external plant or a simulation model are received by the statistical data monitoring agent, which preprocesses the data by removing undesired discrepancies such as outliers and missing data. Processed data are stored in a real-time database for logging and other purposes, and is then sent to the fault detection, isolation, and accommodation (FDIA) and model ID agents for further processing. When the data statistical preprocessor detects a change in the operating point or an abnormal change in data, it alerts the model ID and FDIA agents to further identify the nature of the data change. If the change is in the process operating point, the FDIA agent asks the model ID agent to update the process model parameters. If the change is a process fault (i.e., a sensor or actuator fault), the FDIA agent detects the nature of the fault and notifies the ICAM system supervisor for further processing. If the supervisor decides that a fault can be accommodated, it notifies the FDIA agent to do so. For every event that occurs, the supervisor is notified, which in turn monitors and assesses the logical behavior of the system. Processed data at every agent are sent to an operator interface, which allows operators make the appropriate decision depending on the plant situation.

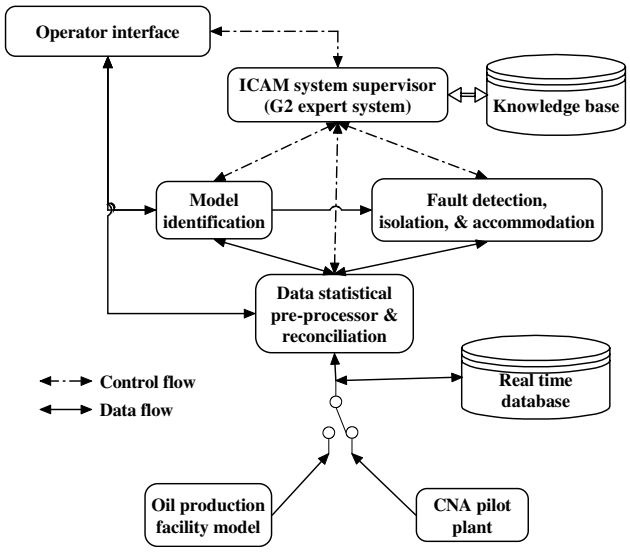

Fig. 1. ICAM system prototype

The ICAM system prototype design is a complex task, in which the following system layers are developed:

- the middleware layer which provides data communication between reactive agents and event communications with the supervisory agent,
- the artificial intelligence (AI) layer (i.e., the supervisory agent) which coordinates the behavior of the reactive agents to achieve robust performance against the external environment dynamic changes, and
- the reactive agents layer, which represent the different system computational and data processing functionalities (e.g., fault detection and isolation, process modeling, etc...). Reactive agents are implemented as MATLAB scripts for ease of development and design.

The design of the different layers and agents of the ICAM system prototype is discussed in the following sections.

III. The Middleware Layer Design

The remote memory access (RMA) communication approach, which is part of the message passing interface (MPI) communication library, was chosen to address data communications between the reactive agents [4], [5]. The RMA protocol separates the communication of data from sender to receiver from the synchronization of sender with receiver, divides into two categories. The first one is active target communication, where data are moved from the memory of one process to the memory of another, and both are explicitly involved in the communication. This communication pattern is similar to message passing, except that all the data transfer arguments are provided by one process, and the second process only participates in the synchronization. The second category is passive target communication, where data are moved from the memory of one process to the memory of another, and only the origin process is explicitly involved in the transfer. This communication paradigm is closest to a shared memory model, where shared data can be accessed by all processes, irrespective of location [28].

In active target communication, a target window can be accessed by RMA operations only within an exposure epoch. Such an epoch is started and completed by RMA synchronization calls executed by the target process. Distinct exposure epochs at a process on the same memory window must be disjoint, but such an exposure epoch may overlap with exposure epochs on other windows or with access epochs for the same or other windows arguments [28]. There is a one-to-one matching between access epochs at origin processes and exposure epochs on target processes. RMA operations issued by an origin process for a target window will access that target window during the same exposure epoch if and only if they were issued during the same access epoch. In passive target communication the target process does not execute RMA synchronization calls, and there is no concept of an exposure epoch. We have chosen the active target communication RMA communication type to achieve high reliability.

Four RMA data communication channels are designed to transfer raw data, processed data, fault accommodation parameters and plant state space model to the corresponding agents respectively. Figure 2 illustrates the general synchronization pattern for the second MPI data channel. Once data have been preprocessed by the statistical agent, it is converted to the MPI type to prepare it for communication on the second MPI channel. An access epoch is started at the statistical processing agent to synchronize with the model ID and FDIA agents. If the model ID and FDIA agents have started their exposure epochs, then the processed data message is transferred to their memory windows. Once the communication on this MPI channel is completed at

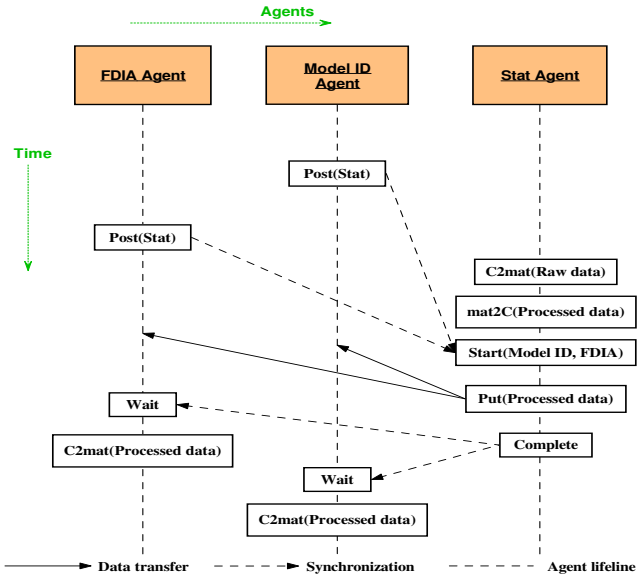

Fig. 2. ICAM system prototype MPI communications sequence for the second MPI data channel

the statistical processing agent, the exposure epochs at the target agents are finalized, and the processed data message is converted to the MATLAB type again for further processing. The other MPI data channels follow the same logic, in which the data conversion and transfer tasks are the same.

When it comes to the communications between the reactive agents and the supervisory agent, the remote procedure call (RPC) paradigm is used to achieve looser connection with supervisory agent. RPC is a client/server infrastructure that increases the interoperability, portability, and flexibility of an application by allowing the application to be distributed over multiple heterogeneous platforms. It reduces the complexity of developing applications that span multiple operating systems and network protocols by insulating the application developer from the details of the various operating system and network interfaces.

The RPC communication part of the ICAM system prototype was designed so that the G2 supervisory agent acts as a client for the reactive agents (i.e., servers). Every reactive agent has an update procedure, which can be called remotely by the supervisory agent. So when the G2 supervisory agent wants to update the state of each agent in its knowledge base, it sends a request to the specified reactive agent and passes a structure object to the reactive agent. The structure object represents the reactive agent state, which contains attributes about the reactive agent's MPI and G2 communication channels, and its internal decisions and response to the external environment.

IV. The Supervisory Agent Design

The supervisory agent is implemented based on the G2 real time expert system shell, which codifies the ICAM system internal and external behavior in its knowledge base [29]. The supervisory agent contains an ontology that represents the different agents of the ICAM system prototype. Each agent in the ontology has its own attributes which represent the agent technical characteristics and methods which represent the agent's behavior. The ICAM system prototype is represented as a logical connection of the corresponding reactive agents, as illustrated in figure 4. Each reactive agent has two connections; the first is the data MPI connection, and the second is the G2 connection with the supervisory agent. The META Agent object, which encapsulates the supervisory knowledge base of the ICAM system prototype, is a representation of the supervisory agent. The ENV Agent object represents the pilot plant model agent, the STAT Agent object represents the statistical preprocessing agent. Likewise, the M-ID Agent and PSV Agent objects represent the model identification agent and the FDIA agent, which exploits the parity space vector (PSV) FDI approach [30], [19], [23], [31], [26]. The attributes of each reactive agent represent the MPI and G2 connections characteristics, the reactive agent internal state, and its response to changes in the pilot plant (i.e., the external environment).

The attributes are updated by means of four rules which fire asynchronously every one second, after which a structure object is sent to the actual reactive agent to update it. The structure object acts as a two-way vehicle, which has the decisions from the supervisory agent and the internal state of the reactive agent. Once the attributes are updated, the supervisory agent reasons about the newly updated values and generates new decisions depending on the current internal state of the reactive agent and its response to the pilot plant current dynamic situations. The master rule base, which manages the ICAM system prototype, is embedded in the META Agent object.

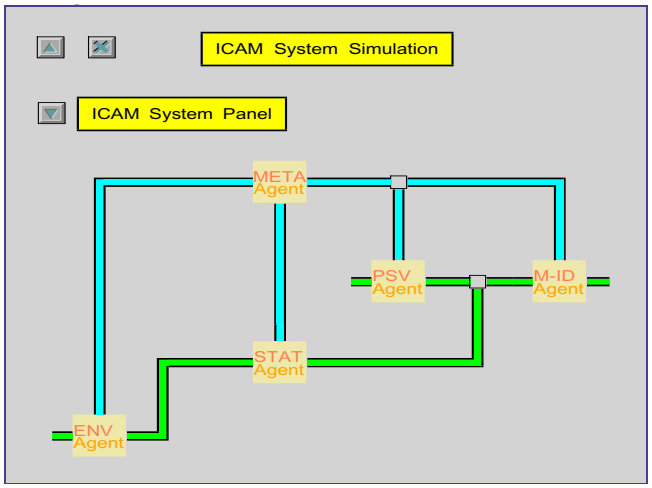

Fig. 4. ICAM system prototype representation in the G2 supervisory agent

Since the supervisory agent of the ICAM system coordinates its internal and external behavior, it is crucial to carefully design the rule-base of the supervisory agent to achieve robust system performance. The rule-base codifies the desired system behavior in response to external environment dynamic changes and to process operator interactions. Figure 3 illustrates the ICAM system prototype event sequence diagram, which is embedded in the supervisory agent rule-base. The rule-base design process is in its preliminary stage; it will be further developed to address more complex

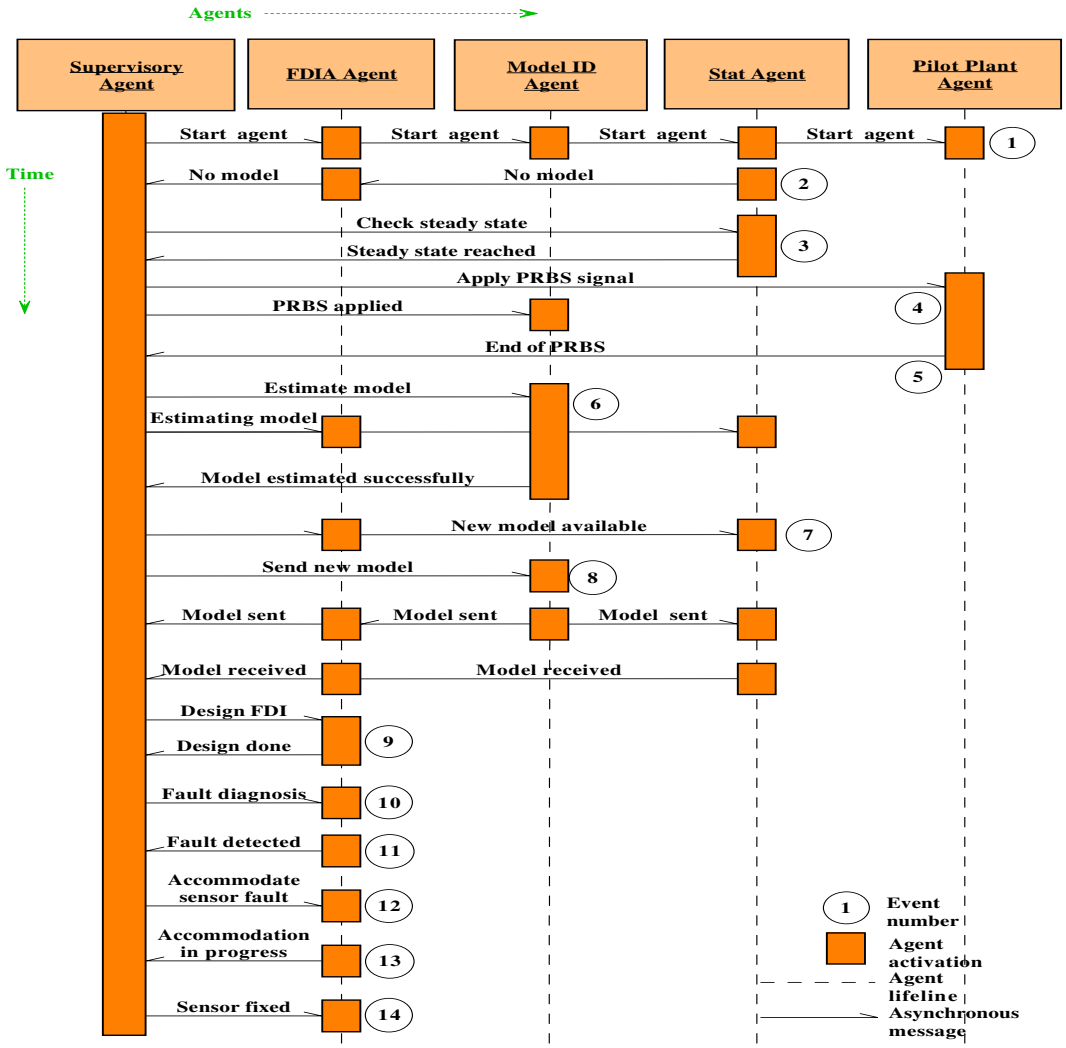

Fig. 3. ICAM system prototype event sequence

situations in future work. The ICAM system supervisory agent starts up the other reactive agents (i.e., event # 1), which are implemented as MATLAB functions and scripts for ease of development and debugging [5].

If the FDI agent or the statistical pre-processing agents do not have any process model (event # 2), they report their status to the supervisory agent, which, in turn, commands the statistical pre-processor to check if the external plant is in steady state (event # 3). If the external plant is in a steady state, the supervisory agent asks the low level control system to apply a small pseudo random binary signal (PRBS) for a specified period of time $\Delta t = 300s$ (event # 4). The model ID agent collects process data during the application of the PRBS signal. Once the low level control system flags back the end of PRBS signal application to the supervisory agent (event # 5), the supervisor flags to the model ID agent to estimate a new process model (event # 6), and informs the statistical pre-processing and the FDIA agents that a new model is being estimated. If the model was estimated successfully, the supervisory agent informs other agents that a new model is available to be updated (event # 7).

The estimated model is then sent to the appropriate agents (event # 8), which, in turn, report the model reception status back to the supervisory agent. The supervisor then requests that the FDIA agent design the FDI filter based on the received process model (event # 9). The FDIA agent starts monitoring the external process for sensor and actuator faults (event # 10) [23], [26]. If the FDIA agent detects a fault in the plant, the fault location, type, time, size and other information are reported back to the supervisor for further processing (event # 11). In the case of a sensor fault, the FDIA agent will also recommend the appropriate accommodation (correction) (event # 12) [27]. The sensor accommodation status is reported continuously to the supervisory agent (event # 13), which terminates the sensor accommodation task if the sensor has been fixed (event # 14).

V. Design of Reactive Agents

The simplified ICAM system prototype consists of four reactive agents, namely, the pilot plant simulation agent, the

data statistical preprocessing agent, the model identification agent, and the FDIA agent, whose design is discussed in the following sections. The interaction of the supervisory agent with the reactive agents are indicated on their flow charts by the event numbers, which have been discussed in the previous section (refer to figure 3).

A. The pilot plant agent design

The pilot plant model represents an oil production facility, which separates oil well fluids into crude oil, sales gas, and water. The simulation model basically consists of two processes, as illustrated in figure 5. The first is a two-phase separator in which hydrocarbon fluids from oil wells are separated into two phases to remove as much light hydrocarbon gases as possible. The produced liquid is then pumped to the three-phase separator (i.e., the second process), where water and solids are separated from oil. The produced oil is then pumped out and sold to refineries and petrochemical plants if it meets the required specifications. Gas is processed further and sent as sales gas.

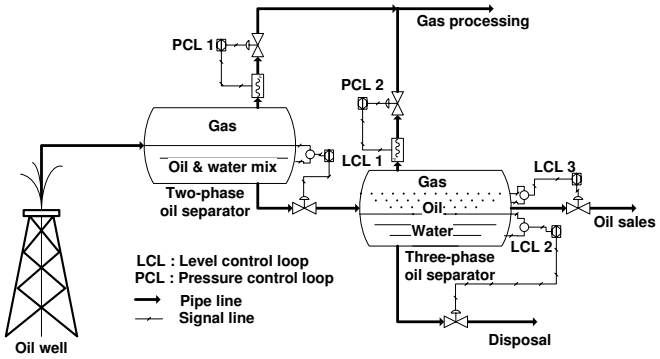

Fig. 5. Oil production facility P&ID

The two separation processes of the simulation model are controlled to maintain the operating point at its nominal value, and to minimize the effect of disturbances on the produced oil quality. As shown in figure 5, the first separation process is controlled by two PI controller loops. In the first loop, LCL1, the liquid level is maintained by manipulating the liquid outflow valve. The second loop, PCL1, controls the pressure inside the two-phase separator by manipulating the amount of the gas discharge. The second separation process has three PI controller loops. An interface level PI controller, LCL2, maintains the height of the oil/water interface by manipulating the water dump valve. The oil level is controlled by the second PI controller, LCL3, through the oil discharge valve, and the vessel pressure is maintained constant by the third PI loop, PCL2 [32].

The plant ordinary differential equation (ODE) model was simulated in real time every $\delta t = 100$ milliseconds using the fixed-step first-order Euler ODE solver. Ten sensor and actuator faults were embedded in the plant model to validate the ICAM system performance and logical behavior during faulty situations. Figure 6 illustrates the flow chart of the pilot plant simulation model agent. After the supervisory agent starts up the plant agent, the plant agent initializes its MPI and G2 communication links, and it enters in a wait loop till it receives a valid scenario to run. Four scenarios are embedded in the plant agent: the first is the default scenario which runs the plant at its nominal operating point; the second scenario allows the set points of the plant to be changed from the nominal operating point; the third scenario applies a disturbance to the plant; and the final scenario simulates the plant during sensor and actuator faults.

The four scenarios add richness to the plant agent and facilitate demonstrating the performance and logical behavior analysis of the ICAM system prototype during different situations. Once the specified scenario is chosen to run, the agent starts the simulation and sends raw data messages to the statistical pre-processing agent. The capability of applying a small PRBS signal is incorporated in the plant agent for model identification purpose. If the plant agent receives a request from the supervisory agent to apply PRBS signals (event # 4), then the plant agent applies a PRBS signal to each plant input for a time period of $T_{ID} = 300$ second, and flags back to the supervisory agent the end of PRBS signal application flag (event # 5).

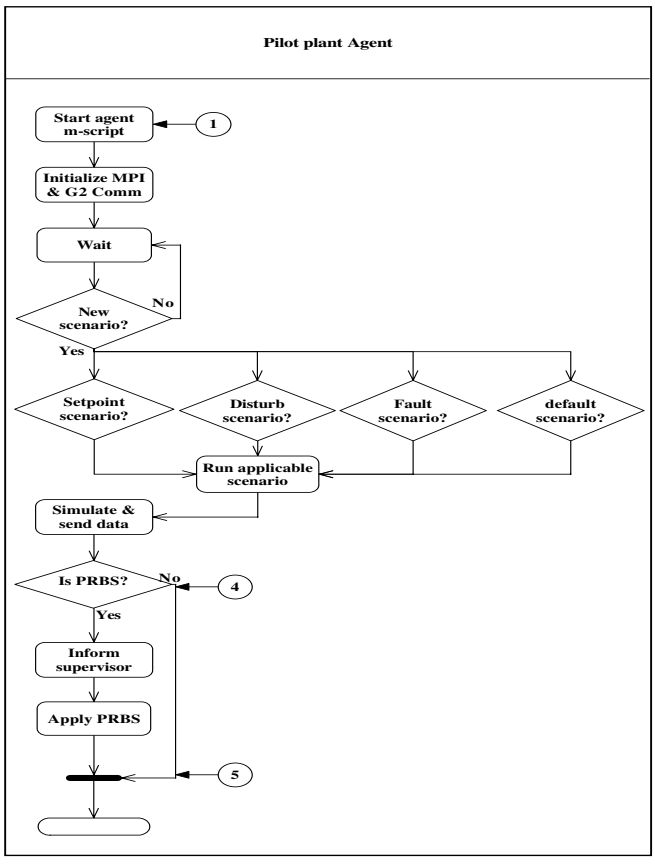

Fig. 6. The pilot plant simulation agent flow chart

B. The statistical pre-processing agent design

Figure 7 illustrates the flow chart of the data statistical pre-processing agent flow chart, where the agent is started up by the supervisory agent (event # 1), and its G2 and MPI communication links are initialized. Once the statistical preprocessor agent receives raw data from the plant agent,

it stores it in a data window for further processing. If the agent receives a request from the supervisory agent to check the steady state status of the plant agent (event # 3), it checks the steady state and report the result back to the supervisory agent. The statistical pre-processing agent informs the supervisory agent that it does not have any plant model after its startup (event # 2).

If there is a new plant model available, the supervisory agent informs the statistical pre-processing agent to update the plant model (event # 7). The agent then removes missing data and outliers by exploiting the median absolute deviation algorithm [33]. The data are then reconciled according to a pre-specified material balance for quality control (this functionality may be implemented in future work), and is sent to other agents for further processing. The processed data are also sent to a graphic user interface (GUI) agent to allow plant operators to interact with processed data (e.g., zoom, store, and plot the data) in a friendly manner. Should the statistical pre-processing agent fail internally, it reports its failure mode status to the supervisory agent for further actions. Internal failure could happen during sensor/actuator faults, which leads to an ill-posed optimization problem to solve in the data reconciliation task (to be implemented in future work).

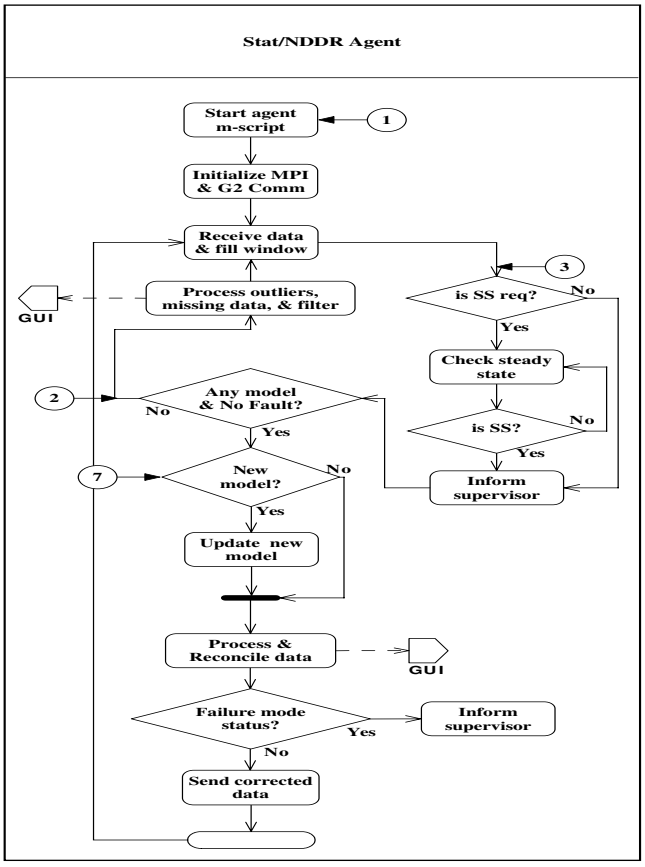

Fig. 7. The statistical pre-processing agent flow chart

C. The model identification agent design

The model ID agent estimates the multivariable plant model by using the subspace method, which uses the canonical variable algorithm (CVA) in its singular value decomposition stage [34], [35], [36]. Figure 8 illustrates the flow chart of the model identification agent, where the agent is started up by the supervisory agent (event # 1). The model ID agent then initializes its own MPI and G2 communication links. The agent stays in an idle state, unless the supervisory agent informs it that a PRBS signal is being applied by the pilot plant simulation agent (event # 4). Consequently, the agent starts receiving processed data from the statistical processing agent.

After the end of the PRBS signal application (event # 5), the supervisory agent informs that model ID agent to estimate a new model (event # 6). If the plant model estimation is successful, the model is sent to the statistical processing and the FDIA agents for further processing (event # 8). If not, the estimated model diagnostics are updated and sent to the supervisory agent to take the appropriate decision.

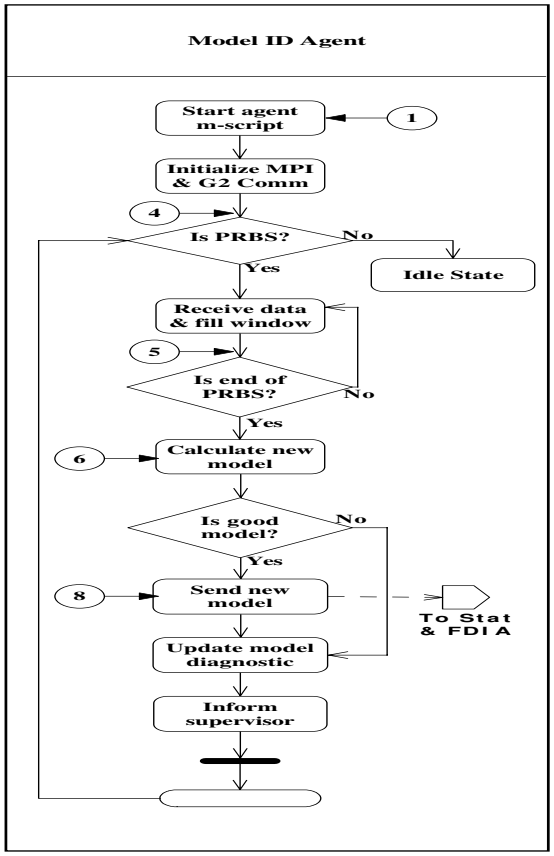

Fig. 8. The model identification agent flow chart

D. The fault detection, isolation, and accommodation agent design

The FDI agent exploits the generalized parity space (GPS) to generate a set of directional residuals, from which process faults can be determined [30], [19], [23], [31], [26], [27].

After the supervisory agent starts up the FDIA agent (event # 1), it initializes it G2 and MPI communication links [27], as shown in figure 9. If the FDIA agent has no plant model, it reports that back to the supervisory agent for further actions (event # 2). When a new plant model is available (event # 7), the FDIA agent updates its knowledge about the plant. The supervisory agent then instructs the FDIA agent to design its FDI filter based on the newly updated plant model (event # 9). When that task is completed, the FDIA agent then starts to monitor the plant data to detect any sensor/actuator faults (event # 10).

If the decision maker of the FDIA agent detects a fault (event # 11), it alerts the supervisory agent and sends the detected fault location, type, time, size, and other information to the supervisory agent for further processing. If the fault is a sensor fault (event # 12), the supervisory agent alerts the FDIA agent to accommodate the sensor fault (event # 13). If the sensor the fault type is bias, the fault size is estimated and used to accommodate the fault without estimating the fault size recursively. If the sensor fault type is a ramp type, this starts the recursive fault size estimation, and the fault is accommodated accordingly. The accommodation process terminates if the accommodation stopping criterion is reached, or if the supervisory agent informs the FDIA agent that the faulty sensor is fixed (event # 14).

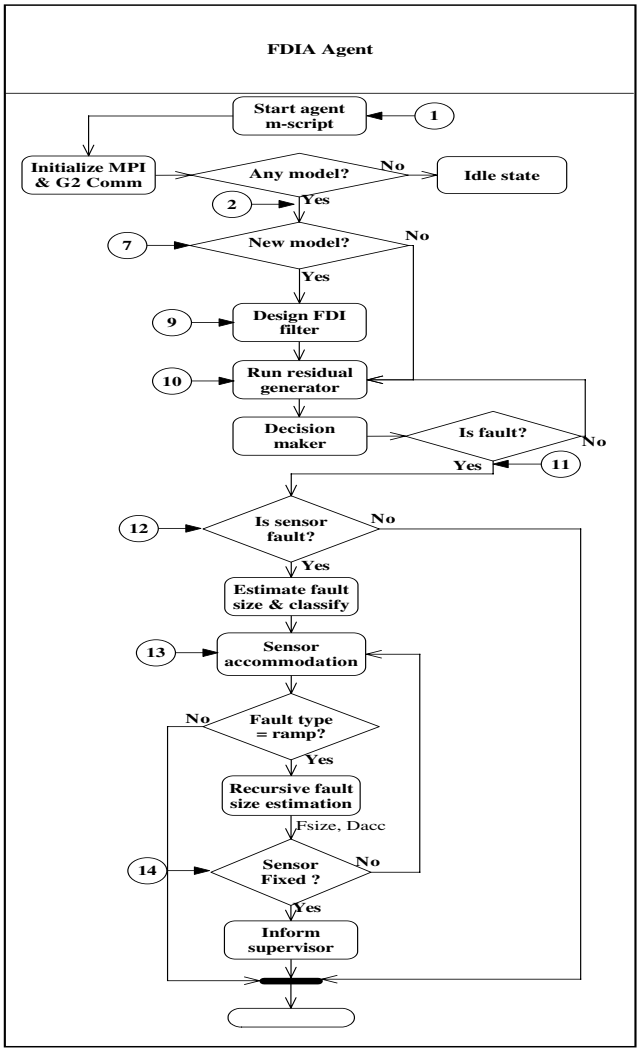

Fig. 9. The FDIA agent flow chart

VI. ICAM System Prototype Deployment Scheme

Having discussed the ICAM system prototype design with respect to the artificial intelligence, the middleware, and its reactive agents' requirements, it is crucial to design the prototype deployment scheme to verify its performance and logical behavior. Figure 10 illustrates the ICAM system prototype deployment scheme. The ICAM system prototype contains two Windows 2003 servers, which provide the local network infrastructure of the system. The prototype also consists of two PC nodes, in which the five agents of the ICAM system prototype are deployed. The first node runs three agents, namely, the pilot plant simulation agent, the model ID agent, and the G2 supervisory agent. The second node runs the data statistical processing agent and the FDIA agent. Each reactive agent consists of its m-script, its middleware task, and its associated MATLAB session. This deployment scheme was set up to conduct a real-time simulation experiment so as to validate the logical behavior of the designed system prototype during plant abnormal situations. The real-time experiment and its results are discussed in the second part of this paper.

VII. Conclusions

We have demonstrated good progress the design and development of the ICAM system. A system prototype design was extended by adding more agents and functionalities to cope with the complex problem of control of large and complex industrial plants. The ICAM system prototype design was discussed in details in terms of the middleware layer, the supervisory agent layer, and the reactive agents layer. A system deployment scheme was also suggested to conduct a real-time system simulation experiment, which will be discussed in the second part of this paper. Our system design approach can be exploited to develop and rapidly prototype real-time distributed multi-agent systems. We believe that the ICAM system will pave the way to real intelligent multi-agent systems for many applications.

VIII. Acknowledgement

This project is supported by Atlantic Canada Opportunities Agency (ACOA) under the Atlantic Innovation Fund (AIF) program. The authors gratefully acknowledge that support and the collaboration of Cape Breton University (CBU), and the College of the North Atlantic (CNA). The authors also acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) for funding the second author's research.

References

[1] J. H. Taylor and A. F. Sayda, "Intelligent information, monitoring, and control technology for industrial process applications," in *The 15th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM)*, Bilbao, Spain, July 2005.

[2] ——, "An intelligent architecture for integrated control and asset management for industrial processes," in *Proc. IEEE International Symposium on Intelligent Control (ISIC05)*, Limassol, Cyprus, June 2005, pp. 1397–1404.

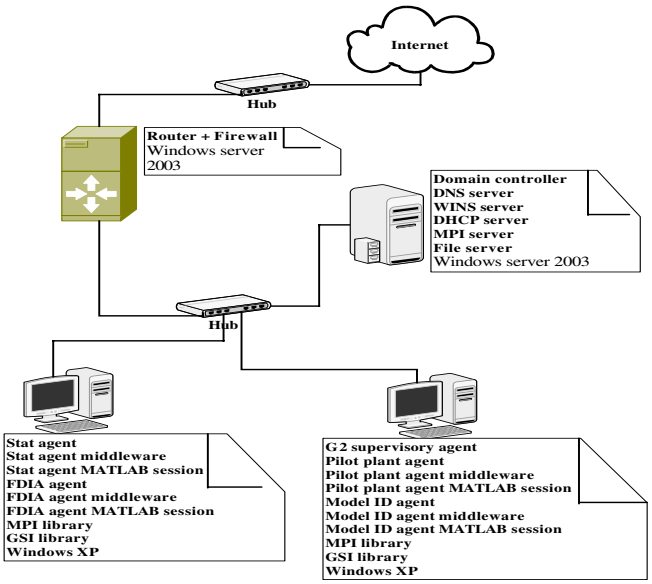

Fig. 10. ICAM system prototype deployment scheme

[3] A. F. Sayda and J. H. Taylor, "An implementation plan for integrated control and asset management of petroleum production facilities," in *IEEE International Symposium on Intelligent Control ISIC06*. Munich, Germany: IEEE, October 4-6 2006, pp. 1212–1219.

[4] ——, "An intelligent multi agent system for integrated control and asset management of petroleum production facilities," in *In Proc. of The 17th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM)*, Philadelphia, USA, 18-20 June 2007, pp. 851–858.

[5] ——, "Toward a practical multi-agent system for integrated control and asset management of petroleum production facilities," in *IEEE International Symposium on Intelligent Control (ISIC)*, Singapore, 1–3 October 2007.

[6] J. H. Taylor and A. F. Sayda, "Prototype design of a multi-agent system for integrated control and asset management of petroleum production facilities," in *submitted to the American Control Conference (ACC)*, Seattle, Washington, June 11–13 2008.

[7] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis part 1, 2, 3," *Computer & Chemical Engineering*, vol. 27, no. 3, pp. 293–346, 2003.

[8] R. J. Patton, "Fault-tolerant control systems: The 1997 situation," in *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, R. J. Patton and J. Chen, Eds., vol. 3. Kingston Upon Hull, UK: IFAC, August 1997, pp. 1033–1054.

[9] P. M. Frank and B. Köppen-Seliger, "New developments using AI in fault diagnosis," *Engineering Applications of Artificial Intelligence*, vol. 10, no. 1, pp. 3–14, 1997.

[10] R. L. Small and C. W. Howard, "A real-time approach to information management in a pilot's associate," in *Proceedings of Digital Avionics Systems Conference*. IEEE/AIAA, 14–17 Oct 1991, pp. 440–445.

[11] S. B. Banks and C. S. Lizza, "Pilot's associate: a cooperative, knowledge-based system application," *IEEE Expert*, vol. 6, no. 3, pp. 18–29, June 1991.

[12] C. A. Miller and M. D. Hannen, "Rotorcraft pilot's associate: Design and evaluation of an intelligent user interface for cockpit information management," *Knowledge-Based Systems*, vol. 12, no. 8, pp. 443–456, Dec 1999.

[13] B. Köppen-Seliger, T. Marcu, M. Capobianco, S. Gentil, M. Albert, and S. Latzel, "MAGIC: An integrated approach for diagnostic data management and operator support," in *Proceedings of the 5th IFAC Symposium Fault Detection, Supervision and Safety of Technical Processes - SAFEPROCESS05*, Washington D.C., 2003.

[14] J. Schmalzel, F. Figueroa, J. Morris, S. Mandayam, and R. Polikar, "An architecture for intelligent systems based on smart sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 4, pp. 1612–1616, August 2005.

[15] T. Cochran, P. Bullemer, and I. Nimmo, "Managing abnormal situations in the process industries parts 1, 2, 3," in *NIST Proceedings of the Motor Vehicle Manufacturing Technology (MVMT) Workshop*, Ann Arbor, MI, 1997.

[16] S. Cauvin, "CHEM-DSS: Advanced decision support system for chemical/petrochemical manufacturing processes," in *CHEM Project Annual Meeting*. Lille, France: http://www.chem-dss.org/, 25-26 March 2004.

[17] E. L. Cochran, C. Miller, and P. Bullemer, "Abnormal situation management in petrochemical plants: can a pilot's associate crack crude," in *Proceedings of the 1996 IEEE National Aerospace and Electronics Conference, NAECON*, vol. v2. Dayton, KY, USA: IEEE, Piscataway, NJ, USA, May 20-23 1996, pp. 806–813.

[18] A. Ogden-Swift, "Reducing the costs of abnormal situations ... the next profit opportunity," in *IEEE Advanced Process Control Applications for Industry Workshop (APC2005)*, Vancouver, Canada, May 2005.

[19] M. Omana and J. H. Taylor, "Robust fault detection and isolation using a parity equation implementation of directional residuals," in *IEEE Advanced Process Control Applications for Industry Workshop (APC2005)*, Vancouver, Canada, May 2005.

[20] W. Larimore, in *Multivariable System Identification Workshop*. Fredericton, New Brunswick: University of New Brunswick, 31 October – 2 November 2005.

[21] C. Smith, C. Gauthier, and J. H. Taylor, in *Petroluem Applications of Wireless Sensors* (PAWS) *Workshop*. Sydney, Nova Scotia: Cape Breton University, 22–23 August 2005.

[22] E. Durfee, V. R. Lesser, and D. D. Corkill, "Trends in cooperative distributed problem solving," *IEEE Transactions on Knowledge and Data Engineering*, vol. 1, no. 1, pp. 63–83, 1989.

[23] M. Omana and J. H. Taylor, "Enhanced sensor/actuator resolution and robustness analysis for FDI using the extended generalized parity vector technique," in *Proc. of American Control Conference*. Minneapolis, Minn.: IEEE, 14-16 June 2006, pp. 2560–2566.

[24] J. H. Taylor and M. Laylabadi, "A novel adaptive nonlinear dynamic data reconciliation and gross error detection method," in *Proc. of IEEE Conference on Control Applications*. Munich, Germany: IEEE, October 4-6 2006, pp. 1783–1788.

[25] M. Laylabadi and J. H. Taylor, "ANDDR with novel gross error detection and smart tracking system," in *12th IFAC Symposium on Information Control Problems in Manufacturing*. Saint-Etienne, France: IFAC, May 17-19 2006.

[26] M. Omana and J. H. Taylor, "Fault detection and isolation using the generalized parity vector technique in the absence of a mathematical model," in *IEEE Conference on Control Applications (CCA)*, Singapore, 1-3 October 2007.

[27] J. H. Taylor and M. Omana, "Fault detection, isolation and accommodation using the generalized parity vector technique," in *submitted to the IFAC World Congress*, Seoul, Korea, July 6–11 2008.

[28] W. Gropp, E. Lusk, and R. Thakur, *Using* MPI-2*: Advanced features of the message-passing interface*, ser. Scientific and Engineering Computation. Cambridge, Massachusetts: MIT Press, 1999.

[29] G2 *for Application Developers Reference Manual*, 8th ed., GenSym Corporation, Burlington, Massachusetts, December 2005.

[30] N. Viswanadham, J. H. Taylor, and E. C. Luce, "A frequency domain approach to failure detection and isolation with application to GE21 turbine engine control system," *Control Theory and Advanced Technology*, vol. 3, no. 1, pp. 45–72, 1987.

[31] M. Omana, "Robust fault detection and isolation using a parity equation implementation of directional residuals," Master's thesis, University of New Brunswick, 2005.

[32] A. F. Sayda and J. H. Taylor, "Modeling and control of three-phase gravity separators in oil production facilities," in *the American Control Conference (ACC)*, New York, NY, 11-13 July 2007.

[33] P. H. Menold, R. K. Pearson, and F. Allgower, "Online outlier detection and removal," in *Proc. of the 7th Mediterranean Conference on Control and Automation (MED99)*, Haifa, Israel, June 28–30 1999.

[34] P. vanOverschee and B. DeMoor, *Subspace Identification of Linear Systems: Theory, Implementation, Applications*. Kluwer Academic Publishers, 1996.

[35] W. E. Larimore, "Canonical variate analysis in identification, filtering and adaptive control," in *In Proc. 29th IEEE Conference on Decision and Control*, Honolulu, 1990, pp. 596–604.

[36] L. Ljung, *System Identification - Theory For the User*, 2nd ed. Upper Saddle River, N.J.: PTR Prentice Hall, 1999.