We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

23

Multi-agent Systems for Industrial Applications: Design, Development, and Challenges

Atalla F. Sayda
University of New Brunswick
Canada

1. Introduction

Complex industrial facilities are characterized by their large size, complexity of the constituent sub-processes and their dynamics, and the massive information overload. For example, the maintenance and management of complex process equipment and processes, and their integrated operation, play a crucial role in ensuring the safety of plant personnel and the environment as well as the timely delivery of quality products. Given the size, scope and complexity of the systems and interactions, it is becoming difficult for plant personnel to anticipate, diagnose and control serious abnormal events in a timely manner (Venkatasubramanian et al., 2003).

This lays huge challenges on the design, development and deployment of a system to manage such facilities in different operating conditions. Multi-agent systems (MAS) and artificial intelligence (AI) are the corner-stone frameworks to design an intelligent real-time system for complex industrial facility management and control. This chapter addresses the different aspects of designing a multi agent system to manage and control complex industrial facilities from conceptualization to deployment. A thorough literature survey is done to give a big picture of what has been accomplished in academia and industry for this topic primarily. The design of an industrial MAS system will be discussed from conceptualization to validation in the following sections.

2. Challenges in developing industrial multi-agent systems

Although industrial multi-agent systems have great impact on complex process plants in terms of higher profitability and better management, the development of such systems is very difficult and exhibits many challenges (Mark et al., 1995; Mylaraswamy, 1996; Mylaraswamy & Venkatasubramanian, 1997; Vedam, 1999; Vedam et al., 1999; Venkatasubramanian, 2005; Venkatasubramanian et al., 2003):

- Diversity of solution techniques, where several approaches are available to perform the main tasks of an industrial facility management system: For example, fault detection and isolation can be performed using model-based quantitative and qualitative fault diagnosis techniques as well as non-model-based methods. Similarly, supervisory control can be performed using several AI techniques such as rule-based expert systems and case-based reasoning. These techniques are diverse in nature and use certain assumptions about the process and performance requirements. Hence, determining the best approaches for

performing the individual tasks of industrial facility management is difficult. Moreover, the chosen approaches may not meet the goals of the overall system. Having these techniques integrated in one intelligent system may seem the only solution. However, the analysis of the accuracy, consistency and stability of such integrated systems is even more difficult.

- Diverse sources of knowledge, which stems from the incomplete and scattered nature of process knowledge such as process manuals, operational expertise, process models, and historical data: Techniques to integrate the knowledge sources into a form that can be used effectively in an intelligent system are of a great necessity. The ontology based knowledge organization approach is an example of such techniques.
- Uncertainty in process models and measurements, which may affect the performance of a complex industrial facility system: Most of the system's tasks depend on accurate process measurement and models. Noisy sensor measurements, process disturbances, and the highly non-linear dynamics of chemical processes in general can be a significant source of the failure of the entire system. Systematic analysis of such uncertainties and their effect on the system performance is required.
- Widely varying time scales of the different system tasks and the operating situations which may happen in the plant: Some operating situations might develop over a few minutes, while others might develop over hours and days. Likewise some tasks of the system might execute in few milli-seconds, while other tasks may take minutes and hours to make decisions such as the supervisory control task.
- Implementation for large scale industrial plants, which has effect on the system software architecture, real-time hardware, field testing and validation, user interface and operator training and acceptance.

3. Survey of industrial multi-agent systems (MAS)

The automation of industrial facility management has been recognized by academia and industry as a vital research area, which many research programs and industrial projects were initiated to investigate. Some projects focused on managing the process during normal operation while others gave abnormal situation management a higher priority. In the late 1980's, the European Commission funded a major research project called ARCHON, which was focused on the problem of getting a number of distinct expert systems to pool their expertise in solving problems and diagnosing faults in several industrial domains. ARCHON was recognized as one of the first real industrial applications of MAS (Jennings & Mamdani, 1996). In this section, an indepth survey of research and industrial projects will be presented. The early projects did not exploit the MAS approach, yet they gave an insight for later research projects, which benefited from the MAS approach. Here eight of the most relevant projects are reviewed.

3.1 The FORMENTOR research project

FORMENTOR, which is a joint venture of major European companies, is supported by the EUREKA program of cooperative international R&D projects with a budget of 33.4 million Euros. The research program, which lasted from 1986 to 1996, "aimed to develop real-time plant supervision software systems to support operators in their decision-making process by enabling them to make effective use of all this information, and avoid disturbances and any

loss of production" (Kim & Modarres, 1987; Wilikens & Burton, 1996). The main technical features of the system are (Wilikens & Burton, 1996):

1. A goal tree-success tree (GTST), which is a representation of the functional model of the process.
2. The multi-layer model (MLM) used to represent the functional components of the plant in a hierarchical way to provide a global overview of the plant state and to guide the action planning process.
3. Two distinct but complementary reasoning modules for diagnosis, both based on this multi-layer structure.

In FORMENTOR the object oriented approach was used to implement the system, which in turn consisted of a collection of modules to perform the different tasks, a communication system, and a global controller to control the activities of the modules. The disadvantages of such a system can be summarized as follows:

1. The use of simple mathematical formulas to represent the behavioral models of the plant components, where the complex interactions among the plant variables are not captured.
2. The absence of a model identification module to address the dynamic and varying nature of chemical plants.
3. Fault diagnosis is largely qualitative, which leads to lack of resolution.
4. Fault mitigation is also qualitative and based on the current state of the system, making the system only reactive.

3.2 Advanced process analysis and control system (APACS)

The APACS project was designed to introduce intelligent agents into a modern nuclear operating environment. APACS was a PRECARN project supported by the Canadian government and several Canadian universities and companies, and developed by a team of more than 10 software designers and engineers. The 9.7 million-Canadian-dollars, five-year project began in the fall of 1990 and was completed in the fall of 1995 (Mylopoulos et al., 1992; Wang & Wang, 1997). "The goal of the APACS project was to develop a generic framework for building an intelligent system that assists human operators of power plants in noticing and diagnosing failures in continuous processes" (Wang & Wang, 1996).

The APACS system consists of three layers: the agent layer which implements the system functionality; the knowledge broker layer which manages communication between the agents; and the information repository layer which stores the system common knowledge. The agents of the APACS system perform the following tasks (Wang & Wang, 1997):

1. The data acquisition agent receives data from the plant's main control computer.
2. The tracking agent continuously updates the data links between the agent system and the actual plant sensor positions.
3. The monitoring agent analyzes the feedwater sensor values and feedwater alarms and then produces symbolic descriptions of the plant's behavior.
4. The human-computer interface (HCI) agent displays the APACS status to the plant operators and serves as the user interface.
5. The diagnostic agent takes the output from the monitoring agent and attempts to generate a qualitative causal explanation that will eventually be useful to the human operators.

6. The verification agent operates a faster-than-realtime numerical, model-driven simulator to measure the correlation of the diagnostic agent's output against the simulator's ideal values.

The entire APACS system was implemented in C++ using Expersoft's XShell (an extended C++ syntax for declaring distributed objects) as its communication environment and CLIPS (a rules-based inferencing environment constructed by NASA) (Wang & Wang, 1997). The APACS project had some of the FORMENTOR project disadvantages such as the absence of a model identification agent, the qualitative nature of the fault diagnosis task, and the absence of a fault mitigation (i.e., accommodation) agent.

3.3 The pilot's associate (PA) program

The first research program to address the intelligent facility management problem in the US was the Pilot's Associate (PA) program, which is a joint effort of the Defense Advanced Research Projects Agency and the US Air Force, managed by the Air Force's Wright Laboratory. The program began in February 1986 as an application demonstration for DARPA's Strategic Computing Initiative. A primary goal of the PA program was to enhance combat fighter pilot effectiveness by increasing situational awareness and decreasing their workload. DARPA wanted to advance the program's technology base, principally in the area of real-time, cooperating knowledge-based systems. The Air Force wanted to explore the potential of intelligent systems applications to improve the effectiveness and survivability of post-1995 fighter aircraft (Banks & Lizza, 1991; Small & Howard, 1991).

"The Pilot's Associate concept developed as a set of cooperating, knowledge-based subsystems: two assessor and two planning subsystems, and a pilot interface. The two assessors, Situation Assessment and System Status, determine the state of the outside world and the aircraft systems, respectively. The two planners, Tactics Planner and Mission Planner, react to the dynamic environment by responding to immediate threats and their effects on the pre-briefed mission plan. The Pilot-Vehicle Interface subsystem provides the critical connection between the pilot and the rest of the system" (Banks & Lizza, 1991). Another project, which followed the PA program to address the facility management problem in attack helicopters, is the Rotorcraft Pilot's Associate (RPA) program. The goal of the US Army funded RPA program was to develop and demonstrate in flight an advanced, intelligent "associat" system in a next-generation attack/scout helicopter (Miller & Hannen, 1999).

3.4 Abnormal situation management (ASM)

The PA and RPA projects paved the way for other projects to develop and automate the industrial facility management process for the process industry in the United States. AEGIS (Abnormal Event Guidance and Information System), which was developed by the Honeywell led Abnormal Situation Management (ASM) Consortium in the United States, is a very important project (Cochran et al., 1997). The AEGIS project proposes a comprehensive facility management framework from an industrial view point. AEGIS built on the experience of military aviation research projects, especially the Pilot's Associate (PA) and the Rotorcraft Pilot's Associate (RPA) (Cochran et al., 1996). It is really worth considering the project and its current status, since it is supported by major oil and gas companies allied with Honeywell and other automation industry key leaders. Furthermore, it is considered a research imperative to learn from it, in terms of experience, stages being successfully accomplished, limitations, and failures incurred during the course of the project. The research program life span started from 1994 and ended in 2008, where the program was funded by the National Institute of

Standards and Technology (NIST). The program focused on the development of a proof of concept system called AEGIS (Abnormal Event Guidance and Information System), which have gone through different development stages.

3.4.1 Hybrid distributed multiple expert framework (DKIT)

The diagnostic toolkit (DKIT) project was initiated as the first step in the design and development of the AEGIS system. The DKIT hybrid framework addressed the use and integration of multiple fault diagnosis techniques to meet the challenges of complex, industrial-scale diagnostic problems (Mylaraswamy, 1996; Mylaraswamy & Venkatasubramanian, 1997). The principle of DKIT is black-board collective problem solving, in which several modules are integrated (Mylaraswamy, 1996):

- Diagnostic experts: a collection of one or more fault diagnostic modules including a signed directed graph (SDG) technique, qualitative trend analysis (QTA), and probability density function based statistical classifier.
- A blackboard: a placeholder for various process states. This is implemented as pigeon holes, each of which corresponds to a well defined process state.
- A scheduler, which consists of a monitor that keeps track of new events or states that are posted on the blackboard; a switchboard which directs the information to relevant subscribers, and a mechanism for conflict resolution between the different diagnostic modules.
- A plant Input-Output Interface, which acts as a channel for all diagnostic modules to receive relevant process measurements.
- An operator interface for presenting diagnostic results to the operator.
- A process equipment library to represent the external process.

The DKIT system was fully implemented in the G2 expert system shell, and was validated on a simulation model of fluid catalyst cracking unit (FCCU). The DKIT framework demonstrated the feasibility of a complex fault diagnosis system, and was further enhanced through the development of the OP-AIDE system, which will be discussed in the next section.

3.4.2 Integrated operator decision support system (Op-Aide)

To address the qualitative fault analysis of previous projects (i.e., the FORMENTOR and APACS systems), an integrated operator decision support system, called Op-Aide, was developed based on the DKIT system architecture to assist the operator in quantitative diagnosis and assessment of abnormal situations (Vedam, 1999; Vedam et al., 1999). Op-Aide consists of six modules (or knowledge sources) and an Op-Scheduler that coordinates them. It provides the interface between different modules in the system and functions as a centralized data base for all the modules. The results of these modules are posted onto it, where they can be accessed by the other modules in the system (Vedam et al., 1999):

- Data Acquisition Module, which acquires on-line data from the plant and makes them available to other modules.
- Monitoring Module: This module monitors the process data for the presence of abnormalities using a principal component analysis (PCA) model of the process.
- Diagnosis Module, which identifies the root causes for the abnormalities. Multiple diagnosis methods are combined in a blackboard architecture.

- Fault Parameter Magnitude Estimation (FRAME) Module, which estimates the magnitude and rate of change of the root causes.
- Simulation Module, which performs a simulation to predict future values of the process outputs.
- Operator Interface Module, where the status of the process and the results of the different modules are constantly communicated to the operator through this module.

Op-Aide has been implemented using blackboard-based architecture in Gensym's expert system shell G2, MATLAB and C. The Op-Scheduler coordinates the functioning of other modules using event and time driven rules and procedures. The results of these modules are represented as objects that are pushed back onto specified slots in the OP-Scheduler. Most of the modules are implemented in G2 except for the FRAME and simulation modules, which are implemented in MATLAB and C respectively.
Although the OP-Aide project came to address the qualitative fault diagnosis disadvantage in the FORMENTOR and APACS systems by introducing two complementary quantitative fault diagnosis modules, it did not address the dynamic nature of the chemical process by embedding a model ID module. Furthermore, operating the situation assessment, which is achieved through the FRAME and simulation modules, is a semi-automatic process done at the request of the operator. OP-Aide did not address the whole performance aspect when it comes to managing large scale plants.

3.4.3 Abnormal Event Guidance and Information System (AEGIS)

The Honeywell ASM Consortium adopted the Dkit architecture as its AEGIS prototype, a next-generation intelligent control system for operator support (Venkatasubramanian et al., 2003). The AEGIS program successfully demonstrated the feasibility of collaborative decision support technologies in the lab test environment, with a high fidelity simulation model of an industrial manufacturing plant. As far as industrial environment testing is concerned, the focus was on abnormality diagnosis and early warning, and assessing and learning from experience, which resulted in effective operations practices and supporting services.
The AEGIS research program team has achieved several goals and developed a well established abnormal situation management awareness and culture through massive consultation, research, and collaboration with oil and gas industry key leaders. Achievements can be summarized in the following points as presented by the director of advanced development at Honeywell, Mr. A. Ogden-Swift, during the 2005 advanced process control applications for industry workshop (APC 2005) (Ogden-Swift, 2005):

- significant user interface (UI) improvements,
- 35% reduction in alarm flooding by introducing a new alarm reconfiguration philosophy,
- integration of operation procedures,
- equipment monitoring through intelligent sensor integration,
- fuzzy/PCA early error detection, and
- improved operator training.

Such achievements were deployed in the new generation of Honeywell's Experion distributed control system. Although the 12 year old AEGIS research program has resulted in a well defined abnormal situation management problem in terms of best practices, goals, and limitations, it did not address the following points, which aim to minimize the workload on process operators:

- full automation of massive process data interpretation,
- full automation of process fault diagnosis and accommodation,
- incorporation of state of the art fault diagnosis techniques which were developed during the past 25 years of academic research,
- reduced manual system configuration by process operators (for example, the operator has to choose the appropriate dataset for process model identification), and
- intelligent techniques such as expert systems to assist operators in the decision making process.

Only one technique was used for early fault detection, a statistical technique based on principal component analysis (PCA). To enable this, the operator has to manually adapt for operating point change by choosing the appropriate data set.

3.5 Advanced decision support system for chemical/petrochemical manufacturing processes (CHEM-DSS)

Another promising project is CHEM-DSS (Decision Support System for Chemical/Petrochemical Manufacturing Processes), which is an initiative of the European Community (EC) Intelligent Manufacturing Systems consortium in collaboration with Japan and Korea. "The aim of the CHEM-DSS project is to develop and implement an advanced Decision Support System (DSS) for process monitoring, data and event analysis, and operation support in industrial processes, mainly in refining, chemical and petrochemical processes" (Cauvin, 2004b).

The CHEM-DSS research project was initiated to compete and build on the two main initiatives in the United States, namely, the Abnormal Situation Management (ASM) Consortium led by Honeywell, and the Intelligent Control Program of NIST. However there was no clear system architecture that demonstrates the behavior of the integrated modules of the system during the course of the project (1998 - 2004). The research instead focused on analyzing the properties of the individual techniques of the system such as FDI, planning, artificial intelligence, signal processing, and scheduling, and twenty-three software toolboxes were developed during the project (from April 2001 to March 2004) (Matania1, 2005).

The heart of the CHEM-DSS integration platform is G2, which integrates the twenty-three software toolboxes. All developed software tools were integrated to a communication manager (CCOM) based on the message-oriented middleware (MOM). In this project the XMLBlaster open-source MOM was used to manage XML messages between the different tools. The data management and user interface functionalities were implemented in the G2 environment (Matania1, 2005).

Furthermore, "the toolboxes have been tested at pilot plants and industrial sites. It was applied to partner facilities to ensure rapid technology transfer. The industrial end-users provided different kinds of processes including a fluid catalytic cracking pilot plant, a paper making process, a gasifier pilot plant, a steam generator, a blast furnace and distillation process. End users can use the developed toolboxes to design their own intelligent diagnostic system according to their requirements" (Cauvin, 2004a).

3.6 Integrated system health management (ISHM)

The ISHM (Integrated System Health Management) research program, which is developed by NASA for space applications, "focuses on determining the condition (health) of every element in a complex System (detect anomalies, diagnose causes, prognosis of future anomalies), and

provide data, information, and knowledge to control systems for safe and effective operation. In the case of NASA, this capability is currently done by large teams of people, primarily from the ground, but needs to be embedded on-board systems to a higher degree to enable NASA's new Exploration Mission (long term travel and stay in space), while increasing safety and decreasing life cycle costs of spacecraft (vehicles; platforms; bases or outposts; and ground test, launch, and processing operations)" (Figueroa, Holland & Schmalzel, 2006). The ISHM research program, whose life span started from 2003 and will end in 2009, was extended to address several applications including military/civilian space and aircraft systems in collaboration with several companies such as Boeing and Honeywell (Derriso, 2005; Figueroa, Holland, Schmalzel & Duncavage, 2006; Garcia-Galan, 2005; Karsai et al., 2005; Maul et al., 2005; Schmalzel et al., 2005).

The ISHM architecture is based on the open systems architecture for condition-based maintenance (OSA-CBM), which is an implementation of the ISO standard # 13374. The ISHM system is deployed as a distributed module system with different functions including anomalies detection, overall systems state identification, anomaly and failure effects mitigation, and systems elements condition evaluation. The ISHM research project supported by NASA used the G2 environment as their intelligent integration framework. In fact, six G2 servers are deployed to monitor International Space Station (ISS) subsystems, including the mechanical, structural, electrical, environmental and computational systems. The G2 servers continually inspect and analyze data transmitted from space during missions (Figueroa, 2005; Maul et al., 2005).

3.7 Distributed architecture for monitoring and diagnosis (DIAMOND)

The DIAMOND project was developed by the University of Karlsruhe in cooperation with three industrial partners and one research institute within the framework of the EU Esprit Program with a budget of one million Euros. The program started in 1999 and ended in 2001, where the program objective was to investigate the feasibility of fault diagnosis system for industrial applications.

The DIAMOND system architecture is a set of distributed cooperating tasks. Each task is associated with a specialized agent, namely the monitoring agent, which is interfaced to the industrial application, a set of diagnostic agents to identify the functional state of the plant, a conflict resolution agent to investigate whether the diagnostic results are contradicting or completing each other, a facilitator agent to manage networking and mediating between different agents, a blackboard agent for storing the diagnoses, and a user interface agent for presenting the results to the operator (Worn, 2004).

The DIAMOND system was implemented using the KQML-CORBA- (knowledge query and manipulation language) based architecture, in which the different agents are implemented as distributed CORBA objects.The system prototype was evaluated while monitoring and diagnosing the water stream cycle chemistry of a coal-fired power plant (Worn, 2004).

3.8 Multi-agent-based diagnostic data acquisition and management in complex systems (MAGIC)

MAGIC is developed by a joint venture of several European universities and companies. The European Commission Information Society of Technology (EC-IST) funded the project with a budget of 3.3 million Euros. The MAGIC research program is a multi-agent system realization of an intelligent fault diagnosis system. "The system aims at developing general purpose architecture and a set of tools to be used for the detection and diagnosis of incipient

or slowly developing faults in complex systems. The early identification of potentially faulty conditions provides the key information for the application of predictive maintenance regimes" (Köppen-Seliger et al., 2003).
The distributed architecture for MAGIC is based on a multi-agents/multi-level concept. The idea is that the task of the complex system's diagnosis and operator support is distributed over a number of intelligent agents which perform their individual tasks nearly autonomously and communicate via the MAGIC architecture. Such an architecture can easily be distributed on existing monitoring and control systems of large scale plants (Garcia-Beltran et al., 2003; Köppen-Seliger et al., 2003). The MAGIC system consists of several model-based and cause-effect diagnostic agents and a process specification agent to specify the process to be monitored and diagnosed. Depending on the process specifications, the appropriate data and knowledge acquisition is performed by another agent. A diagnostic decision agent and a diagnostic support agent propose a final diagnostic decision, which is displayed with other information to an operator interface agent. The MAGIC system prototype is developed for the metal processing industry (Garcia-Beltran et al., 2003; Köppen-Seliger et al., 2003).

3.9 Intelligent control and asset management (ICAM) system

Having shown the current status of asset management research in both academia and industry, we conclude that the AEGIS research program focused on the bookkeeping and human machine interaction tasks rather than a fully automated and functional facility management holistic approach. Furthermore, the CHEM-DSS research program did not give a clear picture of how the different techniques will be integrated, and what software development tools/plans will be used to develop a prototype of the system. A new research program ICAM (Intelligent Control and Asset Management of industrial facilities) was initiated by a joint venture of Atlantic Canadian universities and National Research Council of Canada (NRC) to benefit from the success and limitations of the AEGIS, CHEM-DSS and other projects, to build on their experiences, to complement their developed tasks, and to push the envelope by evaluating and incorporating state of the art of fault diagnosis, artificial intelligence (AI) and wireless sensor networks techniques (Taylor, 2004). This will be embedded in a fully automated system architecture, which will better support process operators and improve operability (Larimore, 2005; Laylabadi & Taylor, 2006; Omana & Taylor, 2005; 2006; 2007; Sayda & Taylor, 2006; 2007a;b;c; 2008a;b;c; Smith et al., 2005; Taylor & Laylabadi, 2006; Taylor & Omana, 2008; Taylor & Sayda, 2005a;b; 2008).
Figure 1 illustrates the ICAM system, to which measured data from the industrial facility are transmitted, and interpreted for better process control and management. ICAM system is composed of a group of servers and operator work stations linked to each other through a high speed Ethernet network. The wireless sensor network is managed by a real time communication server. The database server stores received data in its database after being preprocessed. A group of application servers are the backbone of the ICAM system. The application servers run the tasks of data preprocessing, model identification, fault diagnosis, fault mitigation and accommodation, human machine interaction, and supervisory control. Each server is a computer cluster, which is a group of loosely coupled computers that work together closely to achieve higher performance, availability, and load balance. This will result in better internal coordination among the different ICAM servers. The conceptual model of the ICAM system is discussed in the following sections along with the system requirements analysis.

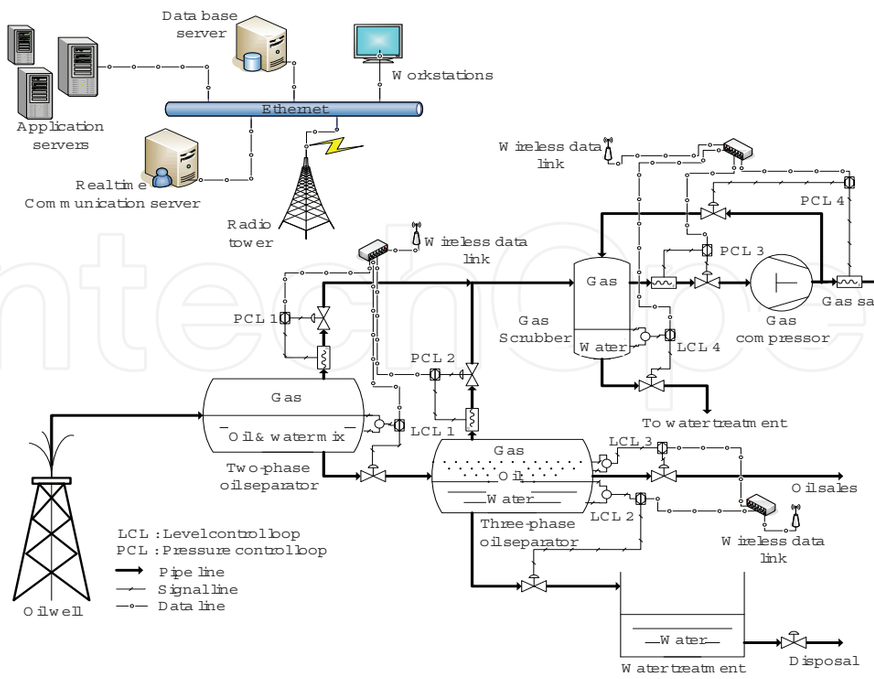


Fig. 1. ICAM Project schematic diagram

4. Conceptual model of the ICAM system

Several conceptual frameworks have been suggested for modeling complex intelligent systems. In the past two decades, the most popular design framework was the expert system, which has several advantages, namely, separation of knowledge and inference, ease of development and transparent reasoning under uncertainty. Moore and Kramer (Moore & Kramer, 1986) discussed the issues of expert system design for real-time process control applications; an intelligent expert system (PICON) was designed and implemented on several process plants to validate expert systems performance in real-time process environments. Implementation results revealed several drawbacks, namely, lack of learning mechanisms, knowledge base validation difficulties, and weak representation power. There are several expert system survey papers to which one may refer for further insight (Liao, 2004; Liebowitz, 1998).

Newell (Newell, 1990) introduced cognitive architectures as a more general conceptual framework for developing complex intelligent systems, based on a human cognition viewpoint. This approach assumes that human cognition behavior has two components, architecture and knowledge. The architecture is composed of cognitive mechanisms that are fixed across tasks, and basically fixed across individuals. These mechanisms, which define the properties of this approach, involve a set of general design considerations, namely, knowledge representation, knowledge organization, knowledge utilization, and knowledge acquisition. Newell argued that these considerations represent theory unification to model complex intelligent systems. Furthermore, this allows model (knowledge) reuse and helps create complete agents opening the way to applications. The performance of several tested cognitive architectures in solving different problems points to a promising future for modeling complex intelligent systems.

Multi-agent systems (MAS) which can be considered as an instantiation of distributed artificial intelligence, is another conceptual framework for modeling complex systems. A MAS is defined as a loosely coupled network of problem solvers that work together to solve problems, that are beyond their individual capabilities (Durfee & Montgomery, 1989). The MAS platform emphasizes distribution, autonomy, interaction (i.e., communication), coordination, and organization of individual agents. Agents in MAS can be defined as conceptual entities that perceive and act in a proactive or reactive manner within an environment where other agents exist and interact with each other based on shared knowledge of communication and representation (Wooldridge, 2002). Each agent contains processes for behavior generation, world modeling, sensory processing, and value judgment together with a knowledge database, as shown in figure 2.

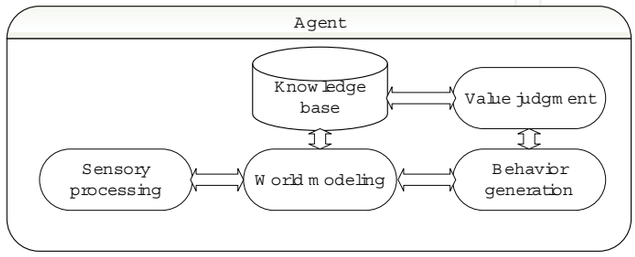


Fig. 2. Agent architecture

Sloman (Sloman & Scheutz, 2002) introduced H-Cogaff, a human-like information processing architecture, which contains many components performing different functions all of which operate concurrently and asynchronously. The H-Cogaff architecture seems to represent a combination of the cognitive architecture and the MAS conceptual frameworks. As illustrated in figure 3, Sloman's architecture provides a framework for describing different kinds of architectures and sub-architectures, and which, to a first approximation, is based on superimposing two sorts of distinctions between components of the architecture: firstly the distinction between perceptual, central and action components, and secondly a distinction between types of components which evolved at different stages and provide increasingly abstract and flexible processing mechanisms within the virtual machine (Sloman, 2001). The reactive components generate goal seeking reactive behavior, whereas the middle layer components enable decision making, planning, and deliberative behavior. The modules of the third layer support monitoring, evaluation, and control of the internal process in the lower layers.

Having reviewed the different conceptual modeling frameworks, it is our opinion that Sloman's H-Cogaff scheme is the best candidate, which would meet most of the requirements of an ICAM system for complex industrial plants. The architecture of the system and its functional modules will be discussed in subsequent sections.

5. ICAM System architecture and behavior model

Figure 4 illustrates the proposed architecture of the conceptual system, which consists of four information processing layers and three vertical subsystems, namely, perception, central processing, and action according to Sloman's H-Cogaff scheme. The lowest horizontal layer above the distributed control system (DCS) contains semi-autonomous agents that represent different levels of data abstraction and information processing mechanisms of the system. The middle two layers (i.e., the reactive and deliberative layers) interact with the external

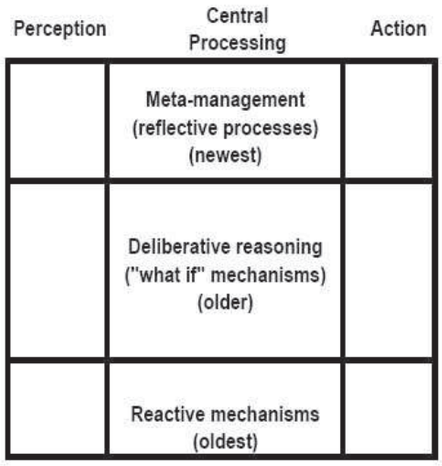


Fig. 3. Human cognition and affect (H-Cogaff) architecture Sloman & Scheutz (2002)

environment via the DCS and thus the industrial process by acquiring perceptual inputs and generating actions. The perceptual and action subsystems are divided into several layers of abstraction to function effectively. This can be achieved, for example, by categorizing observed events at several levels of abstraction, and allowing planning agents to generate behavior (actions) in a hierarchically organized manner.

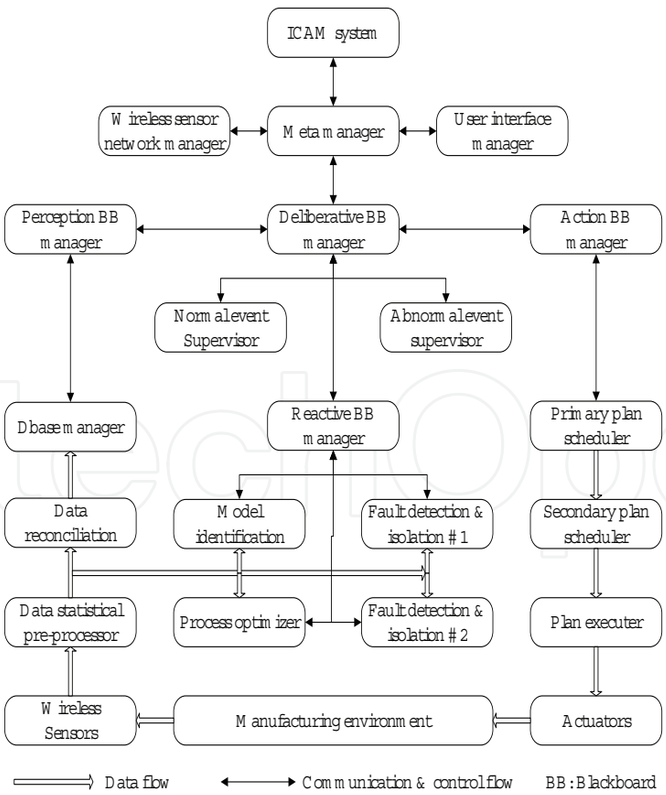


Fig. 4. ICAM system architecture and behavior model

The basic flow of control in the system begins when perceptual input arrives at the lowest level in the architecture. If the reactive layer can deal with this input then it will do so, otherwise, bottom-up activation will occur and control will be passed to the deliberative layer. If the deliberative layer can handle the situation then it will do so, typically by making use of top-down execution of reactive agents. Otherwise, it will pass control to the meta-management layer to resolve any internal conflicts in the architecture or notify the operator that it cannot do so. As illustrated in figure 4, the functionalities of the ICAM system agents in each layer are.

- Statistical pre-processing agent to clean the measured data from undesired discrepancies such as missed data and outliers.
- Data reconciliation agent to reconcile measured data according to mass and energy conservation laws.
- Database agent to store real-time data for historical and other purposes.
- Several fault detection and isolation (FDI) agents to detect any abnormal events using different FDI approaches.
- An optimization agent to generate the optimal operating plans.
- a system identification agent to identify the mathematical model of the industrial process.
- Two intelligent supervisory agents to manage the industrial plant during normal and abnormal situations.
- A meta manager agent to manage and coordinate the different system agents.
- A set of black board agents to facilitate asynchronous communications among the system agents.
- A set of task scheduling agents to execute the required plan according to different time frames.

Rigorous coordination of the behavior of the ICAM system layers and agents is crucial to success. A sound coordination scheme will allow us to assess its performance, and to evaluate how the internal agents of the system interact when certain internal/external events occur. Furthermore, it permits system behavior modeling to simulate the most critical design characteristics such as concurrency, autonomy, task distribution and parallelism, in order to guarantee robust and coherent performance. Due the complexity of modern manufacturing plants, intelligent systems (e.g., ICAM) have to be distributed, which makes the coordination of such systems very difficult and challenging.

Durfee *et al.* (Durfee & Montgomery, 1991) proposed an informal theory that integrates organizational behavior, long term plans, and short term schedules into one coordination framework, and treats coordination as a distributed search process through the hierarchical space of the possible interacting behaviors of the individual agents to find a collection that satisfactorily achieves the agents' goals. The theory emphasizes several topics such as:

- hierarchical behavior representation to express different dimensions of behavior at different levels of detail,
- metrics for measuring the quality of coordination between agents,
- distributed search protocol for guiding the exchange of information between agents during the distributed search,

- local search algorithm for generating alternative behaviors at arbitrary levels of abstractions, and
- control knowledge and heuristics for guiding the overall search process to improve coordination.

Durfee also suggested that introducing a meta-level organization in the intelligent system to manage coordination between agents, and separating knowledge representation into domain-level and meta-level types, would enhance coordination and make it more robust. Agents use domain-level knowledge to influence what goals they pursue, and use meta-level knowledge to decide how, when, and where to form and exchange behavioral models (Durfee et al., 1989). Durfee's informal theory and suggestions give the big picture of how agents should coordinate their activities within an intelligent system or even a society of intelligent agents. So far we have addressed the knowledge and organization separation issues by adopting the H-CogAff architecture proposed by Sloman. ICAM interacts with the external world through its reactive and deliberative agents, whereas the meta-level layer dictates the internal behavior of the system. Furthermore, domain-level knowledge is encoded in the deliberative agents and the meta-level knowledge is encoded in the self reflective layer.
As illustrated by figure 4, the proposed conceptual behavior model of the ICAM system was built upon our previous work in which we defined the architecture of the system, its functional modules, and its coordination mechanisms (Taylor & Sayda, 2005a;b). We adopted Sloman's H-Cogaff architectural scheme because it met most of our system requirements (Sloman, 2001). The behavioral model was drawn as a page hierarchy to make it compatible with hierarchical colored petri net (HCPN) terminology, which could be used to analyze the logical correctness and the dynamic behavior of the system; however, this has not been done.
The prime page in the model is called ICAM which contains all the subpages of the system. Each subpage represents an independent agent which interacts with others by means of communications (represented by thin bidirectional arrows). Other agents may process data received from the plant directly (data flow is represented by open thick unidirectional arrows). The meta manager is the main coordinator of the whole system, which guarantees more robust and coherent performance. The meta manager is basically a rule-based expert system, which codifies all possible system behaviors and agent interactions as a behavior hierarchy in its knowledge base. Agent behavior is represented in the behavior hierarchy by a single structure, which will use the same message structure communicated between agents. This will result in a better system performance. Table 1 illustrates the unified behavior conceptual structure.

Field name	Field content
Tag	Message ID
From	Sender
To	Recipient
What	Goals
How	Plans
When	Schedule
How long	Task length
Why	Meta reasoning

Table 1. Conceptual structure of behavioral message

6. Requirements analysis for the ICAM system

Having proposed a conceptual model, architecture, and behavioral model for the ICAM system, we define the autonomy, communications, and artificial intelligence (AI) requirements of the different agents of such a system. We also discuss the software implementation of the reactive and the supervisory agents.

6.1 Artificial intelligence (AI) requirements for the ICAM system

Among the industrial rule-based expert system shells, the G2 real-time expert system shell from Gensym Corporation (Gen, 2005) is considered the most versatile real-time expert system shell, as it integrates many software technologies and standards. The integration of the G2 expert system development environment with the ICAM system would benefit from and build on the previous G2 integration attempts. The G2 development environment offers a goal-based rapid prototyping design, in which requirements analysis, design, and development tasks are done simultaneously and incrementally during the ICAM system development life cycle. To meet the software requirements during the design and development of the ICAM system supervisory agent, AI design requirements such as the supervisory agent structure and knowledge representation have to be determined.

6.1.1 ICAM system supervisory agent implementation

Modules are the building blocks of complex G2 applications. A modular knowledge base (KB) consists of multiple G2 modules. The modules that make up an application form a module hierarchy, which specifies the hierarchical dependencies between modules (Gen, 2005). Decomposing a large project into multiple small modules allows developers to divide and merge work. Modules can be structural or functional ones. The structural modules contain classes or capabilities that need to be shared in large applications; functional modules implement well defined goals. The ICAM system supervisory agent, which potentially is a very complex artificial intelligence application, forms a good candidate for the modularization design approach. While the modularization design approach may add some overhead on the overall performance of the agent, it effectively organizes knowledge, and simplifies the development and deployment processes.

To meet the module reusability requirement, the guidelines for G2 application development recommend use of a four layer, two-module architecture, in which the graphical user interface (GUI) is in a separate module. The general architecture of the ICAM supervisory agent has two modules. The first module contains the agent's core functionality implementation layer and its application programmer's interface (API) layer, which protects the internal data structures in the core from corruption by other modules. The second module contains the public graphical user interface (GUI) layer and its GUI implementation layer, which interacts directly with the first module through its API layer. The ICAM system supervisory agent interacts with the other reactive agents through their external G2 links. The internal states of the ICAM system agents and the external environment are communicated to enable the supervisory agent to reason and make the correct and appropriate decisions for better system management.

6.1.2 Knowledge representation of the supervisory agent

The ICAM system supervisory agent may contain multi-faceted complex knowledge such as the internal structure of the ICAM system and the structure of the external environment (e.g., manufacturing plant topology, enterprise business structure). To represent such complex

knowledge, organizing the knowledge structure in the core layer of the supervisory agent as a hierarchy of smaller modules would be the solution, as shown in figure 5. Each module is represented in the G2 development environment as a knowledge base (KB). Each KB represents an ontology of specified knowledge. An ontology is important for knowledge-based system development because it can serve as a software specification, similar to the function of a software architecture. Like a software architecture, an ontology provides guidance to the development process. The former provides guidance to the development process by specifying the interdependencies that deal with stages or aspects of a problem-solving process. By contrast to software architecture, however, an ontology involves not only the stages of a process, but also the taxonomy of knowledge types. The two aspects are referred to as task-specific and domain-specific architectures (Mark et al., 1995). The modular knowledge base design approach supports objected-oriented design principles, increases productivity, encourages code reuse and scalability, and improves maintainability.

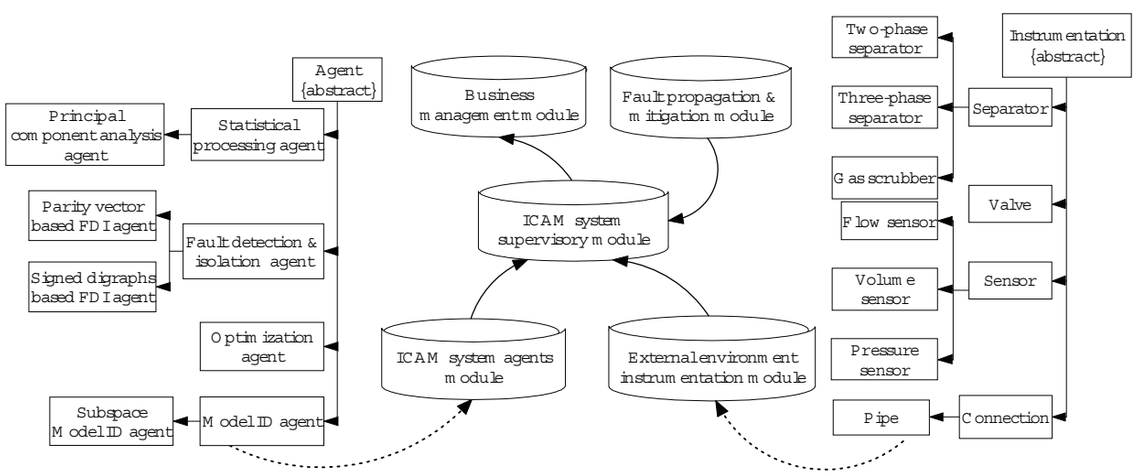


Fig. 5. Knowledge representation structure in the ICAM supervisory agent

The core (private layer) of the ICAM system supervisory agent has five modular KBs, which are organized as a module hierarchy. Basic knowledge about the ICAM system elements is represented in three lower level knowledge bases (i.e., ICAM system agents' KB, external environment instrumentation KB, and fault propagation and mitigation KB). The first knowledge base organizes the different conceptual agents of the ICAM system (e.g. fault detection and isolation agents, optimization agent, etc ...). In contrast, the second knowledge base maps the external environment physical instrumentation (e.g., valves, sensors, and other chemical process equipment) into its class hierarchy. Instrumentation and process faults and their mitigating actions are represented as classes in the third KB. Each basic element (i.e., object) in these knowledge bases has properties to represent its physical or conceptual characteristics; and has methods to represent its behavior. Elements are further organized as a class hierarchy to exploit object-oriented standards such as abstraction, inheritance, and information hiding and encapsulation.

The ICAM system supervisory knowledge base merges the knowledge from the lower level modules into a three-layer knowledge base, where each layer represents a subsystem of connected objects (i.e., classes). The first layer (i.e., the ICAM system structure layer) assembles the conceptual structure of the ICAM system from the agent class hierarchy of the lower level knowledge base. This layer is responsible for managing the internal behavior

of the ICAM system. Fault propagation and mitigating actions are assembled into object trees, and mapped into the second layer, which manages the external environment during abnormal situations. In fact, it isolates instrumentation faults, and presents their propagation maps and their appropriate migrating actions to process operators. The third layer (i.e., process topology layer) represents the external process topology, where different process instrumentation objects are used from the instrumentation knowledge base module. Other knowledge bases can be added to represent other types of knowledge such as the enterprise business management module.

A G2 rule-based system maps out a multi-threaded path of execution, which is potentially different each time the rule is invoked. For this reason, rule-based systems are often more complex, harder to test, debug, and maintain, and less efficient than procedure-based systems based on methods. Thus, rules should be used for specific purposes such as general event detection and event detection based on data driven processing and forward chaining (Gen, 2005). Since the ICAM system knowledge is multi-faceted and complex, its knowledge processing structure should be also distributed and organized according to the class and/or module hierarchy of the supervisory agent. For example, generic rules for event detection of a specific reactive agent can be organized in the class associated with that reactive agent. Rules can also be categorized to achieve certain functionality. For example, the fault propagation and mitigation schemes (i.e., cases) can be implemented as a rule category. This would narrow the scope of rules, where rules are only applied to their specified level in the class hierarchy and/or the module hierarchy. Consequently, rules invocation by forward chaining will be less prone to errors. The distribution of knowledge representation and processing would meet most of the software requirements. This would pave the way for managing complex process plants by dividing them into sub-processes that can be managed by a separate ICAM system. A universal supervisor can then manage the whole hierarchy of sub-processes efficiently.

6.2 Communication requirements for the ICAM system

It is crucial to design the agent structure to achieve specific autonomy requirements in terms of an overlapping scheme for communication and computation along with ease of prototyping and deployment. The design of the system's middleware structure, which acts as an integration model showing the types of connectivity between the different agents, is also important for achieving autonomy. Middleware is connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network. Middleware can take on the following different models (Aldred et al., 2005; Bernstein, 1996; Emmerich, 2000; Fox et al., 2005; Pinus, 2004):

1. Transactional middleware, which permits client applications to request several services within a transaction from a server application.
2. Procedural middleware, which enables the logic of an application to be distributed across the network, and can be executed by Remote Procedure Calls (RPCs).
3. Message-Oriented Middleware (MOM), which has become an increasingly popular solution for interoperability of heterogeneous applications. It provides generic interfaces that send and receive messages between applications through a central message server that takes charge of routing the messages.
4. Object/component middleware (e.g., CORBA, Java RMI, and Microsoft COM/DCOM technologies), which is a set of useful abstractions for building distributed systems. The communication model for this platform is based on a request/reply pattern.

5. High Performance Computing and Communication (HPCC) middleware, which is oriented toward the development of parallel computing hardware and parallel algorithms. The Message Passing Interface (MPI) communication model meets the autonomy and high performance requirements
6. Web Service-Oriented middleware, in which XML-documents (i.e., messages) are exchanged between systems using the simple object access protocol (SOAP). A SOAP message may include, for example, all necessary information for its secure transmission.

Having reviewed the different middleware technologies, it is our opinion that the high performance computing and communication MPI-based middleware meets the ICAM system requirements. The MPI communication library offers many communication modes and protocols, giving system designers the freedom and flexibility to implement their communication specifications and protocols. The MPI library specifies synchronous, buffered, ready, and nonblocking communication modes. In the synchronous mode, communicating processes are blocked till a message transfer operation is completed. However, the non-blocking mode does not block the communicating processes, which allows more flexible implementation in terms of communication/computation overlap. Buffered mode gives designers more manageability over communication buffers, whereas the ready mode guarantees correct message sending operation if a matching receiving operation is posted.

Among pre-specified MPI protocols, designers can choose from several protocols such as the Eager, the Rendezvous, and the One-sided protocols for implementation. The Eager protocol can be used to send messages assuming that the destination can store them. This protocol has minimal startup overhead and is used to implement low latency message passing for smaller messages. The Eager protocol has advantages in terms of programming simplicity and reduction of synchronization delays. However, it requires significant buffering, additional buffer copies, and more CPU involvement at the destination. In contrast, the Rendezvous protocol negotiates the buffer availability at the receiver side before the message is actually transferred. This protocol is used for transferring large messages when the sender is not sure whether the receiver actually has the buffer space to hold the entire message. This protocol is safe and robust, and may save in memory. However, it requires more complex programming and may introduce synchronization delays. The One-sided protocol (i.e., remote memory access (RMA) protocol) moves data from one process to another with a single routine that specifies both where the data are coming from and where they are going. Communicating agents using this protocol must have a designated public memory (i.e., blackboard), which can be remotely accessed. This protocol has nearly the best performance compared to others in terms of synchronization delays; however, it requires a very careful synchronization planning process (Gropp et al., 1999).

Having described the communication design options available in the MPI library and according to the high performance MPI recommendations (Gropp & Lusk, n.d.), it is our opinion that the ICAM system communications should meet the following requirements:

- In order to avoid deadlocks, synchronization time, and serialization problems, the non-blocking communication mode should be used.
- To address the message size and scalability issues, the Rendezvous protocol would be the perfect candidate among the other MPI protocols.
- The problem of buffer contention and achieving fairness in message passing can be resolved by having large communication buffers.

- The one-sided protocol can also be implemented to augment ICAM system communication performance by enabling agents to have their own private blackboards, as was discussed in the previous section.

6.3 Reactive agent software implementation requirements

In order to reconcile efficient computation with ease of prototyping requirements, the ICAM system is deployed as a distributed interconnection of reactive MATLAB computational agents, which runs on a network of several Windows XP workstations. Distributed MATLAB sessions exchange messages by using our newly developed MPI communication protocol. Exchanged messages have two roles: a control role to achieve internal coordination with other agents, and a numerical data processing role to achieve the best interaction with the external environment (e.g., offshore oil processing rigs) (Sayda & Taylor, 2007b).

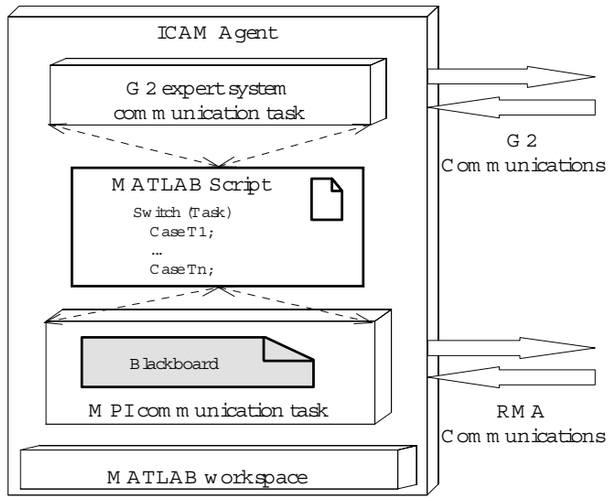


Fig. 6. ICAM system reactive agent deployment structure

Figure 6 shows the structure of a general reactive agent of the ICAM system. The general agent is implemented as a MATLAB m-script, which runs two communication tasks and a computational one. The computational task represents the agent's main functionality (e.g., model ID, fault detection and isolation, etc.). The first communication task is an MPI remote memory access (RMA) protocol, which provides the basic buffered messaging capabilities with minimum overhead (refer to the next section for more details). Furthermore, a public memory window is embedded in each reactive agent for remote access by other agents. The memory window will act as a blackboard for direct transfer of complex numerical data structures among agents. This design decision was made after investigating the advanced features of the newest MPI 2.0 library (Gropp et al., 1999), and to meet the blackboard functionality described in the behavioral model of the ICAM system.

The second communication task manages the connection with the main system supervisor (implemented as a G2 expert system). The general MATLAB template for reactive agents is built as a hierarchical finite state machine (FSM) module, which consists of two FSM layers. The first FSM is responsible for processing the ICAM system events received from the supervisory agent (i.e., the operating system of the ICAM system). The second FSM implements the specific computational functionality of the agent (e.g., FDI, model ID etc.). Further FSM layers can be added depending on the complexity of the reactive agent. Figure 7 illustrates the reactive agent implementation, which first starts its main MATLAB script and

its associated graphical user interface (GUI). After the ICAM agent is instantiated and its buffers are initialized, the MPI communication environment and the G2 expert system link are initialized. The agent's specified computational task is started.

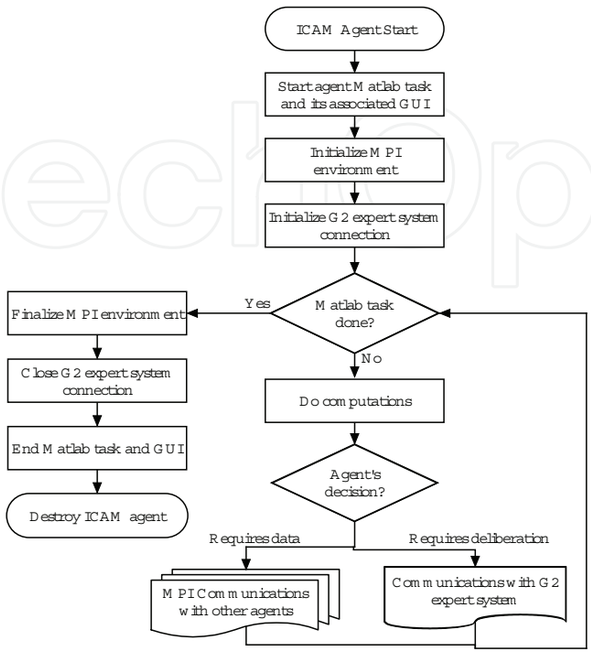


Fig. 7. Reactive ICAM agent implementation flow chart

Once the computations are done, the communication tasks are executed based on the agent's internal state and decisions. If the agent decides that it requires further deliberation about its internal state or its response to the external environment, then messages are exchanged with the ICAM system supervisor (i.e., the G2 expert system). On the other hand, if the agent requires more data for better awareness of the external environment, then it would exchange messages with other agents through its MPI link. if the computational task is done, the task is ended; if not, the computation loop continues to execute. The MPI environment is finalized, and the G2 expert system link is disconnected when the ICAM system shuts down. The proposed agent structure paves the way to design and to rapid prototype any complex multi-agent system for many applications. This definitely enables system designers to implement any communication protocol in addition to exploiting the full power of the MATLAB simulation, computation, and development environment.

7. ICAM System performance verification and validation

A prototype has been developed in order to have the ICAM system requirements deployed in a real-world system. Figure 8 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 are 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 to make the appropriate decision depending on the plant situation.

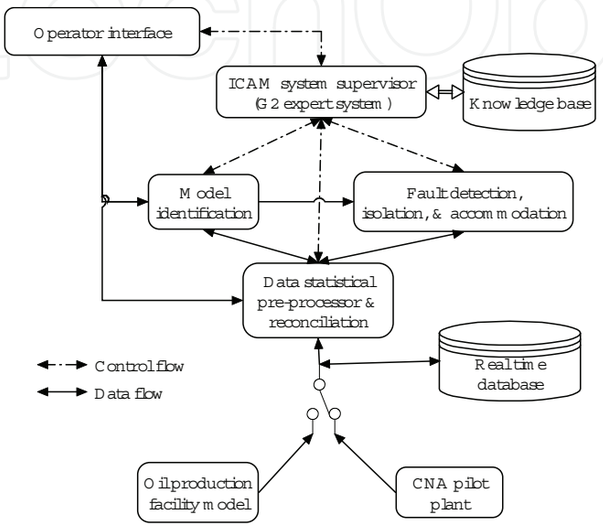


Fig. 8. ICAM system prototype

Real-time simulation experiments were designed to analyze the performance of the ICAM system prototype in terms of its logical behavior and its response to the external environment dynamics. Five different simulation scenarios were applied in real-time experiments. Three scenarios showed successful behavior by simulating three different plant faults, in which the ICAM system had an adequate knowledge about. The other two scenarios simulate situations, where the ICAM system has no knowledge whatsoever about. This would reveal The ICAM system limitation. Due to limited space, the simulation results are not discussed, however, the simulation experiments conclusions are discussed in the following section.

8. Learned lessons & future directions

Designing an intelligent multi-agent system is a very challenging task, as all agents are distributed and semi-autonomous. We faced several design challenges which resulted in limited system capabilities. Some of these design challenges and the future recommendations for solving them are suggested in the following points:

- Although we proposed the hierarchical colored petri nets approach to design the internal logic of the ICAM system reactive agents in our development plan (Sayda & Taylor, 2006), we did design the agents' internal logic in an *ad hoc* manner. We faced some difficulties during the design stage of the ICAM system prototype, as more functionalities were added. For example, the ICAM system crashed during early simulation runs due to communication deadlocks, in which two agents were trying to send messages to each other simultaneously. The problem was solved by imposing conditions on communicating

agents to prevent such deadlocks. Future designs should use the colored petri net approach to verify the logical behavior of the ICAM system and its agents in different scenarios.

- Computation/communication coordination was another design problem, in which computation and communication code blocks were not ordered correctly in the agent code. For example, we combined the process model estimation (computation task) and sending the estimated model to other agents (communication task) into one task in the model ID agent, which proved to be a design flaw. Model estimation took a long time (i.e., over one minute), during which other agents were locked waiting for the estimated model due to synchronization failure. The problem was solved by separating the one functionality into two separate computation and communication functionalities (i.e., separate agent states) and modifying other agents accordingly. Although some design flaws had to be corrected, the ICAM system prototype acted as a set of distributed stochastic colored petri nets during real-time simulation. This implies that a careful agent design should be done along with a thorough system logical behavior analysis. Future design plans would take the stochastic nature of the system and time into account to guarantee robust performance.
- The industrial plant data characteristics also had a major impact on the ICAM system performance. For example, the ICAM system prototype is not robust against noisy data due to the design of the data differentiation-based steady state detection algorithm. Likewise, the FDIA algorithm is not robust to noise, which significantly affects the fault isolation task in moderate to high noisy data situation. We suggest embedding algorithms that are more robust to noise to cope with real-world industrial plants and their noisy measurements.
- Detection and isolation of fast dynamics faults (e.g., faulty gas pressure sensor) is another limitation of the ICAM system prototype. The outlier removal algorithm in the statistical processing agent treats fast dynamics faults as outliers, which changes the nature of processed data sent to the FDIA agent. Data filtering also may change the data characteristic, which may have an impact on the system performance. In addition, the system logical behavior was unpredictable and inconsistent in response to disturbances in process variables. So we suggest developing a better safety net, in which the knowledge of agents' limitations is embedded in the rule base of the supervisory agent. This allows the system to have a better reasoning ability and robust performance during undefined and unpredictable plant situations.
- The incorporation of domain knowledge would definitely improve the performance of the system. Such knowledge is represented by the topology of the industrial plant and its operation procedure in different situations such as startup, normal operation, and shutdown. This knowledge would be better utilized if a learning agent were embedded to deal with new situations in the plant and the internal behavior of the ICAM system itself.

As can be appreciated, those enhancements will require years of additional research and development.

9. References

Aldred, L., van der Aalst, W. M. P., Dumas, M. & ter Hofstede, A. H. M. (2005). On the notion of coupling in communication middleware, *International Symposium on Distributed Objects and Applications (DOA)*, Agia Napa, Cyprus.

Banks, S. B. & Lizza, C. S. (1991). Pilot's associate: a cooperative, knowledge-based system application, *IEEE Expert* 6(3): 18–29.

Bernstein, P. A. (1996). Middleware: A model for distributed services, *Communications of the ACM* 39(2): 86–97.

Cauvin, S. (2004a). CHEM-DSS : Advanced decision support system for chemical/petrochemical industry, *Fifteenth International Workshop on Principles of Diagnosis (DX'04)*, AAAI, Carcassonne, France.

Cauvin, S. (2004b). CHEM-DSS: Advanced decision support system for chemical/petrochemical manufacturing processes, *CHEM Project Annual Meeting*, http://www.chem-dss.org/, Lille, France.

Cochran, E. L., Miller, C. & Bullemer, P. (1996). Abnormal situation management in petrochemical plants: can a pilot's associate crack crude, *Proceedings of the 1996 IEEE National Aerospace and Electronics Conference, NAECON*, Vol. v2, IEEE, Piscataway, NJ, USA, Dayton, KY, USA, pp. 806–813.

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

Derriso, M. (2005). Intelligent vehicle health management for air force space systems, *ISHM/NASA session of the IEEE Sensors for Industry Conference*, IEEE/ISA, Houston, TX, USA.

Durfee, E., Lesser, V. R. & Corkill, D. D. (1989). Trends in cooperative distributed problem solving, *IEEE Transactions on Knowledge and Data Engineering* 1(1): 63–83.

Durfee, E. & Montgomery, T. (1989). MICE: A flexible test bed for intelligent coordination experiments, *Proceedings of the 9th workshop on distributed AI*, Rosario, Washington.

Durfee, E. & Montgomery, T. (1991). Coordination as distributed search in a hierarchical behavior space, *IEEE Transactions on Systems, Man, and Cybernetics* 21(6): 1363–1378.

Emmerich, W. (2000). Software engineering and middleware: a roadmap, *Proc. of the Conference on The Future of Software Engineering*, IEEE Computer Society, Limerick, Ireland, pp. 117–129.

Figueroa, F. (2005). Integrated health with networked intelligent elements (IHNIE) prototype, *ISHM/NASA session of the IEEE Sensors for Industry Conference*, IEEE/ISA, Houston, TX, USA.

Figueroa, F., Holland, R. & Schmalzel, J. (2006). ISHM implementation for constellation systems, *42nd AIAA/ASME/SAE/ASEE Joing Propulsion Conference and Exhibit*, Sacramento, CA, USA.

Figueroa, F., Holland, R., Schmalzel, J. & Duncavage, D. (2006). Integrated system health management (ISHM): systematic capability implementation, *Proceedings of the 2006 IEEE Sensors Applications Symposium*, pp. 202–206.

Fox, G. C., Aktas, M. S., Aydin, G., Gadgil, H., Pallickara, S., Pierce, M. E. & Sayar, A. (2005). Algorithms and the grid, *Conference on Scientific Computing*, Vysoke Tatry, Podbanske.

Garcia-Beltran, C., Exel, M. & Gentil, S. (2003). An interactive toot for causal graph modeling for supervision purposes, *In Proc. IEEE International Symposium on Intelligent Control (ISIC)*, Huston, Texas, pp. 866– 871.

Garcia-Galan, C. (2005). Integrated system health management for exploration mission systems, *ISHM/NASA session of the IEEE Sensors for Industry Conference*, IEEE/ISA, Houston, TX, USA.

Gen (2005). G2 *for Application Developers Reference Manual*, 8.0 edn.

Gropp, W. & Lusk, E. (n.d.). Tuning MPI applications for peak performance. www.mcs.anl.gov/Projects/mpi/tutorials/perf, Argonne National Laborary.

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

Jennings, N. R. & Mamdani, E. M. (1996). Using ARCHON to develop real–world DAI applications parts 1, 2, 3, *IEEE Expert* 11(6): 64–86.

Karsai, G., Biswas, G., Abdelwahed, S., Mahadevia, N., Keller, K. & Black, S. (2005). Intelligent component health management: An architecture for the integration of IVHM and adaptive control, *ISHM/NASA session of the IEEE Sensors for Industry Conference*, IEEE/ISA, Houston, TX, USA.

Kim, I. S. & Modarres, M. (1987). Application of goal tree-success tree model as the knowledge-base of operator advisory systems, *Nuclear Engineering and Design* 104: 67–81.

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

Larimore, W. (2005). *Multivariable System Identification Workshop*, University of New Brunswick, Fredericton, New Brunswick.

Laylabadi, M. & Taylor, J. H. (2006). ANDDR with novel gross error detection and smart tracking system, *12th IFAC Symposium on Information Control Problems in Manufacturing*, IFAC, Saint-Etienne, France.

Liao, S. H. (2004). Expert systems: Methodologies and applications, a decade review from 1995 to 2004, *Expert systems with applications* pp. 1–11.

Liebowitz, J. (1998). *The Handbook Of Applied Expert Systems*, CRC Press, Boca Raton, FL.

Mark, W., Dukes-Schlossberg, J. & Kerber, R. (1995). *Towards very large knowledge bases*, IOS Press/Ohmsha, msterdam/Tokyo, chapter Ontological commitment and domain specific architectures: Experience with comet and cosmos.

Matania1, R. (2005). Interoperability and integration oil and gas science and technologyof industrial software tools, *Oil and Gas Science and Technology* 60(4): 617–627.

Maul, W. A., Park, H., Schwabacher, M., Watson, M., Mackey, R., Fijany, A., Trevino, L. & Weir, J. (2005). Intelligent elements for the ISHM testbed and prototypes (ITP) project, *ISHM/NASA session of the IEEE Sensors for Industry Conference*, IEEE/ISA, Houston, TX, USA.

Miller, C. A. & Hannen, M. D. (1999). Rotorcraft pilot's associate: Design and evaluation of an intelligent user interface for cockpit information management, *Knowledge-Based Systems* 12(8): 443–456.

Moore, R. L. & Kramer, M. (1986). Expert systems in online process control, *Proceedings of the 3rd international conference on chemical process control*, Asilomar, California.

Mylaraswamy, D. (1996). DKIT*: a blackboard-based, distributed, multi-expert environment for abnormal situation management*, PhD thesis, Purdue University.

Mylaraswamy, D. & Venkatasubramanian, V. (1997). A hybrid framework for large scale process fault diagnosis, *Computers and Chemical Engineering* 21: S935–S940.

Mylopoulos, J., Kramer, B., Wang, H., Benjamin, M., Chou, Q. B. & Mensah, S. (1992). Expert system applications in process control, *In Proc. of the International Syposium on Artificial Intelligence in Materials Processing Applications*, Edmonton, Alberta, Canada.

Newell, A. (1990). *Unified theories of cognition*, Harvard University Press, Cambridge, MA.

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

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

Omana, M. & Taylor, J. H. (2006). Enhanced sensor/actuator resolution and robustness analysis for FDI using the extended generalized parity vector technique, *Proc. of American Control Conference*, IEEE, Minneapolis, Minn., pp. 2560–2566.

Omana, M. & Taylor, J. H. (2007). Fault detection and isolation using the generalized parity vector technique in the absence of a mathematical model, *IEEE Conference on Control Applications (CCA)*, Singapore.

Pinus, H. (2004). Middleware: Past and present a comparison.
URL: *http://www.research.umbc.edu/ dgorin1/451/middleware/middleware.pdf*.

Sayda, A. F. & Taylor, J. H. (2006). An implementation plan for integrated control and asset management of petroleum production facilities, *IEEE International Symposium on Intelligent Control ISIC06*, IEEE, Munich, Germany, pp. 1212–1219.

Sayda, A. F. & Taylor, J. H. (2007a). An intelligent multi agent system for integrated control and asset management of petroleum production facilities, *In Proc. of The 17th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM)*, Philadelphia, USA, pp. 851–858.

Sayda, A. F. & Taylor, J. H. (2007b). Modeling and control of three-phase gravity separators in oil production facilities, *the American Control Conference (ACC)*, New York, NY.

Sayda, A. F. & Taylor, J. H. (2007c). Toward a practical multi-agent system for integrated control and asset management of petroleum production facilities, *IEEE International Symposium on Intelligent Control (ISIC)*, Singapore.

Sayda, A. F. & Taylor, J. H. (2008a). A multi-agent system for integrated control and asset management of petroleum production facilities - part 1: Prototype design and development, *accepted for the IEEE International Symposium on Intelligent Control (ISIC)*, San Antonio, Texas, USA.

Sayda, A. F. & Taylor, J. H. (2008b). A multi-agent system for integrated control and asset management of petroleum production facilities - part 2: Prototype design verification, *accepted for the IEEE International Symposium on Intelligent Control (ISIC)*, San Antonio, Texas, USA.

Sayda, A. F. & Taylor, J. H. (2008c). A multi-agent system for integrated control and asset management of petroleum production facilities - part 3: Performance analysis and system limitations, *accepted for the IEEE International Symposium on Intelligent Control (ISIC)*, San Antonio, Texas, USA.

Schmalzel, J., Figueroa, F., Morris, J., Mandayam, S. & Polikar, R. (2005). An architecture for intelligent systems based on smart sensors, *IEEE Transactions on Instrumentation and Measurement* 54(4): 1612–1616.

Sloman, A. (2001). Varieties of affect and the CogAff architecture schema, *proceedings of symposium on Emotions, Cognition, and Affective Computing at the AISB'01 convention*, York, UK.

Sloman, A. & Scheutz, M. (2002). Framework for comparing agent architectures, *Proceedings of the UK Workshop on Computational Intelligence*, Birmingham, UK.

Small, R. L. & Howard, C. W. (1991). A real-time approach to information management in a pilot's associate, *Proceedings of Digital Avionics Systems Conference*, IEEE/AIAA, pp. 440–445.

Smith, C., Gauthier, C. & Taylor, J. H. (2005). *Petroluem Applications of Wireless Sensors* (PAWS) *Workshop*, Cape Breton University, Sydney, Nova Scotia.

Taylor, J. H. (2004). Petroleum applications of wireless systems - UNB's control/information technology subproject, *submitted to Cape Breton University on 11 December 2003 and subsequently to ACOA on 21 September 2004*.

Taylor, J. H. & Laylabadi, M. (2006). A novel adaptive nonlinear dynamic data reconciliation and gross error detection method, *Proc. of IEEE Conference on Control Applications*, IEEE, Munich, Germany, pp. 1783–1788.

Taylor, J. H. & Omana, M. (2008). Fault detection, isolation and accommodation using the generalized parity vector technique, *submitted to the IFAC World Congress*, Seoul, Korea.

Taylor, J. H. & Sayda, A. F. (2005a). An intelligent architecture for integrated control and asset management for industrial processes, *Proc. IEEE International Symposium on Intelligent Control (ISIC05)*, Limassol, Cyprus, pp. 1397–1404.

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

Taylor, J. H. & Sayda, A. F. (2008). Prototype design of a multi-agent system for integrated control and asset management of petroleum production facilities, *accepted for the American Control Conference (ACC)*, Seattle, Washington.

Vedam, H. (1999). OP-AIDE*: an intelligent operator decision support system for diagnosis and assessment of abnormal situations in process plants.*, PhD thesis, Purdue University.

Vedam, H., Dash, S. & Venkatasubramanian, V. (1999). An intelligent operator decision support system for abnormal situation management., *Computers and Chemical Engineering* 23: S577–S580.

Venkatasubramanian, V. (2005). Prognostic and diagnostic monitoring of complex systems for product lifecycle management: Challenges and opportunities., *Computers and Chemical Engineering* 29: 1253–1263.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N. & Yin, K. (2003). A review of process fault detection and diagnosis part 1, 2, 3, *Computer & Chemical Engineering* 27(3): 293–346.

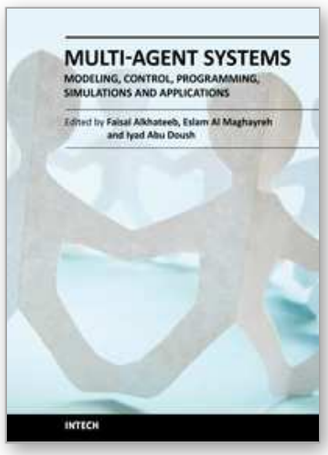
Wang, H. & Wang, C. (1996). APACS: A multi-agent system with repository support, *Knowledge-Based Systems* 9(5): 329–337.

Wang, H. & Wang, C. (1997). Intelligent agents in the nuclear industry, *Computer* 30(11): 28–34.

Wilikens, M. & Burton, C. J. (1996). FORMENTOR: Real-time operator advisory system for loss control. application to a petro-chemical plant, *International Journal of Industrial Ergonomics* 17: 351–366.

Wooldridge, M. J. (2002). *An introduction to multiagent systems*, Wiley, Chichester, England.

Worn, H. (2004). DIAMOND: Distributed multi-agent architecture for monitoring and diagnosis, *Production Planning and Control* 15: 189–200.



Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications

Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-174-9
Hard cover, 522 pages
Publisher InTech
Published online 01, April, 2011
Published in print edition April, 2011

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen