An Intelligent Architecture for Integrated Control and Asset Management for Industrial Processes

James H. Taylor and Atalla F. Sayda

Abstract— **Abnormal event management (AEM) in large manufacturing plants has evolved as a higher and increasingly vital function of process control. In this paper, an intelligent information management and control system is introduced. The different computational agents (i.e., modules) of the system are embodied in a three-layered cognitive hierarchy, which offers intelligent behavior at the system level, as well as at the level of specialized task agents. At the lower level, agents generate goal-seeking reactive behavior. Three different fault detection and isolation agents (i.e., three complementary techniques) are embedded to generate three different assessments and to enhance the fault isolation process. Other utility agents are also incorporated to address topics such as process model identification and optimization. At the middle layer, agents enable decision making, planning, and deliberative behavior. Two case-based reasoning agents are incorporated, the first manages the system in normal operation, while the other handles faulty process situations. A meta-management agent at the highest level monitors and coordinates other agents so as to make the whole system performance more robust and coherent.**

I. INTRODUCTION

Abnormal event management (AEM) in large process plants has evolved as a higher and increasingly vital function of process control. When an abnormal event occurs it may take considerable time to diagnose its causal origin, and to take the appropriate actions to bring the process back to a normal, safe operating state. This may have significant economic, safety, and environmental impact. Unfortunately, AEM is controlled manually in most process plants, which complicates the management and control of such plants. This can be attributed to several factors such as the size and complexity of modern process plants and increasingly massive information overload. The automation of AEM within an information and control infrastructure will reduce maintenance expenses, improve utilization and output of manufacturing equipment, enhance safety, and improve product quality. An integrated control and AEM system involves several subproblem areas including data reconciliation and fusion, fault detection, isolation, and accommodation (FDIA), process model identification and optimization, and supervisory control. The integration of these complementary features into an intelligent fault-tolerant control framework will define a new arena for research in this area [1], [2], [3], [4].

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`

Many research studies, which proposed different combinations of systems theoretic and artificial intelligence techniques to tackle the AEM problem, have delineated a set of required features [1]:

- integrating different problem solving paradigms, knowledge representation schemes and search techniques,
- maintaining global databases of process data and knowledge,
- reasoning about process operations without requiring accurate models,
- coping with data explosion and the need for effective compression and interpretation, and
- understanding, and hence representing, process behavior at different levels of detail.

These requirements are similar to those proposed for intelligent supervisory control systems. For example, a proposed system for producing metal-matrix composite materials incorporated a central database of process data and knowledge, process planning via case-based reasoning, online learning, automated process optimization and model identification, robust control algorithms – all under the direction of an expert system coordinator [5].

This paper addresses the integration of process automation and AEM in large process plants by introducing a design framework for building an intelligent information management and control system. The paper is organized as follows: First, we review available conceptual models of complex intelligent system followed by a detailed structural description of the proposed system architecture. Then, we discuss the development process of the system and project status. Finally, we conclude with future research and development steps.

II. CONCEPTUAL MODEL OF THE SYSTEM

We propose to use a combination of top-down and bottom-up approaches for modeling and developing an *intelligent control and asset management system* (ICAM system). The top-down approach deals with high level abstractions and conceptual tools, which facilitate capturing and modeling the structure and the behavior of the system being developed. Bottom-up modeling refers to developing scenarios that show in detail how the intelligent system should interact with users and complex external environments. Top-down modeling is the primary focus of this paper, so that the system architecture can be well explained and motivated.

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 [6] 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 [7], [8].

Newell [9] 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 a theory unification to model complex intelligent systems. Furthermore, this allows model (knowledge) reuse and helps create complete agents opening the way to applications. Soar and ACT-R, which are two of the most widely used cognitive architectures, represent Newell's approach and support most of the cognitive mechanisms [10]. These architectures are based on different conceptual origins: Soar arose from an artificial intelligence (AI) tradition, and ACT-R arose out of a more experimental psychology tradition. The performance of both 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 [11]. 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 [12]. Each agent contains processes for behavior generation, world modeling, sensory processing, and value judgment together with a knowledge database, as shown in figure 1. 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 [13].

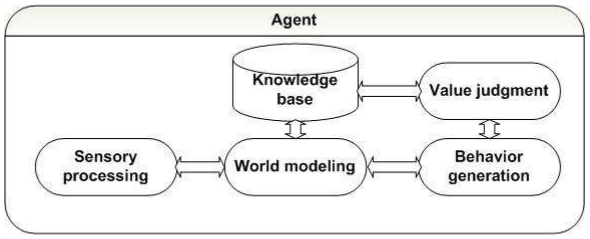

Fig. 1. Agent architecture

Sloman [14] 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 2, 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 [15]. 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.

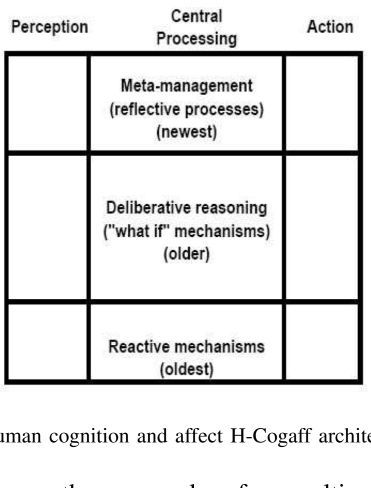
Perception
Central Processing
Action
Meta-management (reflective processes) (newest)
Deliberative reasoning ("what if" mechanisms) (older)
Reactive mechanisms (oldest)

Fig. 2. Human cognition and affect H-Cogaff architecture [14]

MAGIC is another example of a multi-agent system realization of an intelligent fault diagnosis system developed by a joint venture of several European universities and companies [16]. The system aims at developing a 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. The

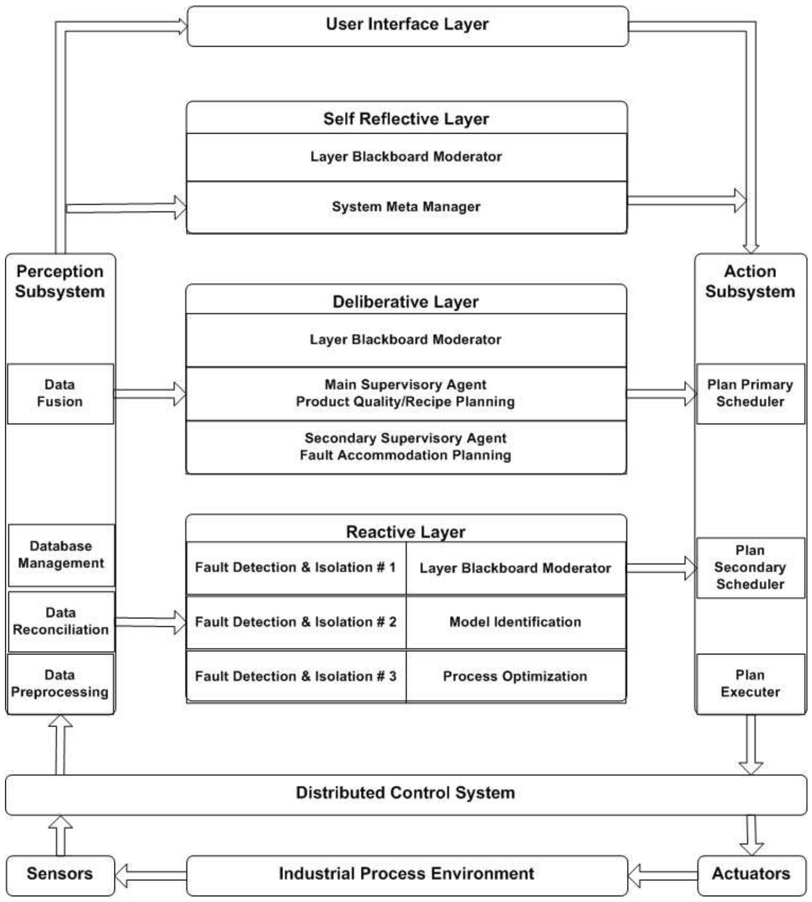
User Interface Layer
Self Reflective Layer
Layer Blackboard Moderator
System Meta Manager
Perception Subsystem
Data Fusion
Database Management
Data Reconciliation
Data Preprocessing
Deliberative Layer
Layer Blackboard Moderator
Main Supervisory Agent Product Quality/Recipe Planning
Secondary Supervisory Agent Fault Accommodation Planning
Reactive Layer
Fault Detection & Isolation # 1
Layer Blackboard Moderator
Fault Detection & Isolation # 2
Model Identification
Fault Detection & Isolation # 3
Process Optimization
Action Subsystem
Plan Primary Scheduler
Plan Secondary Scheduler
Plan Executer
Distributed Control System
Sensors
Industrial Process Environment
Actuators

Fig. 3. ICAM system architecture

distributed architecture for MAGIC is based on a Multi-Agents/Multi-Level (MAML) 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.

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 process plants. The architecture of the system and its functional modules will be discussed in subsequent sections.

III. System functional description and architecture

Figure 3 illustrates the proposed architecture of the system, which consists of four information processing layers and three vertical subsystems, namely, perception, central processing, and action. The horizontal layers above the distributed control system (DCS) contain 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 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.

The system layers interact with each other by means of bottom-up activation and top-down execution. Bottom-up activation occurs when a lower layer passes control to a higher layer because it is not competent to deal with the current situation. Top-down execution occurs when a higher-level agent makes use of the functionalities provided in a lower layer to achieve one of its goals. 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. 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. In the remainder of the this section, the functionalities of the agents in each layer will be discussed.

A. The Perception Subsystem

In order to tackle the problem of data explosion in modern complex process plant, the perceptual subsystem will process data in a hierarchical manner, and categorize it into different levels of abstraction. The data stream is processed serially by different agents, where the first agent function is data acquisition and pre-processing. Gross discrepancies such as outliers and missing data are detected and removed by this agent. The data stream is then exposed to further statistical processing to estimate variances and detect changes in steady state. Such statistical information is communicated to the central processing subsystem to permit it to adapt to new situations. The next agent then reconciles process data in accordance with steady state conservation laws (e.g., material balance). The data is then archived in a database by the database management system. The last agent in the perceptual subsystem, the data fusion agent, aggregates the data to optimally determine operation critical variables. This will help the planning layer assess the situation of the external environment and to make appropriate decisions.

B. The Reactive Layer

Agents in this layer provide a direct response to events that occur in the environment. When an abnormal event occurs, several fault detection and isolation (FDI) agents work concurrently and complimentarily to generate different assessments. The integration of several FDI agents in the system will result in a better performance, as suggested by many FDI survey papers [1], [17], [18]. FDI basically consists of two tasks, as shown in figure 4. The first task is fault detection, which indicates that something is going wrong in the plant. The determination of the exact location of the failure is the fault isolation task. Three different FDI techniques are being evaluated, namely, a directional parity vector model-based FDI technique, a fuzzy signed directed graph (SDG) model-based FDI technique, and a neuro-fuzzy data history-based FDI technique. These approaches are complementary in that they are based on entirely different world views, namely an analytic model, a cause/effect net and heuristic reasoning.

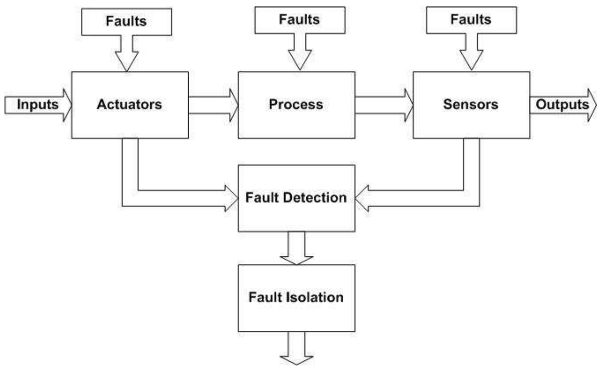

Fig. 4. Fault detection and isolation (FDI) scheme

The first FDI approach exploits the concept of generalized parity space (GPS) to generate a set of directional residuals, from which process faults can be determined. When a fault occurs, it will result in an activity of the parity vector along certain directions or in certain subspaces. Therefore the fault isolation task involves determining which predefined direction the parity vector is most nearly aligned with. The GPS concept was developed using the stable coprime factorization framework [19], where any $n \times m$ proper rational transfer function matrix $P(s)$ can be expressed in terms of stable left coprime factors $\tilde{N}(s), \tilde{D}(s)$, desired control inputs u_d, and the sensor outputs y as follows:

$$P(s) = \tilde{D}^{-1}(s)\tilde{N}(s) = \frac{y(s)}{u_d(s)} \tag{1}$$

which implies that

$$\tilde{D}(s)y(s) - \tilde{N}(s)u_d(s) = 0 \tag{2}$$

Under ideal conditions, when the plant is linear, noise and fault free, equation 2 holds. However, when a fault happens, this equation is violated showing inconsistency between actuator inputs and sensor outputs with respect to the fault-free model. Hence a generalized parity vector $p(s)$ can be defined as

$$p(s) = J(s)[\tilde{D}(s)y(s) - \tilde{N}(s)u_d(s)] \tag{3}$$

where $J(s)$ is a transformation matrix that adds another degree of freedom to achieve the desired FDI response specifications. A systematic approach to calculate an optimal transformation matrix has been effectively developed, enhancing the FDI properties and the scope in terms of the number of faults that can be isolated [20]. Once the generalized parity vector is generated then its magnitude and direction are compared to a threshold and direction respectively to isolate process faults. New adaptive thresholding schemes are under development, to make the approach more robust to model inaccuracy and nonlinear effects. The fault isolation assessment is then sent as a text message to the deliberative layer for further processing.

A new fuzzy signed digraph (SDG) model-based FDI technique is another approach being evaluated. Signed digraphs, which has been widely used to model the cause/effect behavior of process plants, consist of nodes representing the process variables (and parameters) and signed directed arcs representing the cause/effect relationship between these variables. Nodes assume values of $(0), (+), (-)$ representing nominal, above nominal, and below nominal values respectively, whereas arc signs of $(+1), (-1)$ indicate the values of the cause/effect change in the same or opposite direction. If a fault happens, process variables deviate resulting in a set of symptoms, which constitutes the pattern of this fault. In order to decrease the execution time of a conventional SDG based FDI algorithm, an offline fault diagnosis rule base will be developed from the SDG process model, as suggested by Kramer [21]. The use of a fuzzy representation of real-valued functions in the rule base will reduce the granularity of the qualitative process model, and will thus improve discrimination and decrease the generation of spurious alarms [22]. Further investigation is required to decide which inference technique will be used (e.g., backward chaining, forward chaining, fuzzy inference). This agent will send its fault isolation assessment to the higher layer in case of process failure.

The third FDI project involves extending the adaptive neuro-fuzzy inference system (ANFIS) methodology. ANFIS is a data driven modeling approach that combines the reasoning capability of fuzzy logic and the learning capability of neural networks. System knowledge is represented by rules and the membership of each of the input signals are estimated using training data and a neural network model. This step introduces nonlinearity in the estimated weights for all the postulated rules. For each fuzzy rule, the output is computed using a linear model of the input signals. The strength of this approach lies in its ability to use prior knowledge, and to update membership functions that provide a better model for the desired output. This makes the approach suitable for dealing with nonlinear processes [23]. In order to evaluate this approach for fault detection and isolation, we are considering with two possibilities: The first is to use a two-stage FDI scheme, where the nonlinear process is modeled using ANFIS and then a fuzzy inference system isolates process faults. The other possibility would be a single ANFIS stage, which is trained to isolate faults directly by means of different faulty process training data sets.

A model identification agent is incorporated in the reactive layer, in order to improve the knowledge available to the FDI agents about the external environment (i.e., the plant), This agent will exploit an off-the-shelf model identification package to produce a multi-variable model, which will predict changes in process variables to estimate new process parameters (learning task), enhance the fault isolation task, and compensate for faulty sensor signals (estimation task). The different agent tasks will be decided by the deliberative layer, depending on the situation. For instance, if the operating point of the plant changes to meet new required product quality, the deliberative layer will use the modeling agent to estimate the new process model parameters for further processing.

An optimization agent will be embedded in the reactive layer to make the best use of available equipment and raw materials. The agent will receive product quality plans and process operation constraints from the deliberative layer, then the agent will formulate a new optimization problem to solve and come up with the optimal raw material recipe to meet the new product quality. The new recipe is then sent to the DCS system in the form of set-points and parameters for further execution. The Optimizer may play the same role for faulty process situations, whenever possible.

The integration of different FDI agent assessments in a collaborative problem-solving framework, and the interaction between the different agents in the architecture, necessitates the use of a mechanism to achieve such goals. One approach would be a direct interaction between the system agents according to their data flow requirements. Direct interaction promotes the use of private communication protocols. However, this approach is inflexible because it does not address the dynamic scalability of the system in terms of adding new agents or changing the internal architecture of any of the system agents. Another approach is to use an indirect and anonymous communication among agents via an an intermediary such as a blackboard repository [24]. A blackboard agent will be embedded in the reactive layer to manage the interaction and communication among its agents and the higher layers in the architecture to achieve the utmost flexibility. The agent consists of the blackboard itself, which is a global data repository containing input data, partial and complete solutions, plans, and other data organized in a hierarchy to address the different levels of information abstractions in the architecture. A control mechanism, will make runtime decisions about accessing the data in the blackboard. The blackboard agent allows other agents to deposit their assessments, and notify them if some useful information are available or not. This would meet the requirements of concurrency and autonomy for high system performance.

C. The Deliberative Layer

Proactive behavior is achieved in the system in its deliberative layer, which is responsible for governing the system's actions in normal and faulty circumstances. Planning in this layer will not attempt to work in a vacuum. Rather, it will employ a library of pre-specified plans and a problem solving mechanism. There are several problem solving and inference paradigms that may be embedded in this layer of the architecture, such as rule-based reasoning, model-based reasoning, and case-based reasoning (CBR). Case-based reasoning provides a wide range of advantages over other paradigms. For instance, CBR can quickly propose solutions to problems that are not well defined, avoiding the time necessary to derive those answers from scratch, thus more easily meeting real-time requirements. CBR also helps process operators to focus their reasoning on important parts of a problem by pointing out what features of a problem are the crucial ones. In fact, CBR may warn of the potential for problems that have occurred in the past, alerting the operator to take actions to avoid repeating past mistakes [25].

CBR suggests a model of reasoning that incorporates problem solving, understanding, and learning, and integrates it all with episodic memory processes. It actually solves new problems by adapting previously successful solutions to similar problems. To employ all these capabilities efficiently in problem solving, a CBR system typically consists of the following reasoning modules, as depicted in figure 5:

1) Indexing: The case indices are crucial features for characterizing an event and determining how cases are stored in the case memory. The purpose of indexing is to allow a cased-based reasoner to retrieve one or more cases that are similar to a new problem.
2) Retrieval: The indices of a new problem are used to retrieve similar cases from the case memory. Efficient

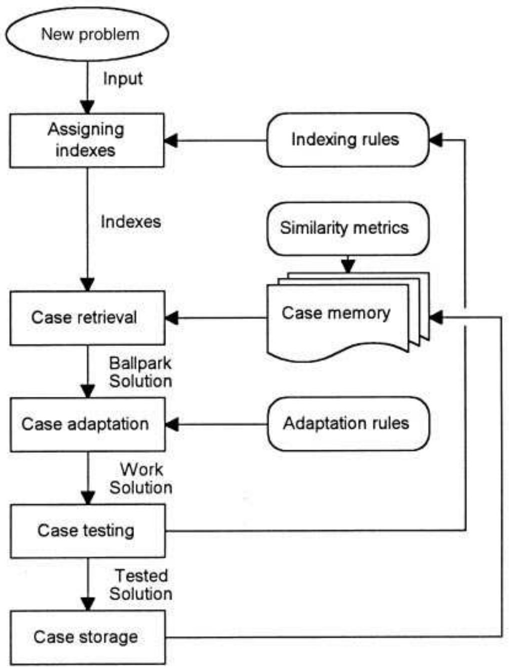

Fig. 5. Scheme of case-based reasoning systems

case retrieval is especially crucial in time-critical situations and when the case memory is large.

3) Adaptation: Since the retrieved cases may not exactly solve the new problem, the solutions from these cases need to be modified and adapted to the new problem.
4) Test: The proposed solution will be tried out to see if it really solves the problem. In the process industry, there are two ways of trying out a solution: in-plant implementation and simulation. For safety reasons, simulation is usually preferred if simulation models are available.
5) Storage: If the new problem is considered to be conceptually different from the existing cases, a new case needs to be created. The new case needs to be properly indexed with the proven explanation and solution as its content.

The deliberative layer supervises the system through two CBR agents. The first agent is the main supervisor, which manages the system during normal operation circumstances. The agent's case library contains product quality profiles, their pre-specified raw material recipes, and the associated process operating conditions. When a certain product specification is required, the main supervisory agent retrieves a set of cases that best match the required attributes and quality specifications. If the matching process is successful, the plan is sent to the action subsystem for execution. If not, the closest matching case is chosen and adapted by using the model-based optimization, in which the main supervisory agent collaborates with modeling, simulation and optimization agents to generate the optimal recipe and operating conditions (e.g., pressure and temperature). The plan is sent to the user interface layer for further modifications by process operators if needed. Once the plan has been approved then it is sent to the action subsystem for execution. The actual quality specifications are monitored by the main supervisory agent, which will add the plan to its "good" case library should the actual and desired specifications match, or to the "bad" repository if they do not. This behavioral paradigm was central to the intelligent processing architecture proposed in [5].

The other CBR agent acts as a backup supervisory agent to manage the system in case of faulty situations. Pre-computed fault accommodation plans are stored in this agent's case library. These plans consist of schemes for sensor/actuator reconfiguration and controller tuning/restructuring, as well as fault propagation scenarios and recommended predictive maintenance procedures. When a process fault happens, the backup agent receives fault assessments for the different FDI agents in the reactive layer. Based on such assessments, the agent retrieves the most closely matching case from its library. Consequently, it alarms the user interface agent about the fault, its possible causes, and recommended mitigating actions for operator feedback and approval. The backup supervisory agent may interfere directly in critical situations to prevent the system performance from deteriorating excessively and to keep it in an acceptable state. Collaboration with the main supervisor may also occur to preserve the product quality at an acceptable level, if possible. This collaboration takes place through a blackboard agent, which also manages the interaction between the deliberative layer and the others.

D. The Meta-management Layer

This self reflective layer provides the ability to monitor, evaluate, and control other agents in the architecture. For example, the deliberative layer is partly driven by decisions made by the reactive layer and perception subsystem, so it may unexpectedly acquire inconsistent information or goals. The same situation may occur in the action subsystem, which may not be able to meet the plan time frames sent by the deliberative layer. The meta-management agent can notice and categorize such situations, and perhaps through deliberation or observation over an extended time period develop a strategy to deal with these situations. Furthermore, the meta-management agent coordinates other agents so as to make the whole system performance more robust and coherent. It determines when other agents have completed their work, what agent to invoke next, and assesses credibility of each agent's behavior by monitoring their internal states.

E. The User Interface Layer

Process operators can interact with the system through its user interface layer, which works concurrently at the top of the architecture. The user interface layer receives different types of information from the different layers and subsystems, namely:

1) faulty components and their possible causes based on the different FDI agents' assessments,

2) fault propagation scenarios based on the reasoning of the SDG based FDI agent,
3) system recommendations in faulty situations such as instructions for control loop restructuring/tuning, predictive maintenance plans, and other mitigating measures,
4) product quality specifications and associated optimal raw material recipes, and
5) internal system diagnostics and other utility tasks such as process modeling and intelligent data trend monitoring facilities.

Its most important obligations are to present process-critical information in a timely manner, and prevent data- and work-overload for the operator.

F. The Action Subsystem

Plans which are sent by the deliberative layer are executed by the action subsystem. The action subsystem consists of hierarchically organized scheduling and execution agents. The main scheduling agent decomposes main plans into sub-plans that have shorter time frames. This results in better execution performance by alleviating the excessive computational burden on the main scheduling agent. The sub-plans are further decomposed by a secondary scheduling agent to simpler tasks in accord with the sub-processes in the plant. Finally, the subtasks are performed by their corresponding agents and the task outcomes are communicated to the DCS for final execution.

IV. System behavioral formalism

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 a certain internal/external event occurs. 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. To address this issue, A coherent coordination scheme has been developed based on the concept of behavioral hierarchy [26]. Petri nets constitute a graphical and mathematical modeling tool for describing and studying systems with such critical characteristics [27].

A Petri net is a particular kind of directed graph, together with an initial state called the initial marking. The underlying graph of a Petri net is a directed, weighted, bipartite graph consisting of two kinds of nodes, called places and transitions, where arcs are either from a place to a transition or from a transition to a place. In a graphical representation, places are drawn as circles, transitions as bars or boxes. Arcs are labeled with their weights (positive integers), where a *k*-weighted arc can be represented by the set of *k* parallel arcs. A marking (state) assigns to each place a nonnegative integer, which represents the number of tokens in that place. Tokens in a certain place may indicate that a number of data items or resources are available. A transition without any input place is called a source transition, and one without any output place is called a sink transition. Figure 6 illustrates the different parts of a Petri net for an arbitrary concurrent system. The parallel or concurrent activities, which are represented by transitions t_2 and t_3, begin at the firing of transition t_1 and end with the firing of transition t_4. Each place in the net has one incoming arc and one outgoing arc. Tokens in places P_1 and P_2 represent the initial state of the net, and indicate that there is one available resource for each place. If this is sufficient the concurrent activities can proceed after t_1 fires.

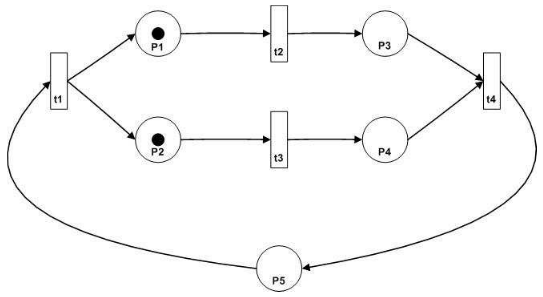

Fig. 6. Scheme of an arbitrary Petri net

A major strength of Petri nets is their support for analysis of critical properties and problems associated with concurrent systems. Two types of properties can be studied with a Petri net model; those which depend on the initial marking, and those which are independent of the initial marking. The former type of properties is referred to as marking-dependant or behavioral properties such as teachability, liveness, fairness, and others. The second type is called structural properties, which depend on the topological structure of the Petri net, namely, controllability, conservativeness, consistency, and others. Since we are designing a real-time dynamic system, the concept of time becomes crucial for performance evaluation and scheduling when modeling such systems. The timed Petri net introduces time delays associated with transitions and/or places in their net models to address real-time performance. The timed Petri net concept will be used to model the behavior of the architecture of the intelligent AEM system.

V. Project status and future work

A joint venture between several Atlantic Canadian universities, the National Research Council of Canada, and local and national companies was established in order to advance wireless sensor technology in the oil and gas industries and to assess the feasibility of an intelligent control and asset management system built on a wireless sensor network. As part of this joint venture, and as the leader in developing the ICAM system, we have formed a task force of five graduate students at the University of New Brunswick to address the integration of control and asset management for a large process industry application. Three team members were assigned the task of developing, testing and evaluating the different proposed FDI techniques. The FDI agents development task has successfully met several major goals, such as quick detection and isolation, isolability, robustness and disturbance decoupling. The task of evaluating the different data processing techniques which will be incorporated in the perception subsystem is assigned to another of the team members, who is presently focussing on data preconditioning and reconciliation. A rigorous review of the available system architectures and their characteristics has been done so as to match them with proposed system requirements. Starting in January 2004, the project has progressed well. Further steps are planned to implement a successful ICAM system prototype, namely:

1) refinement of the FDI agents to address topics such as adaptability, explanation and reasoning capability, and to meet real-time requirements,
2) development of appropriate data pre-processing, reconciliation, and aggregation techniques associated with the perception subsystem,
3) consultation with industrial and automation partner companies to produce final specifications and documentation for the architectural level and execution platform in order to meet industry standards,
4) design and modeling of the internal system coordinator using the Petri net approach,
5) designing the two CBR agents and the pre-computed fault accommodation plans, and
6) modeling the pilot plant which will be used to validate the system performance.

We believe that the successful design and development of the proposed system will lay a corner stone in the area of complex intelligent system development, and will open the doors for other applications such as distributed power plant management.

VI. 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 the University College of Cape Breton (UCCB), the National Research Council (NRC) of Canada, and the College of the North Atlantic (CNA). The authors also acknowledge the support of Natural Sciences and Engineering Research Council of Canada (NSERC) for funding the second author's research.

References

[1] 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.

[2] 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.

[3] 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.

[4] H. E. Rauch, "Fault diagnosis and control reconfiguation," *IEEE Control Systems Magazine*, June 1994.

[5] J. H. Taylor, L. P. Harris, P. K. Houpt, H.-P. Wang, and E. S. Russell, "Intelligent processing of materials: Control of induction-coupled plasma deposition," in *Advanced Sensing, Modelling, and Control of Materials Processing*. Warrendale, PA: Ed. by E. F. Matthys and B. Kushner, TMS Publications, 1991.

[6] R. L. Moore and M. Kramer, "Expert systems in online process control," in *Proceedings of the 3rd international conference on chemical process control*, Asilomar, California, 1986.

[7] J. Liebowitz, *The Handbook Of Applied Expert Systems*. Boca Raton, FL: CRC Press, 1998.

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

[9] A. Newell, *Unified theories of cognition*. Cambridge, MA: Harvard University Press, 1990.

[10] F. E. Ritter, N. R. Shadbolt, D. Elliman, R. Young, F. Gobet, and G. D. Baxter, "Techniques for modeling human performance in synthetic environ-ments: A supplementary review," Human Systems Information Analysis Center (HSIAC), formerly known as the Crew System Ergonomics Information Analysis Center (CSERIAC), Wright-Patterson Air Force Base, OH, Tech. Rep., 2003.

[11] E. Durfee and T. Montgomery, "MICE: A flexible test bed for intelligent coordination experiments," in *Proceedings of the 9th workshop on distributed AI*, Rosario, Washington, September 1989.

[12] M. J. Wooldridge, *An introduction to multiagent systems*. Chichester, England: Wiley, 2002.

[13] N. R. Jennings and E. M. Mamdani, "Using ARCHON to develop real–world DAI applications parts 1, 2, 3," *IEEE Expert*, vol. 11, no. 6, pp. 64–86, 1996.

[14] A. Sloman and M. Scheutz, "Framework for comparing agent architectures," in *Proceedings of the UK Workshop on Computational Intelligence*, Birmingham, UK, September 2002.

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

[16] 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.

[17] J. Gertler, "Survey of model based failure detction and isolation in complex plants," *IEEE Conteol Systems Magazine*, December 1988.

[18] R. Isermann and P. Balle, "Trends in applications of model based fault detection and isolation and diagnosis of technical processes," *Control Engineering Practice*, vol. 5, no. 5, pp. 709–719, 1997.

[19] 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.

[20] 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.

[21] M. Kramer and B. Palowitch, "Rule based approach to fault diagnosis using the signed directed graph," *AIChE Journal*, vol. 33, no. 7, pp. 1067–1078, 1987.

[22] E. Tarifa and N. Scenna, "Fault diagnosis, directed graphs, and fuzzy logic," *Computer and Chemical Engineering*, vol. 21, pp. S649–S654, 1977.

[23] S. R. Jang, "ANFIS adaptive network based fuzzy inference system," *IEEE transaction on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[24] D. D. Corkill, "Collaborating Software: Blackboard and Multi-Agent Systems & the Future," in *Proceedings of the International Lisp Conference*, New York, New York, October 2003. [Online]. Available: http://mas.cs.umass.edu/paper/265

[25] D. B. Leake, *Case-based reasoning: experiences, lessons and future directions*. Menlo Park, California: AAAI Press/MIT Press, 1996.

[26] 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.

[27] T. Murata, "Petri nets: properties, analysis, and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.