Monitoring, Analysis and Diagnosis of Distributed Processes with Agent-Based Systems ⋆

Ali Çinar * **Sinem Perk** * **Fouad Teymour** * **Michael North** ** **Eric Tatara** ** **Mark Altaweel** **

* *Department of Chemical and Biological Engineering, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: perksin@iit.edu)*
** *Argonne National Laboratory, Argonne, IL 60439 USA (e-mail: north@anl.gov)*

Abstract: Multiagent systems provide a powerful framework for developing real-time process supervision and control systems for distributed and networked processes by automating adaptability and situation-dependent rearrangement of confidence to specific monitoring and diagnosis techniques. An agent-based framework for monitoring, analysis, diagnosis, and control with agent-based systems (MADCABS) is developed and tested by using detailed models of chemical reactor networks. MADCABS is composed of three main hierarchical layers, the physical communication layer, the supervision layer and the agent management layer. The supervision layer consists of agents and methods for data preprocessing, process monitoring, fault diagnosis, and control. The agent management layer conducts the assessment of agent performances to assign the priorities for selecting the most useful methods of process supervision for specific types of situations. The paper illustrates the operation of MADCABS for monitoring and fault detection.

Keywords: Multi-agent systems, process monitoring, fault detection, fault diagnosis, process supervision, process control, distributed systems, distributed artificial intelligence, autocatalytic reactions.

1. INTRODUCTION

Multi-layered and adaptive multiagent systems (MAS) provide a powerful framework for developing a new generation of real-time supervision and control systems for distributed and networked processes. The strategy, techniques and tools are being developed at IIT for monitoring, analysis, diagnosis, and control with agent-based systems (MADCABS) that automates knowledge extraction from data, analysis, and decision making. This distributed artificial intelligence framework is expected to enable the consideration of novel configurations for manufacturing such as distributed reactor networks to produce high-value-added specialty chemicals.

There are strong reasons for distributing the activities and intelligence in software for supervision of distributed process operations:

- The complex layout of a manufacturing process yields a problem that is physically distributed,
- The supervision problem is distributed and heterogeneous in functional terms,
- The complexity of the supervision problem dictates a local point of view that contributes to the development of system-wide decisions that may force reexamination of local decisions,
- The supervision system must be able to adapt to changes in the structure or environment of the supervised process or network.

Agents are capable of acting, communicating with other agents, perceiving their environment, and determining behavior to satisfy their objectives. They are endowed with autonomy and they possess resources. However, the MAS framework offers challenges as well: Agents have only partial information about their environments, they may act "selfishly" or initiate actions that may conflict with actions of other agents. This may lead to undesirable or harmful behavior in MADCABS or the supervised process, compromising its profitability and safety.

The nature of the supervision problem dictates the use of multiple layers of agents where lower-level agents perform local well-defined tasks such as information validation from sensors and higher-level agents perform more global tasks over wider regions of the supervised system. Several agents can be used to perform a specific task, each using different methods to enable not only decision by consensus-building but also to reduce the influence of the weaker methods over time. Intelligence and adaptation is provided both at agent and at system level.

MADCABS is composed of three main hierarchical layers, the *physical communication* layer, the *process supervision* layer and the *agent management* layer. The physical layer is where two-way information communication between the

⋆ This work is supported by the National Science Foundation CTS-0325378 of the ITR program.

process and MADCABS takes place. Process information, such as the flowchart of the process is mapped to MADCABS through the physical communication layer. The supervision layer consists of agents and methods for data preprocessing, process monitoring, fault diagnosis, and control. Data preprocessing agents filter the process data, check for outliers and missing data, and provide estimates for them. Monitoring agents detect deviations from normal operation and trigger the fault detection and diagnosis (FDD) agents. When abnormal process operation is validated, FDD is carried out using contribution plots, statistical methods, and process knowledge. Control agents range from simple local PI controllers to decentralized plant-wide grade transition agents. Some agents collaborate to help each other, while others work in a competing manner to satisfy a global objective. The performances of different agents and methods are evaluated in the topmost agent management layer. The agent management layer is responsible for selecting the best performing agents for process monitoring, fault detection and diagnosis, and process control for the current operating conditions. The assessment of agent performances guide the priorities assigned to select the most useful methods of process supervision for specific types of situations.

In this paper, MADCABS modules for monitoring and fault detection, and the information flow among them is discussed. The paper focuses on the architecture and functionality of MADCABS, the automated tools for assessing the success of its various functions, the redundancies in MADCABS, and the adaptation of MADCABS based on the current state of process operations. The communication and cooperation between different MADCABS modules is demonstrated with case studies using autocatalytic CSTR networks. The capabilities of MADCABS in detecting and diagnosing various types of faults are shown.

2. PROCESS MONITORING, FAULT DETECTION AND DIAGNOSIS

2.1 Statistical Process Monitoring Techniques

Multivariate statistical process monitoring (SPM) techniques are used in this study. Multivariate techniques that extract the correlation among variables based on principal component analysis (PCA) provide the basic tool for monitoring continuous processes (Kourti and MacGregor (1996); Cinar et al. (2007); Jackson (1980)). PCA is a multivariate projection method that extracts strong correlations among the variables in a data set, and based on that information defines a new orthogonal coordinate space where the coordinate axes are the highest variance directions. For chemical processes, where large highly-correlated process datasets need to be monitored, singularity problems may arise. PCA is well-suited for reducing the dimensionality of the data and capturing the essential information in the data.

For large processes that involve many processing units and many process variables with different correlation structures, a single PCA model for the whole process may not give sufficient explanation about the process behavior, may provide unreliable information based on many false and missed alarms, and may have difficulty localizing the source cause among so many variables when a fault is detected. Multiblock methods have been proposed in literature for large processes, where the process can be separated into meaningful process blocks, to increase the efficiency and interpretability of the statistical monitoring model. Algorithms to handle multiple data blocks include hierarchical PCA (HPCA) and consensus PCA(CPCA) (Qin et al. (2001); Wangen and Kowalski (1988); Westerhuis et al. (1998); Wold et al. (1996)). Multiblock algorithms enable monitoring of the process both locally and globally. The CPCA method is designed for comparing several blocks of descriptor variables measured on the same objects (Wold et al. (1996); Westerhuis et al. (1998)).

Dynamic PCA (DPCA) is an extension of conventional PCA to deal with multivariate process data that is correlated in time, using a time-lag shift method (Ku et al. (1995)). The flow of action, namely, monitoring, fault detection and diagnosis, is the same for all SPM methodologies independent of which monitoring algorithm is employed.

2.2 Fault Detection, Monitoring and Diagnosis Framework in MADCABS

MADCABS is written in Java, using Repast Simphony as the agent building platform (ROAD (2005)). Among the important features of Repast that MADCABS uses are its object oriented structure, scheduling tools and its built-in automated Monte Carlo simulation framework. Repast also allows users to change, add, delete agents in run time.

In Repast Simphony, a *context* is defined as a container where the agents reside. There are three contexts for monitoring, fault detection and diagnosis agents in MADCABS. The communications between different agents in these contexts are shown in Figure 1. Statistical models are built by the monitoring agents. The fault detection agents, which are the monitoring statistics, assign themselves to the fault detection organizer agents responsible for each subsystem that is monitored. A fault is flagged when a consensus is formed among different fault detection agents on the existence of a fault in the system. The fault flag triggers the diagnosis agent. Diagnosis agent uses information from the neighboring fault detection organizers, finds the most contributing process variables to the fault on the faulty subsystem, and investigates the potential reasons behind the fault.

Monitoring Agents. The monitoring agents are PCAStarter, DPCAStarter and MultiblockStarter agents and a monitoring organizer (Figure 2). After the system reaches steady state and a sufficient amount of normal operation data is available, monitoring starters are scheduled to form models. For all process units, a local statistical model is built using both PCA and DPCA. In the distributed framework, this is achieved by creating as many PCAStarters and DPCAStarters as the number of operating units. Each PCAStarter agent builds a separate PCA model since the data and the PC number that is used to build the models are different for each unit. DPCAStarters work identical to PCAStarters. There is only one MultiblockStarter for the whole process. In this case, the data blocks consist of data from each operating unit. The MultiblockStarter forms a single multiblock model, which

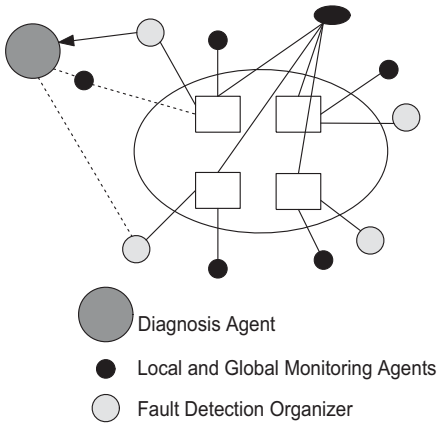

Fig. 1. Monitoring, fault detection and diagnosis agents in a four reactor network

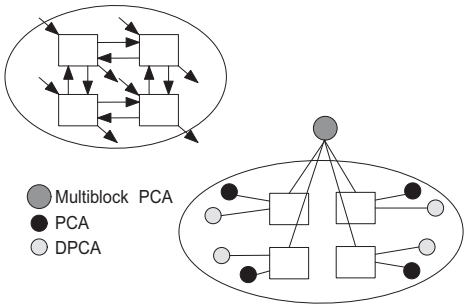

Fig. 2. Monitoring agents. The upper figure represents a four reactor network, which is simplified in the lower figure.

enables the monitoring of the process both locally and globally through block statistics and super statistics. Since there is only one model, the size retained in the model is the same for each block.

The starter agents are subclasses of MonitoringStarterParent class. Consequently, each of the starters extend some of the characteristics from the parent class and they also have class specific properties. The methods in each starter such as the buildModel and startProjection are common methods inherited from the parent, as well as the user-specified parameters. The number of principal components to be retained in the model is a superclass variable and it is overridden in each child class. Each model generates two monitoring statistics for each subsystem, a T^2 statistic and an SPE statistic. Three monitoring methods generate a total of six fault detection agents for each subsystem. The fault detection agents are contained in the fault detection context.

Fault Detection Agents. The fault detection agents, which are the T^2 and SPE statistics for each subsystem, assign themselves to the fault detection organizer of their subsystem (Figure 3). The fault detection organizer is responsible for keeping count of its fault detection agents, declaring consensus fault, keeping history of the performances of different fault detection agents under different fault scenarios, and in case of a consensus fault decision, triggering the diagnosis agent.

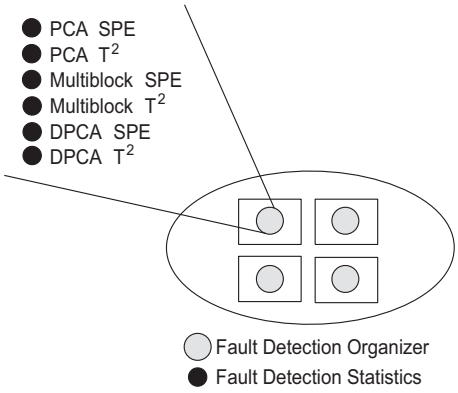

Fig. 3. Fault detection agents.

Statistics values and confidence limits are among the class variables for each fault detection agent. If the value of the statistic goes outside of the limits, the agent flags a fault. Fault detection organizer keeps track of all the fault flags given by its statistics. There are several criteria to form a consensus among different fault detection agents. The simplest would be to flag a fault, if the majority of the fault detection agents are flagging fault. This would require four of the six agents to flag a fault in order to declare that there is a fault in the unit. In the following, this strategy will be referred to as the "*number weighted*" consensus criteria.

Another criterion is based on the performances of fault detection agents over time, and based on their reliability, their decision is given more weight compared to less reliable fault detection agents. This is referred to as the "*reliability weighted*" consensus criteria. At each time point, when a new observation is available and new monitoring statistics are calculated, fault detection agents either detect a fault or not. Based on their decisions, they are given an instantaneous performance reward. The rewarding strategy is designed so that a missed alarm is penalized the most and the correct detection of fault is rewarded the most. A set of instantaneous performance rewards or penalties is given in Table 1, where the rows show the consensus and columns show the individual agent decisions. If the fault detection agent flags a fault but the consensus decision indicates otherwise, the agent is penalized for a false alarm. If the agent does not flag a fault, but the consensus flags a fault, then the agent is penalized for a missed alarm. A missed alarm is considered to be worse than a false alarm, and this is reflected in the instantaneous performance calculations. The instantaneous performances are summed in time for each detection agent, and makes up the accumulated performances. The reliability of an agent is determined by the accumulated performance values divided by the total accumulated performance value of all agents in that unit. The reliability weights are then considered in the consensus decision making.

Table 1. Instantaneous performance rewards

	Not faulty	Faulty
Not faulty	0.5	-0.5
Faulty	-1	1

The challenging problem of SPM methods is the missed and false alarm rates. For some cases, where the fault is

diffusing in the process and affecting the neighboring units and also with minor faults, the consensus flag may be oscillatory. This oscillation affects the performance mechanism in an undesired way such that an agent that has been flagging fault in the oscillatory period may not be reliable enough at that point to affect the consensus decision, and it will be penalized for flagging fault although the flag was right. Or an insensitive method could be rewarded if it did not flag the fault and again this would affect the consensus in an undesired way. In order to prevent these, the performances of agents are updated after fault episodes. A fault episode starts when a consensus fault is flagged. And the episode continues until no consensus fault is flagged for eight consecutive time points. At that point, looking back in history, the performances of agents are updated.

In order to design an automated fault detection framework, where the decisions of fault detection and diagnosis highly influence the succeeding tasks, reliability of the decisions is very important. Some of the monitoring methods may perform better than the others for various states of the process. Use of agent-based cooperation between different methods that are competing for the same task results in better overall performance than if these methods were used independently. Several monitoring agents have been implemented in MADCABS to provide diversity. The aim is to design an automated fault detection framework that can detect the faults on time, and that gives fewer false and missed alarms than if the monitoring methods were used independently. Comparison of different combinations of monitoring methods and the false and missed alarm rates of the corresponding monitoring statistics indicates that cooperation among agents improved the false and missed alarm rates.

Table 2. False and missed alarm summary of different fault detection agent combinations, using reliability weight condition

Agent	Missed Alarm	False Alarm
PCA	1.09	0.17
Multiblock PCA	20.98	0.01
DPCA	0.18	0.02
PCA-Multiblock PCA	1.21	0.06
DPCA-Multiblock PCA	0.75	0.07
PCA-DPCA	0	0.03
ALL	0	0.03

Diagnosis Agent. Diagnosis agent works in an event-driven way. It is activated when a consensus fault is flagged by a fault detection organizer. The responsibility of the diagnosis agent is to investigate the type and severity of the fault under consideration. Contribution plots are used as a diagnostic tool. The contributions of process variables to each fault detection statistic are calculated for different monitoring methods. The contribution plots do not indicate the source cause of the fault, but identify the variables that have contributed to the inflation in the SPM statistic. The diagnosis agent performs contribution plot analysis and determines the variables that inflated the SPM statistic that went out-of-control. At this point a sequence of events is activated. First, the most contributing variable to each statistic is chosen by checking if the variable contribution value is beyond the 3-*sigma* confidence limits and if it makes up a significant amount of the contributions. Each fault detection agent identifies their most contributing variables, and the common top most contributor to all is identified. That variable is then eliminated from the monitoring model data matrix and a new statistical model is built using the remaining variables. The aim is to detect if the fault is a sensor fault or a process fault. The assumption that is made here is, that a process fault usually has a fault signature and is realized in more than one variable. On the other hand, a sensor fault, especially a single sensor fault does not affect the other variables if it is not being used for control.

If the new observation, after the variable is eliminated, is in-control with the new model, it is declared as a potential sensor fault, however, the projection onto the new model continues in parallel to check if it will turn out to be a process fault later since some of the minor process faults can be misinterpreted as sensor faults in the beginning. If the new observation is not in-control with the new model, this means other variables have also been affected from the fault, and it is declared as a potential process fault. In addition to discriminating the types of faults, diagnosis agent estimates the severity of the fault by looking at how much the variable contributions have gone outside the confidence limits, how many of the neighboring fault detection organizer are signaling fault and how many of the fault detection statistics in the unit have identified the same fault signature.

3. MONITORING AND FAULT DETECTION OF A REACTOR NETWORK

3.1 Autocatalytic CSTR Network

Reactor networks hosting multiple species have a very complex behavior (Figure 2). As the number of steady states of the network increases, autocatalytic species are allowed to exist in the network that would otherwise not exist in a single CSTR. The cubic autocatalytic reaction for a single autocatalytic species is

$$R + 2P \rightarrow 3P \quad \text{and} \quad P \rightarrow D \tag{1}$$

R is the resource concentration, P is the species concentration, D is a dead species. The reaction rate for the first reaction, species growth rate constant, is k and k_d is the species death rate constant. The feed flow rates and interconnection flow rates are treated as manipulated variables. Each reactor has an inlet and outlet flow. The resource concentration in each reactor along with species concentrations is also available.

MADCABS is designed to work with process data from a real process plant or a process simulator. The data are stored in a database in MADCABS for use by MADCABS agents. For the case studies, the data are obtained from a simulator of the CSTR network, where multiple competing species coexist in the network and consume the same single resource. The ordinary differential equations modeling the operation of the reactors are written in C and connected to Repast Simphony through a Java Native Interface (JNI).

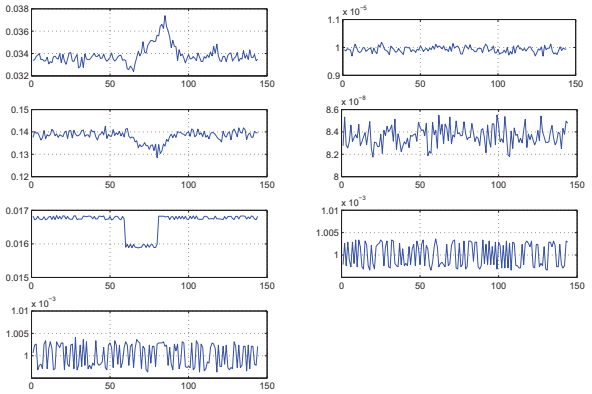

Fig. 4. A 5% process fault in the top right corner reactor 3.(a) Resource concentration in the reactor, (b) Species 1 concentration in the reactor, (c) Species 2 concentration in the reactor, (d) Species 3 concentration in the reactor (e) Feed flow rate into the reactor (Variable 5), (f) Outflowing interconnection to reactor 2 (g) Outflowing interconnection to reactor 7.

3.2 Monitoring and Fault Detection with MADCABS

The case studies use a four-by-five rectangular CSTR network, where three species coexist feeding from the same resource. Faults with different magnitudes and types are simulated to show the effectiveness of the agent-based monitoring, fault detection and diagnosis framework in MADCABS.

Fault Detection and Diagnosis. In Figure 4, a step decrease in the feed flow rate is introduced to reactor 3. The process fault affected the host species in the reactor since they start to die. The resource concentration in the reactor has increased after a delay after the dominant species start dying. From the figure, the variables that have been contributing to the fault are seen in the first three rows of the first column, the resource concentration in the reactor, dominant species concentration in the reactor and the feed flow rate to the reactor, which was reset to its original value after some time.

The contribution plots are given in Figure 5. For the five fault detection statistics that detected the existence of a fault in the system, PCA T^2, DPCA statistics and multiblock SPE showed that the main contributor to the fault is variable 5. PCA-SPE statistic had a smearing effect, where the signature of the fault could not be seen. Therefore, having multiple statistics improved diagnosis results as well. The fault has been detected on time by five fault detection agents.

Multiblock T^2 agent is insensitive to faults with magnitude less than 10%. A contribution chart that shows how many fault detection agents detected a contributing variable (Figure 6) indicates that variable 5, the feed flow rate to Reactor 3, is the common most contributor.

Table 2 provides a summary of 100 runs for each scenario. Multiblock PCA suffered from its insensitive T^2 agent, which had the highest missed alarm rate. The effect of the

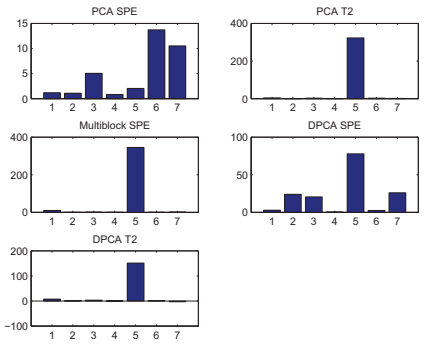

Fig. 5. Contribution plots for reactor 3 at the time of detection.

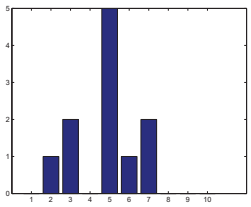

Fig. 6. Number of fault detection agents and common contributors for reactor 3.

insensitive statistic to the performance is realized in the performance of every combination with multiblock PCA. Combinations with DPCA improved the false and missed alarm rate. Especially PCA-DPCA performance is superior to any of the other less diverse combinations and seems to have a large impact on the combined performance when all three are used together. In summary, the results show that having multiple methods working together improves the effectiveness of the combined monitoring and fault detection.

Another type of fault in processes is sensor faults, where sensors might be defective and may provide false readings. A sensor fault should be identified in a timely manner since the measured variable can be used in computing the control actions and an erroneous reading may move the process to an undesired state, and may even destabilize the system. In general, correct and timely diagnosis and communication between control and diagnosis is required. A sensor fault in the form of a ramp decrease is given to the resource concentration sensor, variable 1 in reactor 6 (the figure is not provided because of space limitations). This sensor fault does not affect the other variables since it is not used for control.

The contribution plots show that the most contributing variable to the fault is variable 1. When this variable is taken out of the statistical model data, and a new model is built with one less variable, the new model reveals that the process is in-control. This indicates a potential sensor fault. The diagnosis results for reactor 6 are shared with control agents and also preprocessing agents in MADCABS so that preprocessing agents can provide reliable estimates instead of the faulty sensor, and control agents will continue to provide the necessary control actions to continue the desired operation level.

As another fault scenario, four consecutive faults are introduced to reactor 3. The fault is again introduced to the feed flow, and is of magnitude 1%. The fault introduction times and the detection times of the best performing combinations are given in Table 3. The reliability weight based consensus formation is shown to provide much earlier detection times than the majority based consensus criteria. Since the adapting reliability weight of the fault detection agents are taken into consideration, the first criteria provided earlier detection times, for consecutive faults. The results showed some kind of a learning pattern. However, this is going to be tested with different validation cases, where the training is followed by validation with different fault magnitudes. In Table 2, performance of DPCA-PCA combination was the same as the ALL combination, however, in Table 3, the detection times showed that when ALL of the monitoring agents are used in fault detection the fault is detected earlier than DPCA-PCA combination.

Table 3. Fault detection times (four consecutive process faults)

Agents	Fault#1	Fault#2	Fault#3	Fault#4
Actual Fault Times	220	250	290	330
PCA-DPCA (reliability)	221.9	250.9	290.4	330.8
ALL(reliability)	221.8	250.6	290.1	330.1
ALL(number)	223.1	254.4	294.0	333.6

The average number of agents that are flagging a fault when the consensus gives a fault flag is listed in Table 4 where two different consensus forming criteria are compared. When the agents' reliabilities increase with fault detection, less agents are required to declare a fault. When the presence of the majority of the fault detection agents is required to give a fault flag, the missed alarm rates increase and detection is delayed.

Table 4. Number of agents that flag fault at the time of detection (four consecutive process faults)

Agents	Fault#1	Fault#2	Fault#3	Fault#4
ALL(reliability)	4.06	3.46	3.48	3.25
ALL(number)	4.26	4.19	4.13	4.20

When the performances of different monitoring methods are compared the worst performing method is the multiblock PCA method because of the insensitive T^2 statistic. The overall performance of the PCA method is close to DPCA, but inferior because of the sensitive SPE statistic that gives many false alarms. The methods and the statistics are ranked and the results are provided in Table 5.

Table 5. Performances of the monitoring agents

Rank	Agent
1	DPCA SPE
2	Multiblock PCA SPE
3	PCA T^2
4	PCA SPE
5	DPCA T^2
6	Multiblock PCA T^2

4. SUMMARY AND CONCLUSIONS

PCA, DPCA and multiblock PCA methods are widely used multivariate SPM methods in process industries. However, all these methods are prone to false and missed alarms. The common practice in SPM is to test different monitoring tools form the literature, improve and tune the algorithms and find the best method that provides reliable monitoring. Considering the shortcomings of relying on a single SPM tool, consensus from several SPM tools is desirable. This is especially important for distributed and networked processes. Since there are multiple units, a monitoring system that is giving frequent false alarms on different operating units will be misleading and will not be relied on.

In order to improve the effectiveness of monitoring, several monitoring methods have been used together in the proposed framework. Some of the methods performed well on minor faults and disturbances, but had problems in contribution charts. Others gave good diagnostic results. Combining all these methods improved the effectiveness of the proposed overall monitoring, fault detection and diagnosis framework.

MADCABS provides an excellent environment to assess the performance of various SPM and fault detection methods for specific regions of process operation and adapt the reliance to different techniques based on prior experience and recursive assessment of performances. The agent management layer offers the tools and metrics to assess the performance of the monitoring, detection and diagnosis tools and dynamically update the confidence to specific techniques in a context-dependent way.

REFERENCES

Cinar, A., Palazoglu, A., and Kayihan, F. (2007). *Chemical Process Performance Evaluation.* CRC Press, Boca Raton, FL.

Jackson, J.E. (1980). Principal components and factor analysis: Part I-principal components. *J Qual Technol*, 12, 201–213.

Kourti, T. and MacGregor, J. (1996). Multivariate SPC methods for process and product monitoring. *J Qual Technol*, 28, 409–428.

Ku, W., Storer, R.H., and Georgakis, C. (1995). Disturbance detection and isolation by dynamic principal component analysis. *Chemometr Intell Lab*, 30, 179–196.

Qin, S., Valle, S., and Piovoso, M. (2001). On unifying mutliblock analysis with application to decentralized process monitoring. *J Chemometr*, 15, 715–742.

ROAD (2005). Repast organization for architecture and design. *Repast Simphony.* Available at http://repast.sourceforge.net.

Wangen, L. and Kowalski, B. (1988). A multiblock partial least squares algorithm for investigating complex chemical systems. *J Chemometr*, 3, 3–20.

Westerhuis, J., Kourti, T., and MacGregor, J. (1998). Analysis of multiblock and hierarchical PCA and PLS models. *J Chemometr*, 12, 301–321.

Wold, S., Kettaneh, N., and Tjessem, K. (1996). Hierarchical multiblock PLS and PC models for easier model interpretation and as an alternative to variable selection. *J Chemometr*, 10, 463–482.