Intelligent Control and Asset Management of Oil and Gas Production Facilities

by

Atalla F. Sayda

M.Sc.EE., University of New Brunswick, 2002
B.Sc.EE., Damascus University, 1996

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

In the Graduate Academic Unit of Electrical and Computer Engineering

Supervisor:	James H. Taylor, Ph.D., ECE/Control Systems
Examining Board:	name1, Ph.D., dept., Chair
	name2, Ph.D., ECE
	name3, Ph.D., ECE
External Examiner:	name, Ph.D., department, university

This thesis is accepted

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

January, 2008

Dedication

To my family

Abstract

Driven by the technical demand of the offshore oil and gas industry in Atlantic Canada, a joint venture among several Atlantic Canadian universities, and local and national companies was established in order to advance wireless systems technology in the oil and gas industry, and to assess the feasibility of an intelligent control and asset management system built on a wireless sensor network. As part of this research project, our team at the University of New Brunswick (UNB) is developing an *intelligent control and asset management system* (ICAM system) to manage the massive information flow from offshore oil rigs. The objective of this PhD thesis is to design the ICAM system architecture, to analyze its multi-faceted requirements, and to verify and validate its performance and logical behavior in normal and abnormal process situations.

The conceptual model of the ICAM system was defined along with its architecture, functional description and general logical behavior. A development plan to design such a complex system and the appropriate development tools were decided. The artificial intelligence (AI) requirements of the system were analyzed in terms of knowledge representation and processing, and the appropriate AI paradigm. The communication requirements were also analyzed after conducting a thorough review of middleware (i.e., communications) technologies. The structure, implementation and deployment of the system agents were defined based on the suggested system requirements.

A simple prototype of the ICAM system was designed in terms of the middleware layer, the intelligent supervisory agent of the system, and the reactive agents of the system prototype. The verification and validation of the system were demonstrated, where several scenarios were applied to the system to analyze its performance in real time and its logical behavior. The oil production facility simulation model, upon which the system's verification and validation have been demonstrated, was developed. A system performance analysis was conducted to detect any computational bottlenecks. Although the system prototype design has limitations, simulation results have demonstrated an effective system logical behavior and performance in real time.

Acknowledgement

I would like to express my gratitude and regards to professor James H. Taylor, my research supervisor, his encouragement, friendship, patience, and invaluable suggestions during the course of this research have played a vital role.

This work would not have achieved any success without the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). The support of Gensym Corporation and their great staff was a corner stone to the success of this project. I would also like to thank Verari Systems, Inc. for their great help and support.

This research project is also supported by Atlantic Canada Opportunities Agency (ACOA) under the Atlantic Innovation Fund (AIF) program. I gratefully acknowledge that support and the collaboration of the Cape Breton University (CBU) and the College of the North Atlantic (CNA).

I extend my appreciation to my research partners, Ms. Maira Omana, Ms. Pilar Moreno, Mr. Mazyar Laylabadi, Ms. Jing He and Ms. Liqiang Wang for their great work, contributions, and invaluable help.

I would also like to express my appreciation to the faculty and staff of the Department of Electrical and Computer Engineering at the University of New Brunswick for their patience and continuous encouragement.

Table of Contents

List of Tables

List of Figures

Symbols and Nomenclature

μ_w	Water viscosity
Φ	Longest oil droplet path angle
τ	Aqueous phase retention time
θ	Sector angle of the cross sectional area
$\tilde{N}(s), \tilde{D}(s)$	Left coprime factors
ε	Volume fraction of unseparated hydrocarbon
A_c	Cross-sectional area of the aqueous phase
d_m	Oil droplet diameter
F_{g1}	Flashed gas flow
F_{g2}	Dissolved gas flow
$F_{g_{out}}$	Gas discharge outflow
F_{h1}	Separated hydrocarbon fluid flow
F_{h2}	Unseparated hydrocarbon fluid flow
F_{in}	Oil-well fluid inflow
$F_{o_{out}}$	Oil discharge flow

F_o Oil flow

$F_{W_{out}}$ Aqueous phase outflow

F_{wat} Water outflow

F_W Dumped water outflow

h Oil-water interface height

$J(s)$ Transformation matrix

L Separator length

$L1$ Virtual separator length

Mw_g, Mw_o Gas and oil molecular weights

Mw_h, Mw_w, Mw_{in} Hydrocarbon, water, and oil-well fluid molecular weights

N_{gas} Number of gas moles in the gas phase

$P(s)$ Rational transfer function matrix

$p(s)$ Generalized parity vector

P_{v_i} Vapor pressure of component i

R Separator radius

SG_g, SG_o Gas and oil specific gravities

SG_h, SG_w Hydrocarbon fluid and water specific gravities

SG_h, SG_w, SG_{in} Hydrocarbon, water, and oil-well fluid specific gravities

T Absolute separator temperature

u_d Desired control inputs

v_h Oil droplet horizontal rising velocity

V_{oil}, V_{gas} Oil phase and gas phase volumes

v_v Oil droplet vertical rising velocity

V_{wat} Volume of the aqueous phase

x_i Mole fraction of the component i in the liquid phase

y Sensor outputs

y_i Mole fraction of the component i in the vapor phase

Z_g, Z_o, Z_w Gas, oil, and water molar fractions

AI Artificial intelligence

ANFIS Adaptive neuro-fuzzy inference system

ASM Abnormal situation management

BB Blackboard

CBR Case-based reasoning system

CFD Computational fluid dynamic

CVA Canonical variate analysis

DCS Distributed control system

DNS Domain name space

DSS Decision support system

FDIA Fault detection, isolation, and accommodation

FSM Finite state machine

GDA G2 diagnostic assistant

GPS Generalized parity space

GSI G2 standard interface

GUI Graphical user interface

HCPN Hierarchical colored petri net

HPCC High performance computing and communication

ICAM Intelligent control and asset management system

KB Knowledge base

LAN Local area network

MAS Multi-agent systems

MOM Message-oriented middleware

MPI Message passing interface

NetBIOS Network basic input/output system

OLE Object linking and embedding technology

OPC OLE for process control

PAWS Petroleum applications of wireless systems

PCA Principal component analysis

PRBS Pseudo random binary signal

QTA Qualitative trend analysis

RMA Remote memory access protocol

RPC Remote procedure call

RTD Residence time distribution

SDG Signed directed graph

SOAP Simple object access protocol

TCP/IP Transmission control protocol/Internet protocol

UDP User datagram protocol

WINS Windows name resolution service

Chapter 1

Introduction

1.1 Asset Management for Process Industries

Recent technological advances have resulted in increasingly complicated processes, systems and products that pose considerable challenges in their design, analysis, manufacturing and management for successful operation and use over their life cycles. In the process industries, 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. In a large process plant, there may be as many as 1500 process variables observed every few seconds, leading to information overload. Furthermore, the measurements may be insufficient, incomplete and/or unreliable due to a variety of causes such as noise, sensor biases or failures. In addition, the emphasis on quick diagnosis aggravates the situation by causing psychological strains on plant personnel [103].

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, abnormal event management is controlled manually in most process plants, which complicates their management and control. When it comes to operating process plants under normal conditions, manual management may also contribute to poor product quality and high cost. Managing process plants effectively under normal and abnormal situations define the asset management concept.

"Asset management is a systematic process of effectively maintaining, upgrading, and operating process assets. It combines engineering and economic principles, and provides the tools to facilitate a more organized and flexible approach to making decisions necessary to achieve high profitability. Maximum profitability is gained by maximizing adaptation and efficiency, minimizing unscheduled outages, and partial output, eliminating industrial injuries, and minimizing risk to the environment" [1]. Asset management may be viewed as the unifying element between production planning, quality and process control, safety maintenance, effectiveness and profitability. Oil companies, power and water utilities and other industries have recognized that, despite all their cost-cuttings, reorganizations, new technology, productivity and quality initiatives, the picture is fragmented. Inefficiency and conflicting objectives, lack of coordination and missed opportunities are still plentiful. Hence, it is certainly healthier and better suited to apply asset management to a complex process or manufacturing environment.

1.2 Intelligent System for Automated Plant Asset Management

Asset management and control of modern process plants involves many tasks of different time-scales and complexity, including:

- data reconciliation and fusion, where data are collected, filtered and combined

to detect and remove consistencies, reduce effects of error sources, and estimate parameters and variables not directly available,

- fault detection, isolation, and accommodation (FDIA), in which data are interpreted to derive the operational status of the external plant and the health of the control system,

- process model identification and optimization, in which process models are identified on a timely basis to allow adaptation to the non-linear and dynamic nature of complex process plants, and

- supervisory control, which coordinates all other asset management tasks, and tunes or reconfigures the control system if necessary to improve process operations.

The automation of these complementary tasks within an information and intelligent control infrastructure will reduce maintenance expenses, improve utilization and output of manufacturing equipment, enhance safety, and improve product quality [103, 73, 25, 76]. Many research studies, which proposed different combinations of systems theoretical and artificial intelligence techniques to tackle the asset management problem, have delineated a set of required features [103]:

- 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,

- understanding, and hence representing, process behavior at different levels of detail, and
- keeping the role of an operator primary and active in the operating environment, which is managed with the assistance of on-line intelligent systems.

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, on-line learning, automated process optimization and model identification, robust control algorithms – all under the direction of an expert system coordinator [92].

1.3 Challenges in Developing Automated Plant Asset Management Systems

Although automated asset management 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 [103, 63, 64, 100, 101, 102, 57]:

- Diversity of solution techniques, where several approaches are available to perform the main tasks of an automated asset 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 asset 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 asset management 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 abnormal situations which may happen in the plant: Some abnormal situations might develop over a few minutes, while others might develop over hours and days. Likewise some tasks of the asset management system might execute in few milli-seconds, such as the data reconciliation task 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.

1.4 Thesis Context and Organization

Driven by the technical demand of the offshore oil and gas industry in Atlantic Canada, a joint venture between several Atlantic Canadian universities, the National Research Council of Canada, and local and international companies was established in order to advance wireless systems 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. The petroleum applications of wireless systems (PAWS) project scope is to develop a control and information management system which consists of two subsystems [91]. The first subsystem is a wireless sensor network which will alleviate the need for data cables in offshore oil rigs and improve flexibility for adding and reconfiguring sensors. The second subsystem intelligently manages the massive data flow from oil rigs or refinery processes and interprets it so as to help operators make more appropriate decisions during abnormal events and, through intelligent control, improve process economics.

As part of the PAWS project, our team is developing an *intelligent control and asset management system* (ICAM system) to manage the massive information flow from offshore oil rigs [91]. The objective of this PhD research is to design the ICAM system architecture, to analyze its multi-faceted requirements, and to build, verify and validate the performance and logical behavior of a prototype ICAM system in several environmental situations. The remainder of the thesis is organized as follows:

A literature review of the intelligent asset management systems in industry and academia is conducted in chapter 2. The ICAM system research project as part of a bigger research program called petroleum applications of wireless systems (PAWS) is also introduced in chapter 2.

The conceptual model of the intelligent asset management system is defined, based on the human cognition-affect framework in chapter 3. The system architecture and its functional description are also discussed in chapter 3, as is the general logical

behavior of the system based on Durffee's informal theory of coordination. Finally, a development plan and the appropriate tools are surveyed and selected in this chapter for designing and implementing such a complex system.

The artificial intelligence (AI) requirements of the system are analyzed and discussed in chapter 4 in terms of knowledge representation and processing, and the appropriate AI paradigm. The communication requirements are also analyzed and discussed in chapter 4 after conducting a thorough review of middleware (i.e., communications) technologies. Among the middleware technologies the Message Passing Interface (MPI) technology was chosen, and the MPI communication requirements were further analyzed and refined. The structure, implementation and deployment of the system agents are also discussed in chapter 4.

A simple prototype of the system is designed and discussed in terms of the middleware layer, the intelligent supervisory agent of the system, and the reactive agents of the system prototype in chapter 5. The deployment scheme of the system is also thoroughly discussed in chapter 5.

The verification and validation of the system are demonstrated in chapter 6, where several scenarios were applied to the system to analyze its performance in real-time and its logical behavior. The research findings are summarized in chapter 7, and future system enhancements are also suggested. Finally, the oil production facility simulation model, upon which the systems verification and validation are based, is described and discussed in appendix A.

Chapter 2

Literature review

The automation of asset 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 this chapter, a survey of such research and industrial projects will be presented; here eight of the most relevant ones are reviewed.

2.1 The FORMENTOR Research Project

FORMENTOR, which is a joint venture of major European companies, namely Aerospatiale Protection Systemes (France), Cap Gemini Innovation (France), Det Norske Veritas (Norway) and the Institute for Systems Engineering and lnformatics of the Joint Research Centre of the Commission of the European Communities (Italy), 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" [108]. BP-Chemicals, one of the FORMENTOR consortium industrial partners, provided a butadiene plant as a test-bed to validate the final decision support

system [108, 46]. The main technical features of the system are [108]:

1. A goal tree-success tree (GTST), which is a representation of the functional model of the process. A GTST for a process relates high level safety and production objectives of the process to the functions carried out by its components. All the objectives of the process are described in terms of sub-objectives which may also be further refined. The state of each node in the GTST is either deduced based on the state or its sub-nodes or set by the output of the process fault diagnosis.

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. The MLM describes the structural and behavioral model of the plant as a hierarchy of components and their interrelationships. At the lowest level, the basic components of the process are described in terms of simple mathematical formulas, which do not capture the actual interactions among the variables in the component.

3. Two distinct but complementary reasoning modules for diagnosis, both based on this multi-layer structure. The first module, the Heuristic Causal Reasoning module, identifies known problems, looks up the known solutions, and then adapts the implementation of these high-level solutions to the plant state. The other, the Model-Based Reasoning module, uses the MLM to detect significant discrepancies between expected and real plant behavior and then to locate the malfunctions which could explain these discrepancies. The results of the diagnosis are posted onto the appropriate nodes in the GTST. The GTST model is used to assess the consequences of the

violation of the subgoals and to determine the corrective actions.

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.

2.2 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 Precarn Associates, undertaken by Ontario Hydro, CAE Electronics, the University of Toronto, Stelco Canada, Shell Canada, Hatch Associates and PRECARN Associates Inc, 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 [106, 65]. "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" [105].

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 [106]:

1. The data acquisition agent receives data from the plants main control computer.
2. The tracking agent continuously updates the data links between the agent system and the actual plant sensor positions, making sure that good communication between agents and the feed water system is constantly maintained.
3. The monitoring agent analyzes the feedwater sensor values and feedwater alarms and then produces symbolic descriptions of the plants 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 agents output against the simulators ideal values.

The entire APACS system was implemented in C++ using Expersofts XShell (an extended C++ syntax for declaring distributed objects) as its communication environment and CLIPS (a rules-based inferencing environment constructed by

NASA) [106]. 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.

2.3 The Pilots Associate (PA) Program

The first research program to address the asset management problem in the US was the Pilots 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 pilots' situational awareness and decreasing their workload. DARPA wanted to advance the programs 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 [86, 6].

"The Pilots 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" [6]. Another project, which followed the PA program to address the asset management problem in attack helicopters, is the Rotorcraft Pilots Associate (RPA) program. The goal of the US Army funded RPA program was to develop and demonstrate in flight an advanced, intelligent associate system in a next-generation

attack/scout helicopter [61].

2.4 Abnormal Situation Management (ASM)

The PA and RPA projects paved the way for other projects to develop and automate the asset 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 [11]. The AEGIS project proposes a comprehensive asset management framework from an industrial view point. AEGIS built on the experience of military aviation research projects, especially the Pilots Associate (PA) and the Rotorcraft Pilots Associate (RPA) [10]. It is really worth considering the project and its current status, since it is supported by major oil and gas companies (i.e., Shell, Exxon, Chevron, BP, and Nova Chemicals) 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 will end 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).

2.4.1 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 [63, 64]. The principle of DKIT is black-board collective problem solving, in which several modules are integrated [63]:

- 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. Diagnostic results are presented in the form of individual methods' results, combined results, individual method confidence, supporting trends for a fault hypothesis, propagation paths for a fault.
- 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.

2.4.2 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 [100, 101]. 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 [101]:

- Data Acquisition Module, which acquires on-line data from the plant and makes them available to other modules. The on-line data are compressed using B-Splines-based data compression algorithm and then stored in a historical archive.
- Monitoring Module: This module monitors the process data for the presence of abnormalities using a principal component analysis (PCA) model of the process. If the residual of the PCA model is consistently above its confidence limit over a wait time, then the presence of an abnormality is indicated.
- Diagnosis Module, which identifies the root causes for the abnormalities. Multiple diagnosis methods are combined in a blackboard architecture, namely, a diagnosis system that combines PCA and signed directed graphs (PCA-SDG), and the B-Splines-based adaptive system for trend analysis.
- Fault Parameter Magnitude Estimation (FRAME) Module, which estimates the magnitude and rate of change of the root causes. A dynamic optimization is performed to estimate these values.

- Simulation Module, which performs a simulation to predict future values of the process outputs. These outputs are then compared against the process constraints to determine if any process constraints will be violated and the time at which they occur.
- 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.

2.4.3 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 [103]. 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) [68]:

- 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.

2.5 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. The DSS will be developed such as to be able to interface with commercial plant database and process control software" [9].

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) [58].

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 [58].

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" [8].

2.6 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)" [23]. 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 [82, 22, 59, 45, 29, 13].

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 [59, 21].

2.7 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 [19].

The DIAMOND system was implemented using the KQML-COBRA- (knowledge query and manipulation language) based architecture, in which the different agents are implemented as distributed COBRA objects.The system prototype was evaluated while monitoring and diagnosing the watersteam cycle chemistry of a coal-fired power plant [19].

2.8 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, namely Gerhard-Mercator Universitat, University of Kalsruhe, and Laboratoire d'Automatique de Grenoble. 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" [47].

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 [47, 28]. 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 [47, 28].

2.9 Other Related Work

Cinar *et al.* have successfully combined multivariate statistical data analysis with expert systems for process fault diagnosis. Basically the multivariate statistical data analysis module developed in MATLAB was converted into C code, and then linked with a G2 expert system through a G2 standard interface (GSI) link [67, 89]. Cinar *et al.* exploited only the G2 diagnostic assistant (GDA) capability (i.e., a graphical design tool similar to Simulink/MATLAB). Thornhill *et al.* used the Computer Aided

Engineering Exchange (CAEX) IEC/PAS 62424 standard for representation of process information in XML. The CAEX Plant Analyzer prototype incorporates process knowledge by linking the plant topology and a simple reasoning engine developed in Prolog with the results from plant disturbance signal-based analysis [110]. Chan *et al.* designed a rule-based expert system for the management of petroleum contaminated sites, where a variety of methodologies and tools are employed and integrated, e.g. IMT (Inferential Modeling Technique) for knowledge analysis, decision trees and object-oriented methodology for knowledge representation, and fuzzy set theory for uncertainty modeling. The system was implemented with the real time expert system shell G2 [19, 30].

2.10 Petroleum Applications of Wireless Systems (PAWS)

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 asset 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 PAWS (Petroleum Applications of Wireless Systems) 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 [91]. This will be embedded in a fully automated system architecture, which will better support process operators and improve

operability [96, 70, 95, 50, 87, 78, 80, 71, 93, 52, 79, 72, 81, 94, 97]. The final developed system should be deployed and validated on a pilot plant which emulates an offshore oil production facility, as illustrated in figure 2.1. This thesis presents the requirements analysis, architecture, implementation plan and a prototype of such a system.

The pilot plant basically consists of three processes. The first is a two phase separator in which hydrocarbon fluids from oil wells are separated into two phases to remove as much light hydrocarbon gases as possible. The produced liquid is then pumped to a three-phase separator, where water and solids are separated from oil. Oil is heated in this process to remove as many suspended water droplets from the oil phase as possible. The produced oil is pumped out and sold to refineries and petrochemical plants if it meets the required specifications. Flashed light and medium gases from the separation processes are sent to a gas scrubber where medium hydrocarbon and other liquid remnants are separated from gas and sent back for further treatment. Produced gas is then compressed and pumped out for sales (refer to appendix A).

Figure 2.1 shows the different level and pressure control loops, which maintain the produced oil at the required specifications. As the PAWS project scope suggests, all the process control instrumentations will be hooked up to a wireless communication system. Measured data are transmitted to the control room where the intelligent control and asset management (ICAM) system interprets these data for better process control and management. ICAM 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 mitiga-

tion 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.

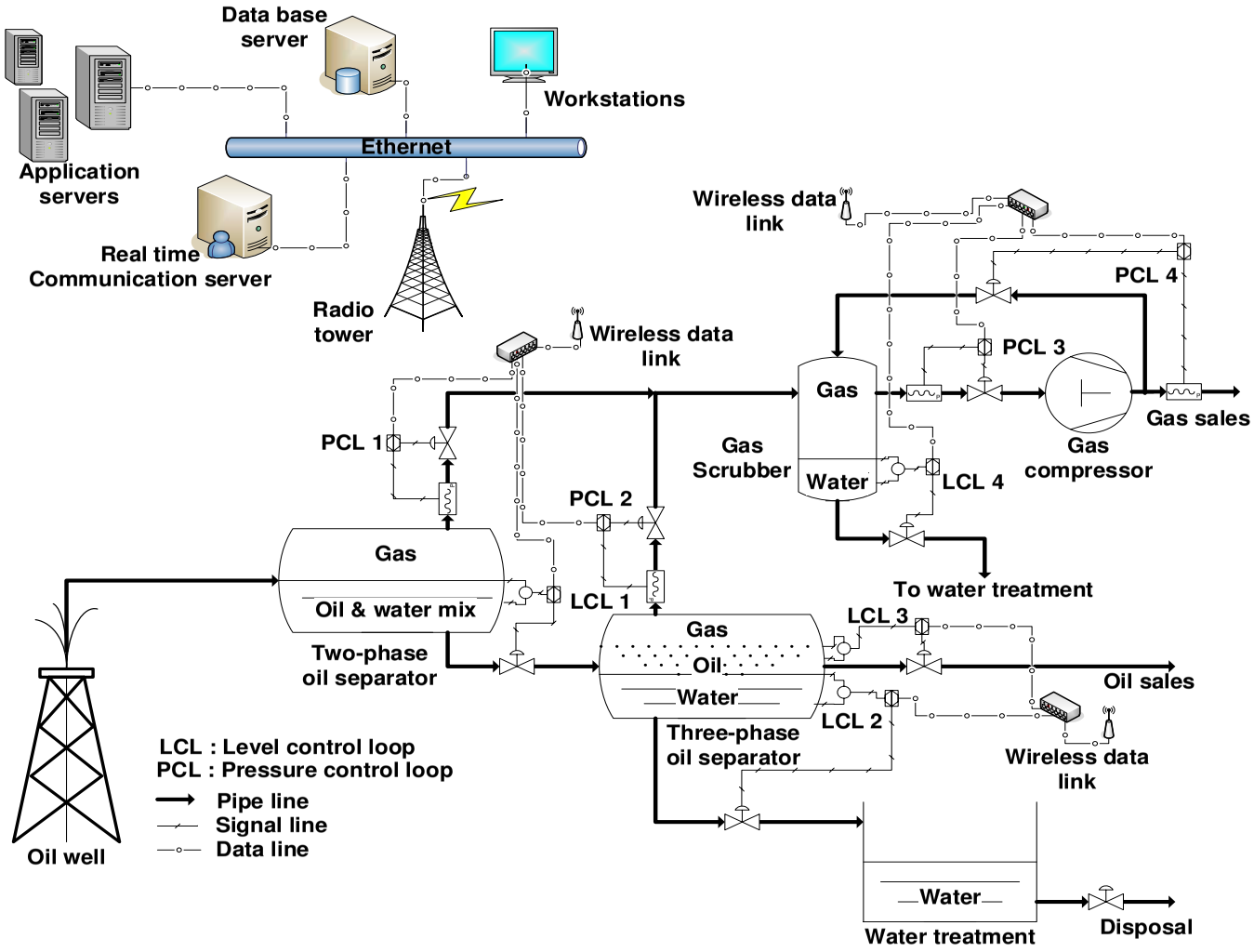

Figure 2.1: PAWS Project schematic diagram

Chapter 3

ICAM System Conceptualization: Architecture and Functional Description

3.1 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 [62] 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 [55, 54].

Newell [66] 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 mecha-

nisms, 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. 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 [77]. 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 [16]. 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 [109]. Each agent contains processes for behavior generation, world modeling, sensory processing, and value judgment together with a knowledge database, as shown in figure 3.1.

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 [44].

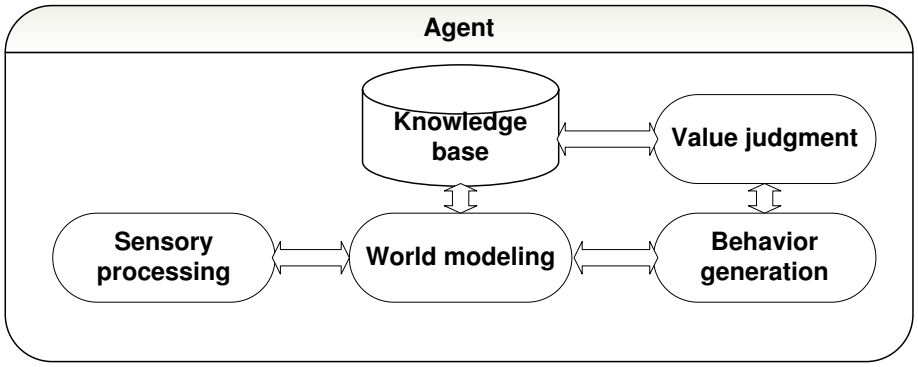

Figure 3.1: Agent architecture

Sloman [85] 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 3.2, 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 [84]. 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 process plants. The architecture of the system and its functional modules will be discussed in subsequent sections.

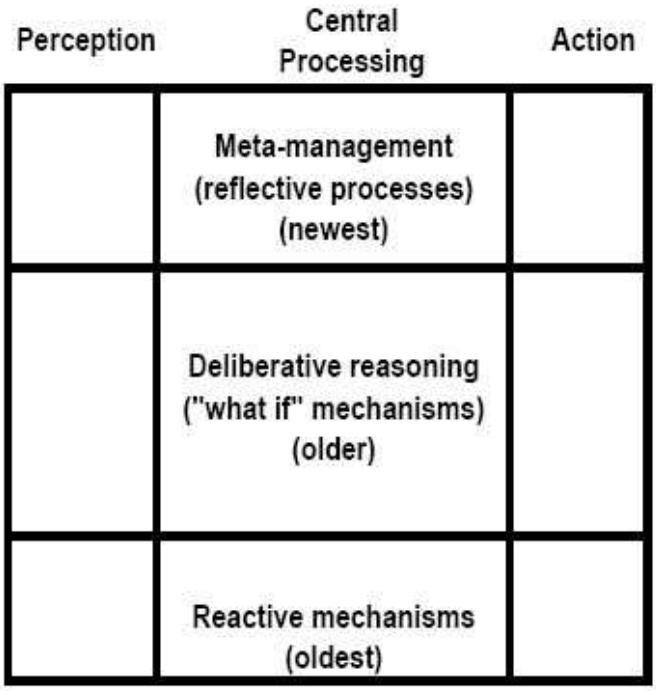

Figure 3.2: Human cognition and affect (H-Cogaff) architecture [85]

3.2 System Functional Description and Architecture

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 [90].

Figure 3.3 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. 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 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 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. In the remainder of this section, the functionalities of the agents in each layer will be discussed.

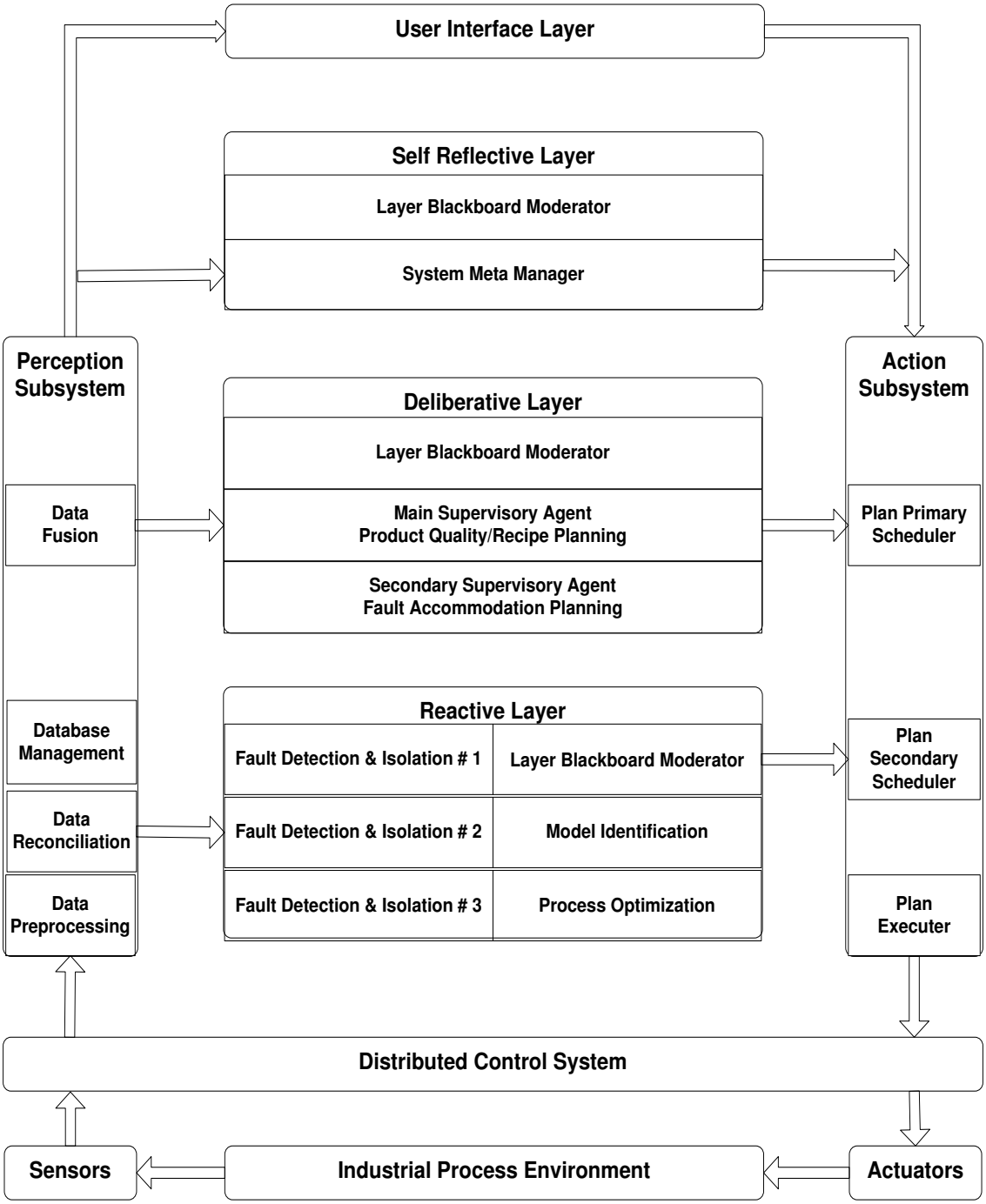

Figure 3.3: ICAM system architecture

3.2.1 The Perception Subsystem

In order to tackle the problem of data explosion in modern complex process plants, the perceptual subsystem will process data in a hierarchical manner, and categorize them into different levels of abstraction. The data stream is processed serially by different agents, where the first agent's 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) and the underlying process model. The data reconciliation agent exploits the adaptive nonlinear dynamic data reconciliation approach [93, 52]. The data are 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.

3.2.2 The Reactive Layer

Agents in this layer provide a direct response to events that occur in the environment. Here we describe both agents built for the present ICAM system prototype as well as those to be implemented in the future.

When an abnormal event occurs, several fault detection and isolation (FDI) agents work concurrently and complementarily 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 [103, 32, 41]. FDI basically consists of two tasks, as shown in figure 3.4. 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 have been 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 as discussed below. 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.

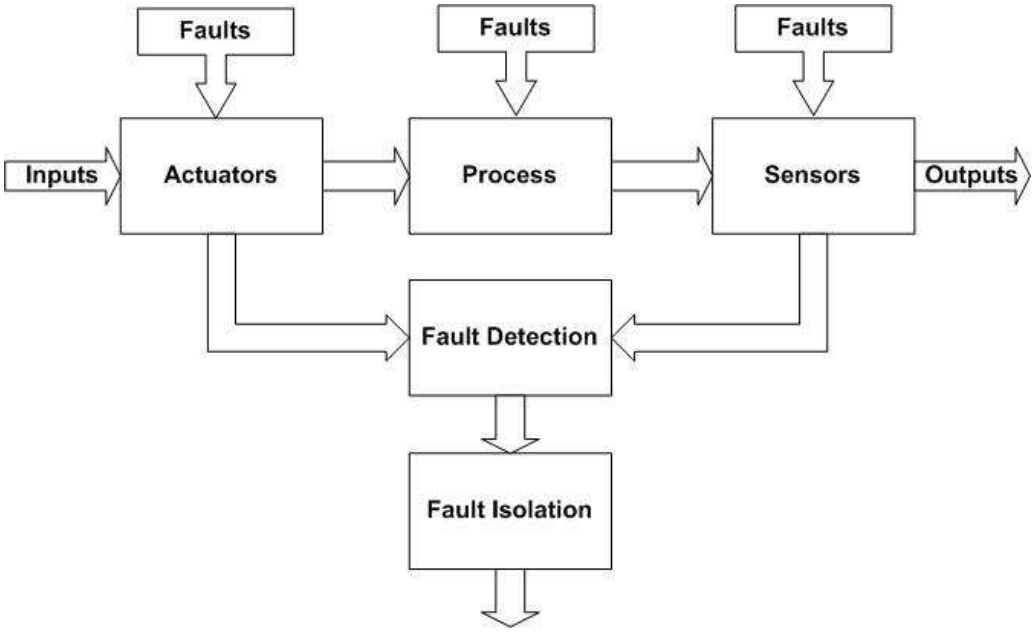

Figure 3.4: Fault detection and isolation (FDI) scheme

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 [104], 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:

$$P(s) = \tilde{D}^{-1}(s)\tilde{N}(s) = \frac{y(s)}{u_d(s)} \tag{3.1}$$

which implies that

$$\tilde{D}(s)y(s) - \tilde{N}(s)u_d(s) = 0 \tag{3.2}$$

Under ideal conditions, when the plant is linear, noise- and fault-free, equation 3.2 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

$$p(s) = J(s)[\tilde{D}(s)y(s) - \tilde{N}(s)u_d(s)] \tag{3.3}$$

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 [104, 70, 71, 69, 72]. Once the generalized parity vector is generated then its magnitude and direction are compared to a threshold and direction set 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. This agent is implemented and incorporated in the ICAM system prototype.

A new fuzzy signed digraph (SDG) model-based FDI technique is another approach evaluated [107]. Signed digraphs, which have 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 is developed from the SDG process model, as suggested by Kramer [48]. 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 [88]. This FDI technique has been implemented and evaluated; however, further investigation and testing is required to decide which inference technique could be used (e.g., backward chaining, forward chaining, fuzzy inference) [107]. This agent would also 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 [39]. 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 memberships 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 [42]. In order to evaluate this approach for fault detection and isolation, we are considering 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 also 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 detection and isolation task, and compensate for or accommodate faulty sensor signals (estimation task). The different agent tasks are decided by the deliberative layer, depending on the situation. For instance, if the operating point of the plant changes to meet new required product specifications, the deliberative layer will use the model identification agent to estimate the new process model parameters for further processing [50]. This agent was built by the author and incorporated in the ICAM system prototype.

An optimization agent may also be embedded in the reactive layer to make the best use of available equipment and raw materials. The agent receives product quality plans and process operation constraints from the deliberative layer, and then the agent formulates 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 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 intermediary such as a blackboard repository [12]. A blackboard agent could 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 abstraction 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 is available or not. This would meet the requirements of concurrency and autonomy for high system performance. We think that merging the two approaches in a hybrid one would combine their benefits. Every agent in the hybrid approach has its own blackboard, through which other agents can interact directly. This would achieve more autonomy for each agent and high system performance. The hybrid approach was incorporated in the ICAM system prototype.

3.2.3 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 abnormal circumstances. Planning in this layer will not attempt to work in a vacuum. Rather, it will conceptually employ a library of pre-specified plans and a problem solving mechanism. The artificial intelligence (AI) requirements of the ICAM system have to address different issues such as coordinating the system's internal behavior (i.e., how the different agents interact) versus managing the external industrial environment. The choice of the appropriate AI paradigm is very crucial to the high performance and real-time requirements of the ICAM system. Different AI paradigms (e.g., rule based expert systems, case-based reasoning (CBR) systems, neural nets (NN), etc.) have different

strengths and disadvantages.

When it comes to selecting an appropriate AI paradigm, we were at first attracted by the case-based reasoning (CBR) approach, as it promised to meet the high performance, learning and real-time requirements of the ICAM system [96, 95]. The CBR paradigm is a novel problem-solving strategy and machine learning technique. In principle, it solves problems by retrieving a "nearest neighbor" past problem from its case base, evaluating any differences, and adapting the past problem solution to handle the new circumstances. Every new problem that is handled successfully is added to the case base; if the new solution is a failure that information is also stored. While this approach is apparently systematic and easily automated, there are several major drawbacks: (1) developing an algorithm to extract a "nearest neighbor" problem is domain-specific and may be very difficult, and (2) "adapting the past problem solution to handle the new circumstances" is also much easier said than done. Although many CBR programs were developed during the 80's and mid 90's [4, 53], the CBR development process slowed greatly due to the problems mentioned above (and others), so we are shifting to another paradigm.

A rule-based expert system combines a domain-specific knowledge base with an inference engine that processes knowledge encoded in the knowledge base to respond to conditions in the plant or to process operators' decisions. An expert system consists of the following components [33]:

1. User Interface: The means by which the expert system and the users communicate.

2. Explanation Facility: The system explains to the user the way it reasons.

3. Working Memory: A global data base where all the facts used by the rules are stored.

4. Inference Engine: Makes inferences by deciding which rules are satisfied by

facts, prioritizing the satisfied rules, and executing the rule with the highest priority.

5. Knowledge Acquisition Facility: An automatic way to introduce new knowledge without the necessity of coding it.

Traditional rule-based expert systems have a few well-known drawbacks, such as difficult knowledge acquisition, lack of a memory of tackled problems or previous experience, poor inference efficiency, ineffectiveness in dealing with exceptions and novel situations and lack of learning mechanisms, to name a few. However, the development of new software standards and technologies for rule-based expert systems continued to progress. Such development has enabled rule-based expert systems to overcome many of their drawbacks, and to compete with the CBR AI paradigm. In fact, if we can limit ourselves to "crisp" problems then the "nearest neighbor" problem does not arise, and we can use rules to define a case base, and retrieve and implement solutions. Semi-automated procedures can also be implemented to allow operators or process engineers to enter new cases and thus implement a limited form of learning.

The deliberative layer supervises the system through two rule-based reasoning supervisory agents. The first agent is the main supervisor, which manages the system during normal operation circumstances. The agent's knowledge base contains product quality profiles, their pre-specified raw material recipes, and the associated process operating conditions. The other supervisory 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 knowledge base. These plans consist of schemes for sensor/actuator reconfiguration and controller tuning/restructuring, as well as fault propagation scenarios and recommended predictive maintenance procedures.

3.2.4 The Self-reflective Layer

This self reflective layer conceptually 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.

3.2.5 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.

3.2.6 The Action Subsystem

To complete our conceptual model, plans which are sent by the deliberative layer would be 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 accordance 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.

3.3 ICAM System Conceptual Behavior Model

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.* [17] proposed an informal theory that integrates organizational be-

havior, 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 [15]. 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

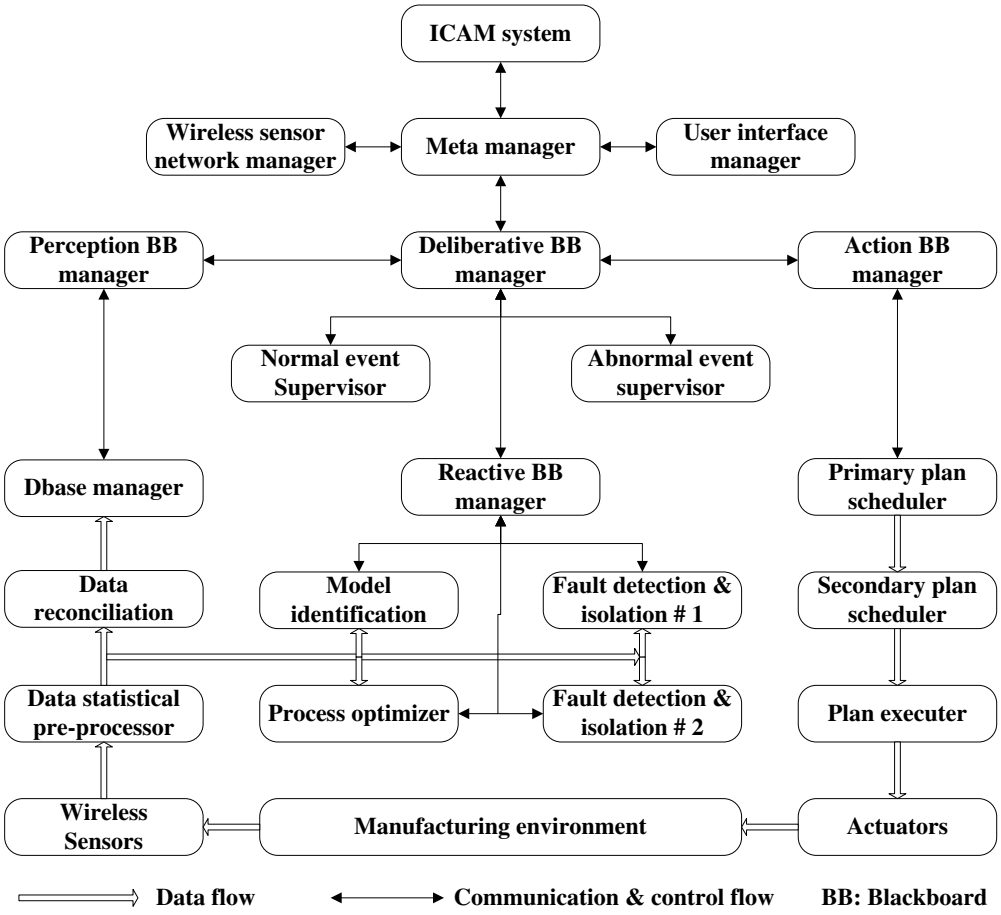

Figure 3.5: ICAM system conceptual behavior model

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 3.5, 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 [96, 95]. We adopted Sloman's H-Cogaff architectural scheme because it met most of our system requirements [84]. 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. We follow the top-down approach to explain the conceptual behavior model of the ICAM system.

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 3.1 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 3.1: Conceptual structure of behavioral message

The meta manager may interact with other ICAM agents through a group of distributed blackboard agents, which act as a post office. Each blackboard (BB) agent would consist of a global data repository and a control mechanism, which processes the different received messages and notifies other agent about newly available messages. Once the meta manager receives a message from any of the BB agents, it monitors the logical behavior by comparing the sender and the recipient identities and status with its current state. Furthermore it assesses the dynamic behavior of

the system in this event by assigning a time out for the recipient to respond. Should the recipient not respond within the preset time out, the meta manager proposes an alternative internal action. Each BB agent broadcasts messages to the recipients and to the higher level BB agents. The highest level BB agent would send the massage to the meta manager for further processing.

The deliberative BB manager interacts with the lower level BB managers and the two rule-based reasoning supervisory agents of ICAM. The main supervisor manages the system during normal manufacturing operations. When a certain product specification is required, the main supervisor retrieves a set of plans that best match the required quality specs. If the matching process is successful, the plan is sent to the deliberative BB manager, which in turn communicates it to the action BB manager for execution. If not, the closest matching plan is chosen and adapted by using model-based optimization, in which the main supervisor collaborates with the model identification and optimization agents to generate the optimal recipe and operating conditions (e.g., pressure and temperature). This collaboration process is achieved through the deliberative and reactive BB managers. The plan is then sent to the user interface agent for further modifications by process operators if needed. Once the plan has been approved it is then sent to the action BB manager for execution. The actual quality specifications are monitored by the main supervisory agent, which will add the plan to its "good" knowledge base 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 [92]. The meta manager is acknowledged in every step, to guarantee a coherent internal coordination.

The second supervisor is a backup agent to manage the system in case of faulty operations. When a fault happens (e.g., a sensor or actuator failure), the backup agent receives fault assessments from the different fault detection and isolation (FDI) agents through the reactive BB manager. Based on such assessments, the supervisor

retrieves the most closely matching case from its knowledge base. Consequently, it alarms the user interface through the appropriate BB managers about the fault, its possible causes, and recommended mitigating actions for operator feedback and approval. The backup supervisor may interfere directly in critical situations in collaboration with the main supervisor and the meta manager. This would prevent the system performance from deteriorating excessively and would keep it in an acceptable state.

Four reactive agents respond directly to events as they are continuously updated with new data about the external environment. Data from the external plant are received by the statistical data monitoring agent, which preprocesses the data by removing undesired discrepancies. Data are then reconciled with material balance laws in the data reconciliation agent [93, 52]. The database manager then stores processed data in its database. The perception BB manager connects these data processing agents with the system. A model identification agent estimates new process parameters to improve the system knowledge about the process, should it receive a message from the perception BB manager to do so. The production efficiency is improved by a process optimization agent, monitored by the main supervisor as discussed earlier. When abnormal events occur, several FDI agents may collaborate with the backup supervisor and other BB managers to detect and isolate such faults. The first agent to detect abnormal process variations is the statistical data monitor, which alerts FDI and backup agents through appropriate BB managers. Again, the meta manager is acknowledged at every step to ensure coherent coordination among the agents.

Plans received by the action BB manager are sent to the main and secondary scheduling agents where plans are decomposed into tasks with shorter time frames and in accord with the sub-processes of the plant. Subtasks are then executed by the execution agent. Massive data flow is handled by a wireless sensor network agent,

which manages real time communications between the control room and the offshore oil facility. Process-critical information is pro-actively presented to process operators in a timely manner through the user interface agent.

3.4 ICAM System Development Plan

Having discussed the architecture and behavioral model of the ICAM system, it is crucial to prepare an implementation plan and choose the appropriate development tools. The implementation plan is a preliminary one for prototyping and performance analysis purposes. Industry consultation will determine the final system requirements and deployment plan. Our conceptual implementation plan is composed of the following phases:

1. *ICAM system logical behavior analysis using the colored petri nets (CPN) approach:* The CPN modeling approach combines Petri nets and programming languages, which enables modeling concurrency, synchronization, and communication in systems. The module concept is hierarchical, allowing a set of modules to be composed to form new modules. A CPN model can be used to verify a number of dynamic properties of the system under consideration such as liveness, boundedness, and fairness [49].

2. *ICAM system deployment scheme using the message passing interface (MPI) parallel programming model:* The message passing model is a parallel programming approach which posits a set of processes that have only local memory but are able to communicate with other processes by sending and receiving messages [35]. MPI is a specification and a library which provides the infrastructure for communications among several parallel computational processes.

3. *Canonical variate analysis (CVA) integration for massive data flow processing:* Massive dataflow from industrial process should be preprocessed so as to re-

move any inconsistencies, to model the process, and to generate better process measurements in accordance with mass balance laws. The canonical variate analysis (CVA) approach is a well established statistical approach, which can meet the dataflow preprocessing requirements mentioned earlier. This decision was made as a result of a three day model identification workshop held at the University of New Brunswick [50]. The integration of the CVA-based Adaptx package with ICAM would result in robust data preprocessing and modeling compared with other approaches.

4. *Bottom-up ICAM system rapid prototyping using the* MATLAB *simulation environment:* A prototype has to be developed in order to have the ICAM system requirements deployed in a real-world system. The ICAM prototype should be deployed as distributed MATLAB computational modules, which run on a network of several Windows XP workstations. The different computational agents employ different approaches based on the agent's task.

5. *Interfacing with the College of North Atlantic (CNA) oil production pilot plant:* This step would be the ultimate platform for verifying the ICAM system design decisions and for validating the ICAM system performance and logical behavior in a real-world industrial environment. To accomplish this, we suggested upgrading its equipment and instrumentation, especially the sensors. In addition to installing a LAN of workstations, we will use the MATLAB simulation package including the OPC (object linking and embedding (OLE) for process control) toolbox, which will act as an interface between the ICAM system and the pilot plant. This would be very productive and would guarantee a good development and utilization process for the CNA facility even after PAWS is completed. This part of the plan assumed CNA will complete pilot plant upgrades in a timely way. Otherwise, a pilot plant simulation model [80] should be used as a backup platform (refer to appendix A).

Chapter 4

Conceptual ICAM System Implementation Requirements

Having proposed a conceptual model, architecture, behavioral model, and implementation plan 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.

4.1 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 [31] is considered the most versatile real-time expert system shell, as it integrates many software technologies and standards. The G2 platform uniquely combines real-time reasoning technologies, including rules, work flows, procedures, object-oriented modeling, simulation, and graphics, in a single development and deployment environment. G2 can transform real-time operations data into automated decisions and actions, and can maintain an understanding of the behavior of processes over time. This would enhance the whole ICAM system performance, and enable the ICAM system to intelligently coordinate its internal behavior and interact with the external industrial process as well.

The integration of the G2 expert system development environment with the

ICAM system would benefit from and build on the previous G2 integration attempts. This would enhance the whole ICAM system performance, and enable the ICAM system to intelligently coordinate its internal behavior and interact with the external industrial process as well. 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. G2 also adheres to software development standards such as object-oriented design, modularity, reusability, scalability, application programmer's interface (API), and user interface standards [31]. To meet such 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 and processing have to be determined.

4.1.1 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 [31]. 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. Figure 4.1 illustrates the general architecture of the ICAM supervisory agent, which accordingly 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.

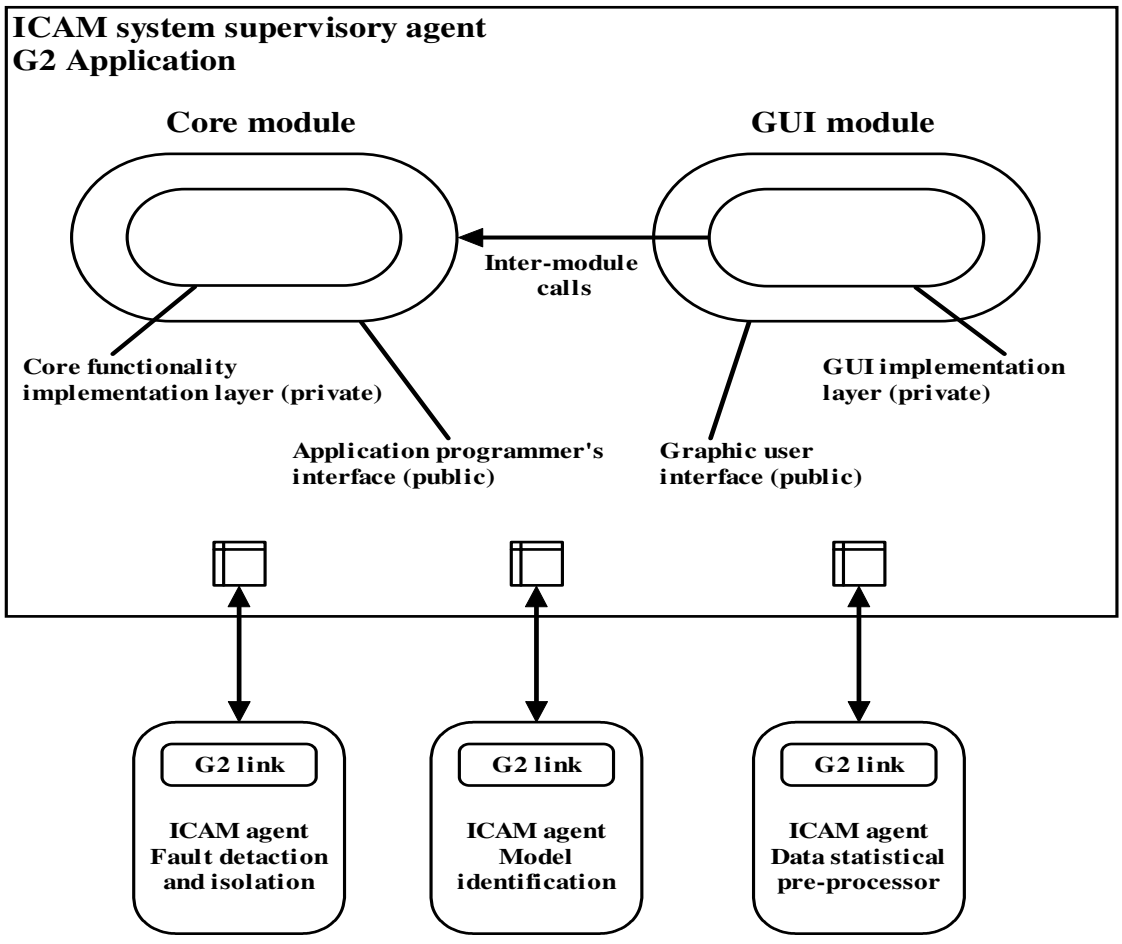

Figure 4.1: ICAM system supervisory and reactive agents architecture

4.1.2 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 4.2. 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 [57]. The modular knowledge base design approach supports objected-oriented design principles, increases productivity, encourages code reuse and scalability, and improves maintainability.

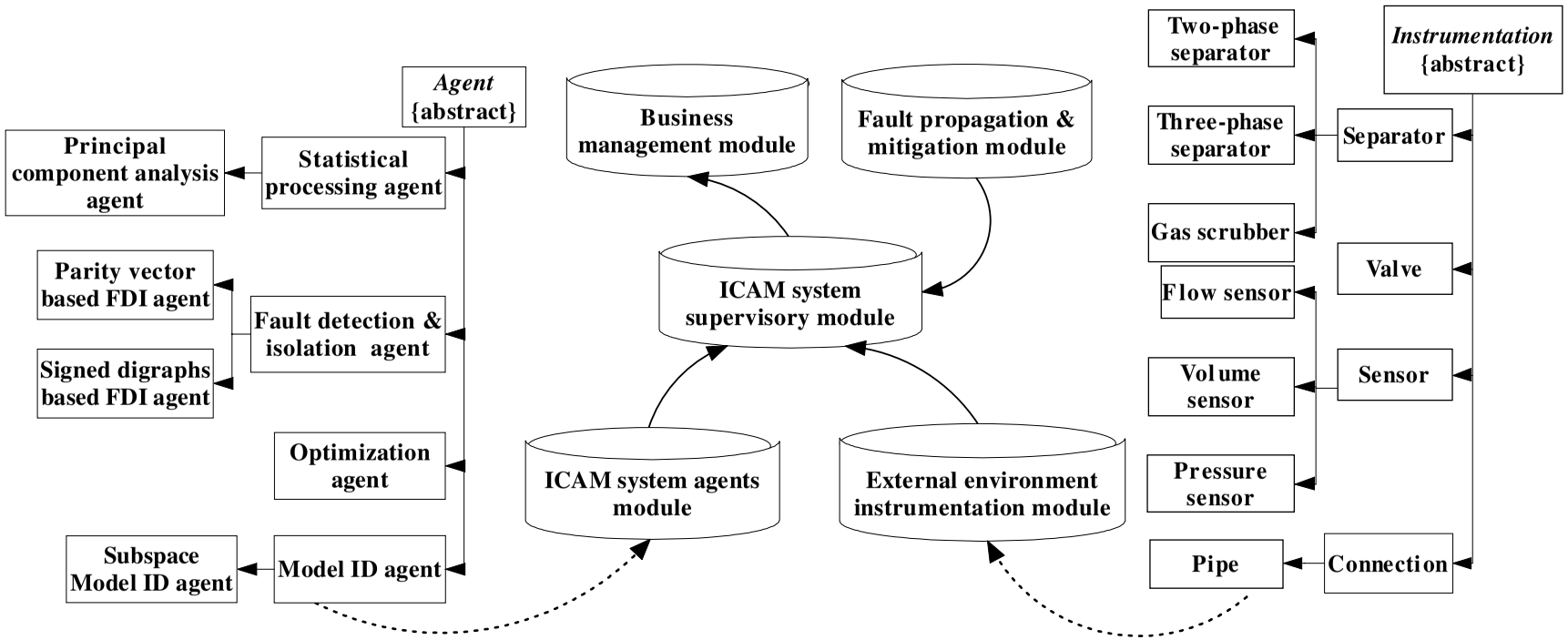

Figure 4.2: Knowledge representation structure in the ICAM supervisory agent

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. An abstract class, which hides its basic properties and methods, is first designed. More properties and methods are added to the higher level classes in the class hierarchy, which inherit from the abstract class.

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), as illustrated in figure 4.3. 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.

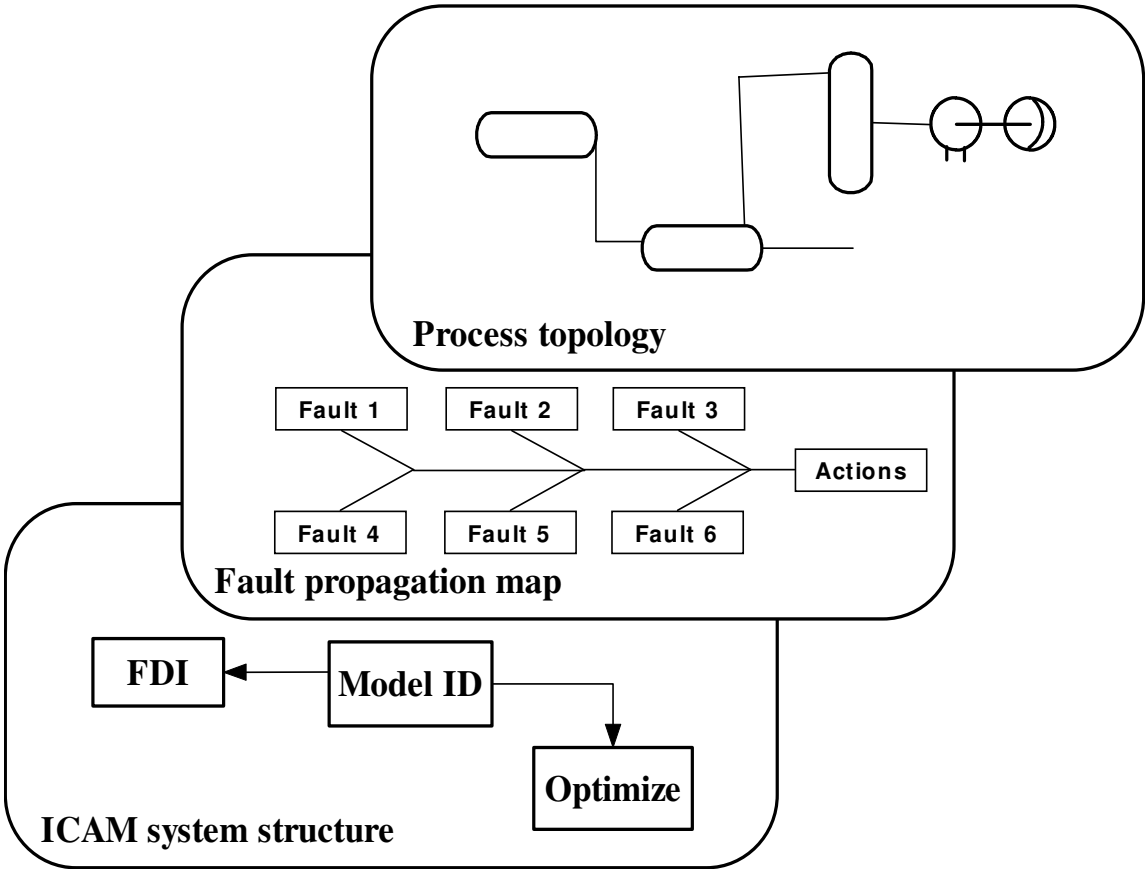

Figure 4.3: Layers of the supervisory agent knowledge

4.1.3 Knowledge processing in the supervisory agent

The G2 development environment offers several programming constructs for processing data such as procedures, methods, and rules. Procedures, which are independent of any class, define a general functionality, and can be invoked by rules, other procedures, and GUI actions. In contrast, methods, whose invocation and structure are similar to procedures, define classes' behavior. Methods structure the behavior of different objects through method inheritance between different classes of the class hi-

erarchy. Procedures and method invocation can be single-threaded (i.e., sequential) or multi-threaded (parallel). Rules, which are statements with antecedent and consequent parts, represent expert decision-making power by reasoning over data or event conditions ("facts"). Rules are invoked by forward chaining, backward chaining, or scanning.

Forward chaining is a form of deductive reasoning, in which rules are invoked to attempt to draw conclusions from existing facts. If the antecedent of a rule is true, then its consequent part is executed to create new facts, thus causing a chain of rule invocations in a rule base. Backward chaining is the process of determining the value of a goal or a variable by looking for rules that conclude that goal. Backward chaining follows two different search strategies. The breadth first search strategy examines all rules that could determine the value of the current goal and sets their antecedents as sub-goals before backtracking through other rules to determine the validity of each sub-goal. In contrast, the depth first search strategy backtracks through all of the rules in a knowledge base that could lead to determining the value of a goal by a single rule. Scanning is another rule invocation method in which rules are invoked automatically based on a fixed, user-defined frequency.

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 [31]. 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.

4.2 Communication Requirements for the ICAM System

Having proposed the ICAM system development plan [78], 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" [2]. Middleware can take on the following different models [3, 24, 7, 18, 74]:

1. Transactional middleware, which "permits client applications to request several services within a transaction from a server application. The client can request services using synchronous or asynchronous communication through the two-phase commit protocol (2PC). This middleware platform is rather scalable, because it supports load balancing and replication of server components. Moreover, it supports software and hardware heterogeneity, because

the components can be located on different hardware and operating system platforms" [74].

2. Procedural middleware, which "enables the logic of an application to be distributed across the network. Program logic on remote systems can be executed by Remote Procedure Calls (RPCs) as simply as calling a local procedure" [2]. "Remote clients can invoke these procedures across the network using the network protocols. These protocols are low-level, such as Transmission Control Protocol/Internet Protocol (TCP/IP) or User Datagram Protocol (UDP). If a client wants to receive some services, then it makes a request to a server. This request consists of a message, which includes the marshalled parameters. On the other side, the server receives this message, unmarshalls the parameters, executes the requested service and sends the result back to the client. The RPC middleware supports synchronous communication in which an RPC client is blocked until the remote procedure has been executed or an error occurs. Partial synchronization decoupling is achieved by using the RPC style between distributed processes (i.e., semi asynchronicity). The RPC has a good heterogeneity support but limited scalability" [74].

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. MOM loosely couples applications, in contrast to tightly coupled point-to-point integration. Employing middleware architectures reduces interapplication links and reduces application maintenance. Such architectures also enable the addition of new applications with minimum impact on existing ones. One tradeoff is the overhead associated with installing and maintaining the middleware itself" [58]. "MOM is analogous to email in the sense it is asynchronous and

requires the recipients of messages to interpret their meaning and to take appropriate action" [2]. "MOM supports both synchronous (via message passing) and asynchronous (via message queuing) communication. Asynchronous communication is achieved in the natural way. The message is sent to a server, without blocking a client. The client does not need to wait for a reply and can proceed with other actions" [74].

4. Object/component middleware (e.g., COBRA, Java RMI, and Microsoft COM/D-COM 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: an object remains passive until a principle performs an operation on it. This kind of model is adequate for a local area network (LAN) with a small number of clients and servers, but it does not scale well to large networks like the Internet. The main reason is that the request/reply model only supports one-to-one communication and imposes a tight coupling between the involved participants because of the synchronous paradigm. The component middleware supports both synchronous and asynchronous communication with limited scalability" [27].

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 by offering many advantages such as expressivity, ease of debugging, and most importantly high performance. MPI is a specification and a library which provides the infrastructure for communications among several parallel computational processes. MPI gives system designers the freedom to implement their own protocols that best fit their systems' requirements [35, 20].

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. This allows the message to be decoupled from, and transmitted over, any appropriate transport protocol. It also allows the message to be decoupled from explicit point-to-point connection protocols. The SOAP protocol is purposefully constructed to be extensible. Important extensions include reliability, security, and addressing. SOAP however specifies both interfaces and message structure so broad interoperability can be achieved. This is achieved at a performance cost that we return" [24].

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. However, the MPI model offers many design choices, which requires another level of requirements analysis, as discussed in the following section.

4.3 Refined MPI Communications Requirements for the ICAM System

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 [36].

Having described the communication design options available in the MPI library and according to the high performance MPI recommendations [34], 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.

4.4 Reactive Agent Software Implementation

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) [80].

Figure 4.4 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 [36], and to meet the blackboard functionality described in the behavioral model of the ICAM system.

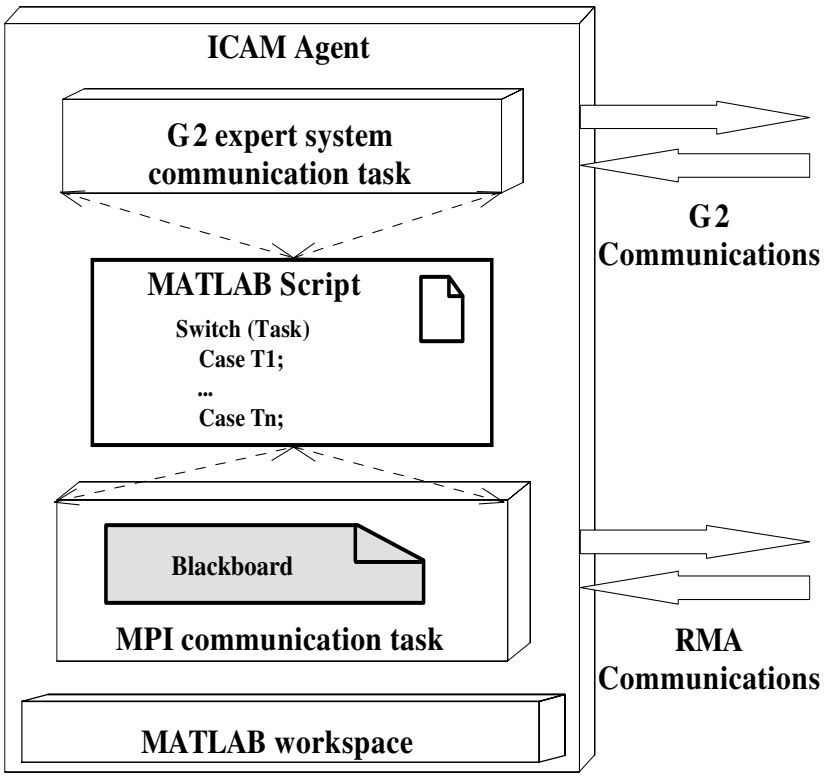

Figure 4.4: ICAM system reactive agent deployment structure

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 4.5 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.

Once the computations are done, the communication tasks are executed based on

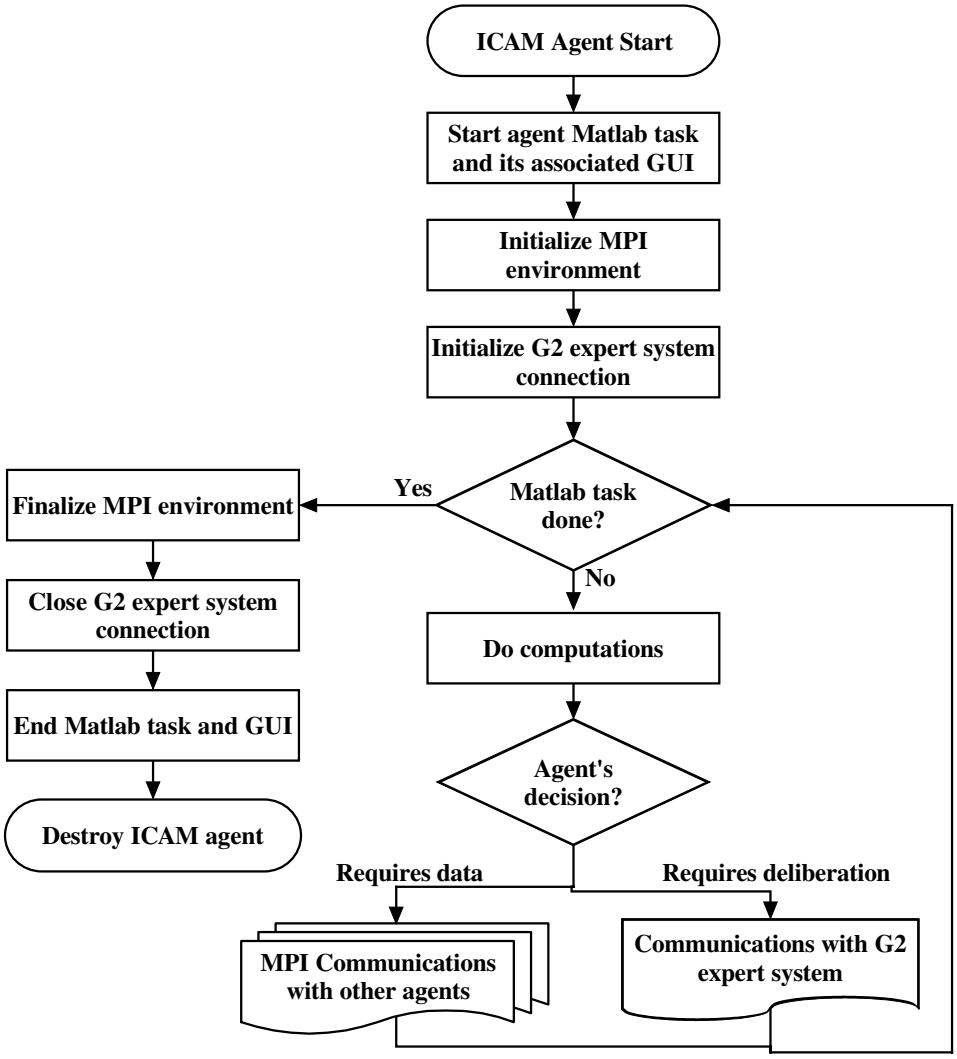

Figure 4.5: Reactive ICAM agent implementation flow chart

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.

4.5 Conceptual ICAM System Deployment Requirements

Having analyzed the artificial intelligence, the communications, and the reactive agents' implementation requirements of the ICAM system, it is crucial to analyze the system deployment requirements in terms of the required software development tools and hardware setting. Figure 4.6 illustrates the deployment scheme of the ICAM system, which facilitates its development, validation, and operation in real-time simulation mode and in real-world industrial applications. The deployment scheme includes some of the ICAM system agents for simplicity, and can be easily extended by adding more reactive agents. The backbone of the ICAM system is a Windows Server 2003 based local area network (LAN), which consists of the following essential servers:

- A DHCP server that offers dynamic configuration of IP addresses and related information to DHCP-enabled clients (i.e., agents) of the ICAM system. It also enables the automatic, centralized management of IP addresses and other TCP/IP configuration settings for network clients.

- A domain name space (DNS) server that maintains information about a portion of the DNS database and that responds to and resolves DNS queries. The DNS database is a hierarchical, distributed database that contains mappings of DNS domain names to various types of data of the network computers, such as IP addresses.

- A windows name resolution service (WINS) server for network basic input/out-

put system (NetBIOS) names. WINS is used by hosts running NetBIOS over TCP/IP (NetBT) to register NetBIOS names and to resolve NetBIOS names to Internet Protocol (IP) addresses.

- An active directory: stores information about objects (e.g., computers, users, groups, etc...) on a network and makes this information available to users.
- A domain controller: a server that contains a writable copy of the Active Directory database, participates in active directory replication, and controls access to network resources.
- A file server which is responsible for the central storage and management of data files so that other computers on the same network can access the files.
- An MPI server which provides access to the MPI library and manages the MPI environment of the ICAM system.
- A G2 expert system shell server which manages the G2 connections with the ICAM system reactive agents and runs the main knowledge base of the ICAM system.
- A Firewall that provides a security system for the flow of network traffic, usually to prevent unauthorized access from outside to an internal network.
- A router which is a device that connects networks using different communications protocols so that information can be passed from one network to the other. The router connects the ICAM system LAN to the internet.
- An Ethernet hub which is a common connection point for devices in a network; typically used to connect segments of a local area network (LAN).

The conceptual ICAM system network also contains five PCs, four of which represent some of the ICAM system reactive agents, including: the pilot plant simulation

agent, the data statistical processing agent, the model ID agent, and the FDIA agent. Each of these machines would include the basic software tools, which enable the specified reactive agent to function properly in the system. These tools include: the MPI high performance communication library, the G2 standard interface (GSI) library, MATLAB, and the reactive agent itself with its associated middleware task. The fifth machine is designated for connection with the real-world industrial plant. This machine would include the same software tools as the other four in addition to an OLE for process control (OPC) client and an OPC data hub for connection over the internet.

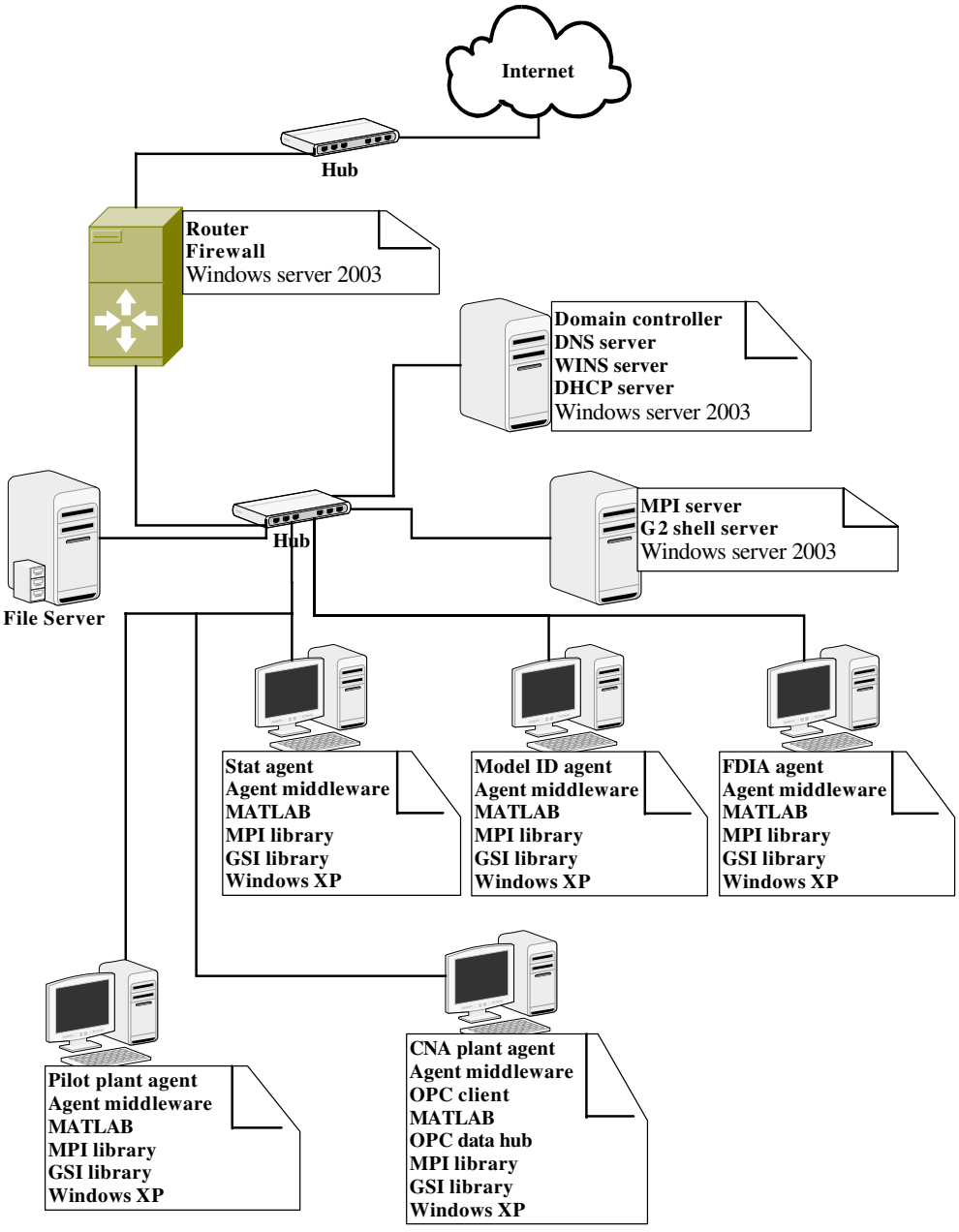

Figure 4.6: ICAM system deployment scheme

Chapter 5

ICAM System Prototype Design

The detailed design of a simplified ICAM system prototype is described in this chapter. We will discuss each agent's design in terms of its control flow (i.e., flow chart), its components, and the techniques used to build its components.

5.1 The ICAM System Prototype

A prototype has been developed in order to have the ICAM system requirements deployed in a real-world system. Figure 5.1 portrays the simplified ICAM system prototype. Real-time data from the external plant or a simulation model are received by the statistical data monitoring agent, which preprocesses the data by removing undesired discrepancies such as outliers and missing data. Processed data are stored in a real-time database for logging and other purposes, and 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.

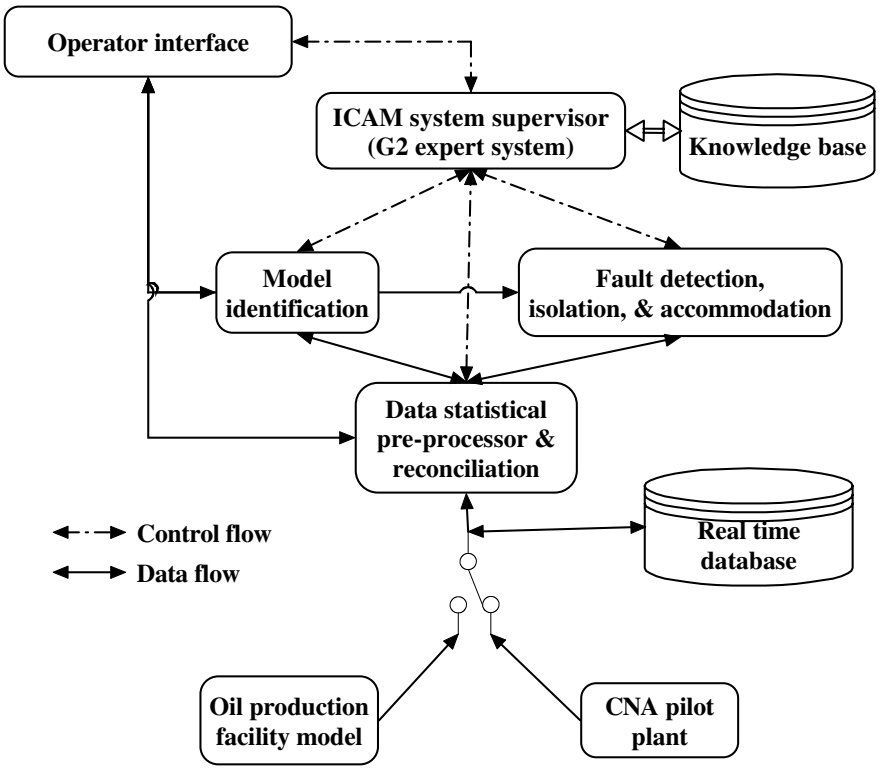

Figure 5.1: ICAM system prototype

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

- the middleware layer, which provides data communication between reactive agents and event communications with the supervisory agent,
- the artificial intelligence (AI) layer (i.e., the supervisory agent), which coordinates the behavior of the reactive agents to achieve robust performance against the external environment dynamic changes, and
- the reactive agents layer, which represents the different system computational and data processing functionalities (e.g., fault detection and isolation, process

modeling, etc...). Reactive agents are implemented as MATLAB scripts for ease of development and design (refer to chapter 4).

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

5.2 The Middleware Layer Design

The remote memory access (RMA) communication approach, which separates the communication of data from sender to receiver from the synchronization of sender with receiver, divides into two categories. The first one is active target communication, where data are moved from the memory of one process to the memory of another, and both are explicitly involved in the communication. This communication pattern is similar to message passing, except that all the data transfer arguments are provided by one process, and the second process only participates in the synchronization. The second category is passive target communication, where data are moved from the memory of one process to the memory of another, and only the origin process is explicitly involved in the transfer. This communication paradigm is closest to a shared memory model, where shared data can be accessed by all processes, irrespective of location [36].

In active target communication, a target window can be accessed by RMA operations only within an exposure epoch. Such an epoch is started and completed by RMA synchronization calls executed by the target process. Distinct exposure epochs at a process on the same memory window must be disjoint, but such an exposure epoch may overlap with exposure epochs on other windows or with access epochs for the same or other windows arguments [36]. There is a one-to-one matching between access epochs at origin processes and exposure epochs on target processes. RMA operations issued by an origin process for a target window will access that target window during the same exposure epoch if and only if they were issued during the

same access epoch. In passive target communication the target process does not execute RMA synchronization calls, and there is no concept of an exposure epoch. We have chosen the active target communication RMA communication type to achieve high reliability.

Figure 5.2 illustrates the general synchronization pattern for active target RMA communication of the ICAM system prototype, where four RMA communication channels are designed to transfer raw data, processed data, the accommodation parameters and the plant state space model to the corresponding agents respectively. The synchronization on the first channel occurs between the access epoch at the pilot plant agent (origin agent) and the exposure epoch at the statistical processing agent (target agent). This ensures that the raw data message is transferred reliably. The raw data message is converted from/to the MATLAB type to/from the MPI type before/after the communication task. The synchronization between complete and wait ensures that the put call of the origin agent completes before the memory window is unexposed with the wait call.

Once data have been preprocessed by the statistical agent, they are converted to the MPI type to prepare them for communication on the second MPI channel. An access epoch is started at the statistical processing agent to synchronize with the model ID and FDIA agents. If the model ID and FDIA agents have started their exposure epochs, then the processed data message is transferred to their memory windows. Once the communication on this MPI channel is completed at the statistical processing agent, the exposure epochs at the target agents are finalized, and the processed data message is converted to the MATLAB type again for further processing.

If the model ID agent has identified a new model of the pilot plant, it sends the new model message to the FDIA and the statistical processing agents on the third MPI channel. The conversion and transfer procedure is similar. If a sensor/actuator

fault occurs and the system starts the fault accommodation task, a fourth MPI channel is designed to transfer the real-time accommodation data from the FDIA agent to the pilot plant agent. Again, the conversion and transfer procedure is similar.

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

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

Figure 5.3 illustrates the G2 communication sequence of the ICAM system prototype, where the G2 supervisory agent has four rules to update each reactive agent state in its knowledge base. The rules are fired asynchronously every one second by passing a request to the specified reactive agent. Once the update procedure is executed in the reactive agent side, the reactive agent replies by sending its updated state (structure object) to the supervisory agent.

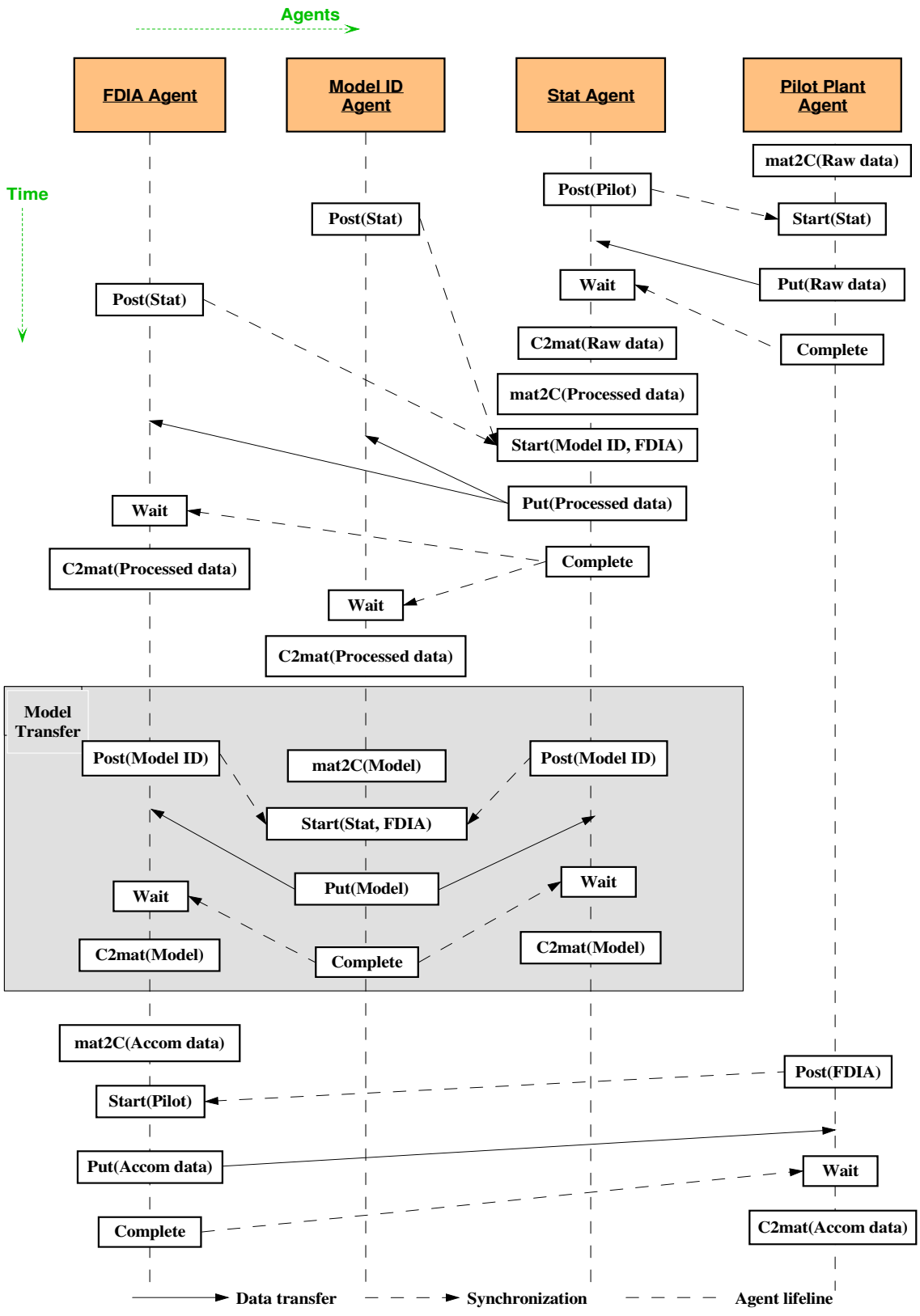

Figure 5.2: ICAM system prototype MPI communications sequence

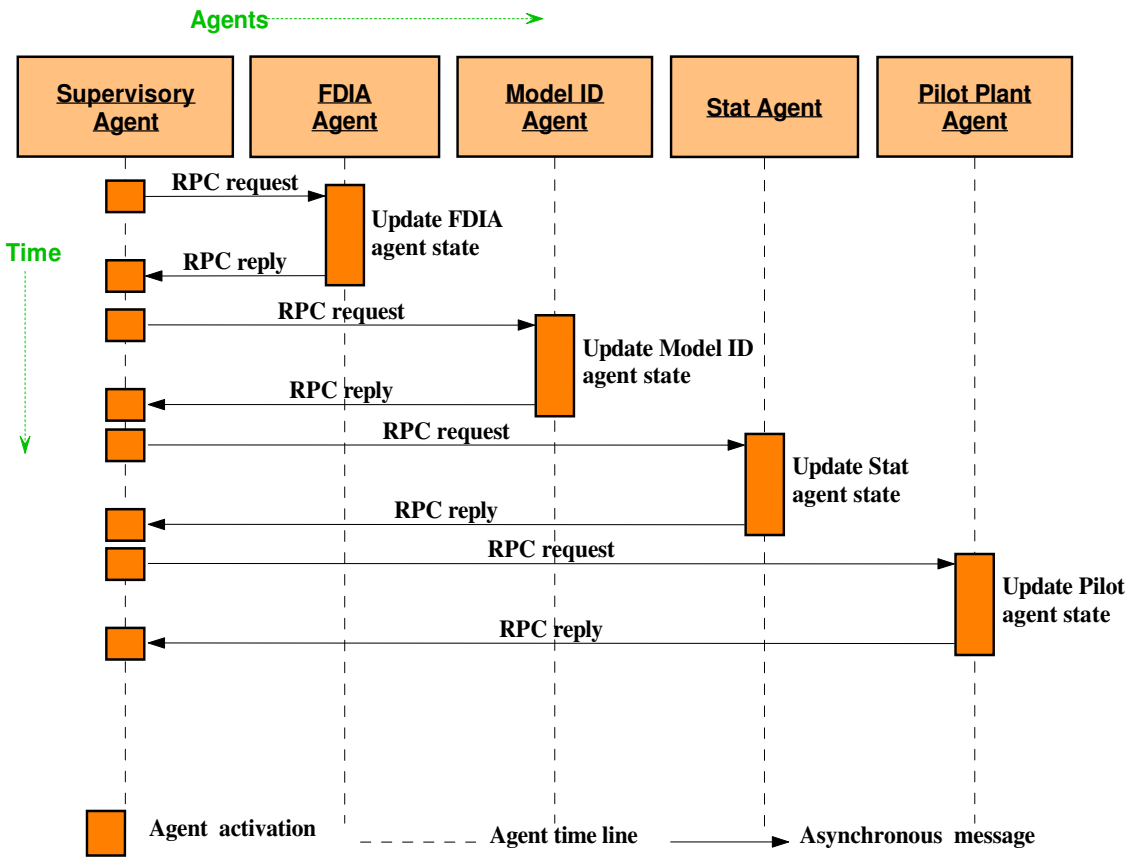

Figure 5.3: ICAM system prototype G2 communications sequence

5.3 The Supervisory Agent Design

The supervisory agent is implemented based on the G2 real time expert system shell, which codifies the ICAM system internal and external behavior in its knowledge base [31]. The supervisory agent contains an ontology that represents the different agents of the ICAM system prototype. Each agent in the ontology has its own attributes which represent the agent technical characteristics and methods which represent the agent's behavior. The ICAM system prototype is represented as a logical connection of the corresponding reactive agents, as illustrated in figure 5.4. Each reactive agent has two connections; the first is the data MPI connection, and the second is the G2 connection with the supervisory agent. The META Agent object, which encapsulates the supervisory knowledge base of the ICAM system prototype, is a representation of the supervisory agent. The ENV Agent object represents the pilot plant model agent, the STAT Agent object represents the statistical preprocessing agent. Likewise, the M-ID Agent and PSV Agent objects represent the model identification agent and the FDIA agent, which exploits the parity space vector (PSV) FDI approach [104, 70, 71, 69, 72]. The attributes of each reactive agent represent the MPI and G2 connections characteristics, the reactive agent internal state, and its response to changes in the pilot plant (i.e., the external environment).

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

are discussed in the following sections.

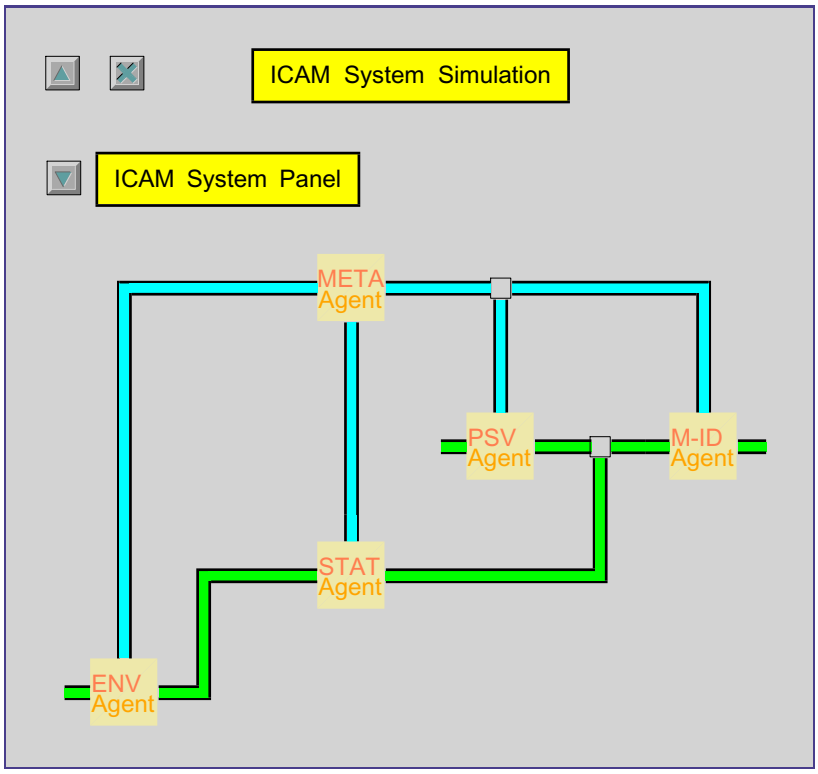

Figure 5.4: ICAM system prototype representation in the G2 supervisory agent

5.3.1 ICAM system ontology design

An ontology is important for knowledge-based system development, as it acts as a software specification. Figure 5.5 illustrates the ontology of the ICAM system prototype, where the different agents of the ICAM system are represented as a class hierarchy. The class hierarchy includes a class for starting up the system, a class for linking the supervisory with the other reactive agents, and two graphical connection classes to show the status of the MPI and G2 communication channels of each reactive agent.

The root class in the main class hierarchy is the ICAM agent, from which other reactive agents inherit their main attributes. The meta-manager class is responsible for executing the master rule-base of the ICAM system prototype. Each reactive

agent of the ICAM system is represented by a class, including a statistical preprocessing agent, a model identification agent, a pilot plant simulation agent, and an FDI agent. The ICAM system ontology supports scalability. For example, the FDI agent may have two types: qualitative- and quantitative-based FDI agents; a signed digraph (SDG) based FDI agent would be a qualitative FDI agent, and a general parity vector space based FDI agent is a quantitative FDI agent. The model identification agent may have different types such as a recursive least square agent or a subspace model ID agent. The ontology is also open to have new types of agents such as an optimization agent that may, in turn, have different types. Once the ontology is designed, the ICAM system representation in the supervisory agent is then designed as an interconnection of the system agents, as shown in figure 5.4.

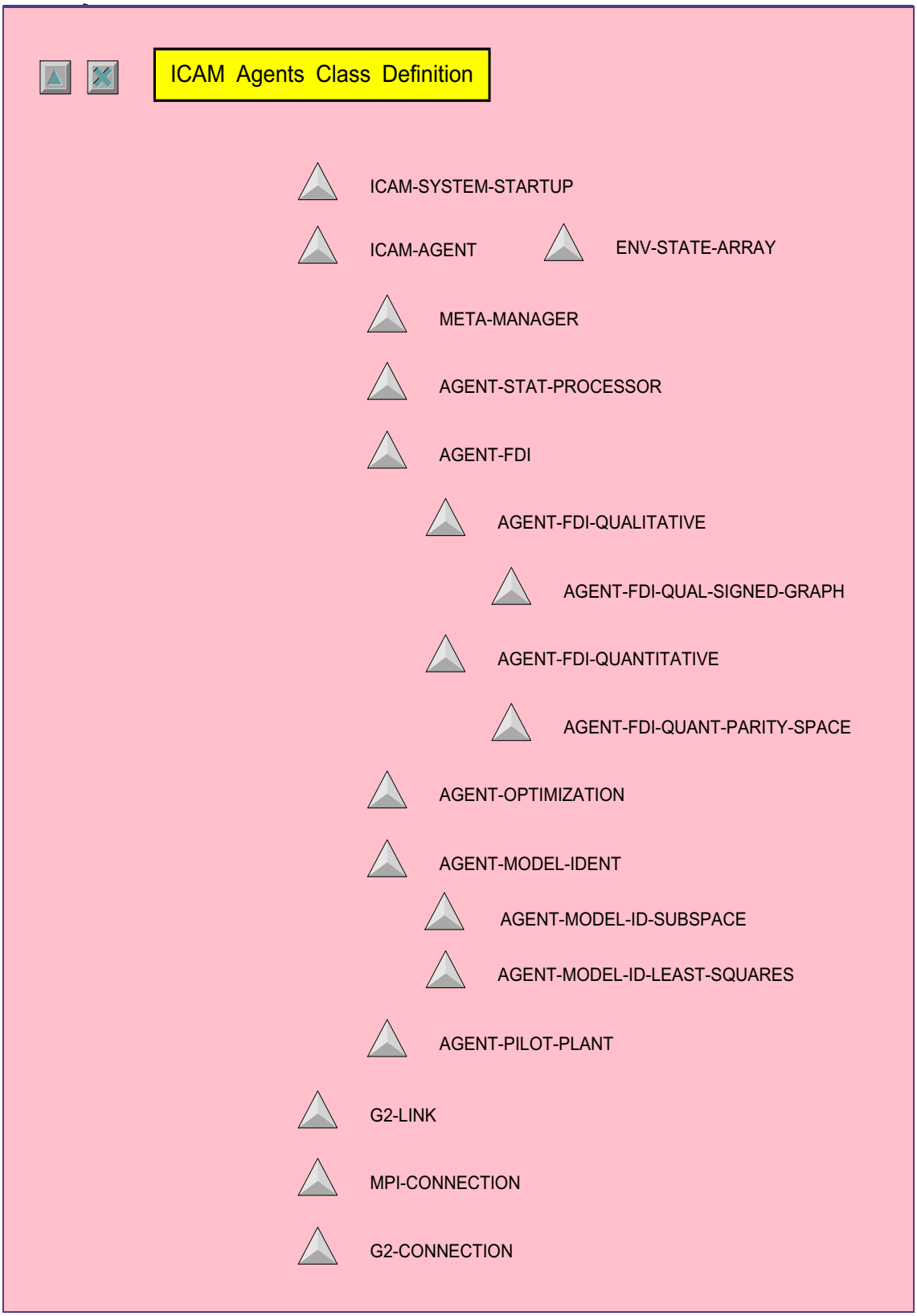

Figure 5.5: ICAM system ontology

5.3.2 The supervisory agent rule-base design

Since the supervisory agent of the ICAM system coordinates its internal and external behavior, it is crucial to carefully design the rule base of the supervisory agent to achieve robust system performance. The rule base codifies the desired system behavior in response to external environment dynamic changes and to process operator interactions. Figure 5.6 illustrates the ICAM system prototype event sequence diagram, which is embedded in the supervisory agent rule base. This event sequence was developed by the UNB PAWS team (i.e., Sayda, Omana, and Moreno) in group meetings led by Dr. Taylor. The rule-base design process is in its preliminary stage; it will be further developed to address more complex situations in future work. The ICAM system supervisory agent starts up the other reactive agents (i.e., event # 1), which are implemented as MATLAB functions and scripts for ease of development and debugging [81].

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

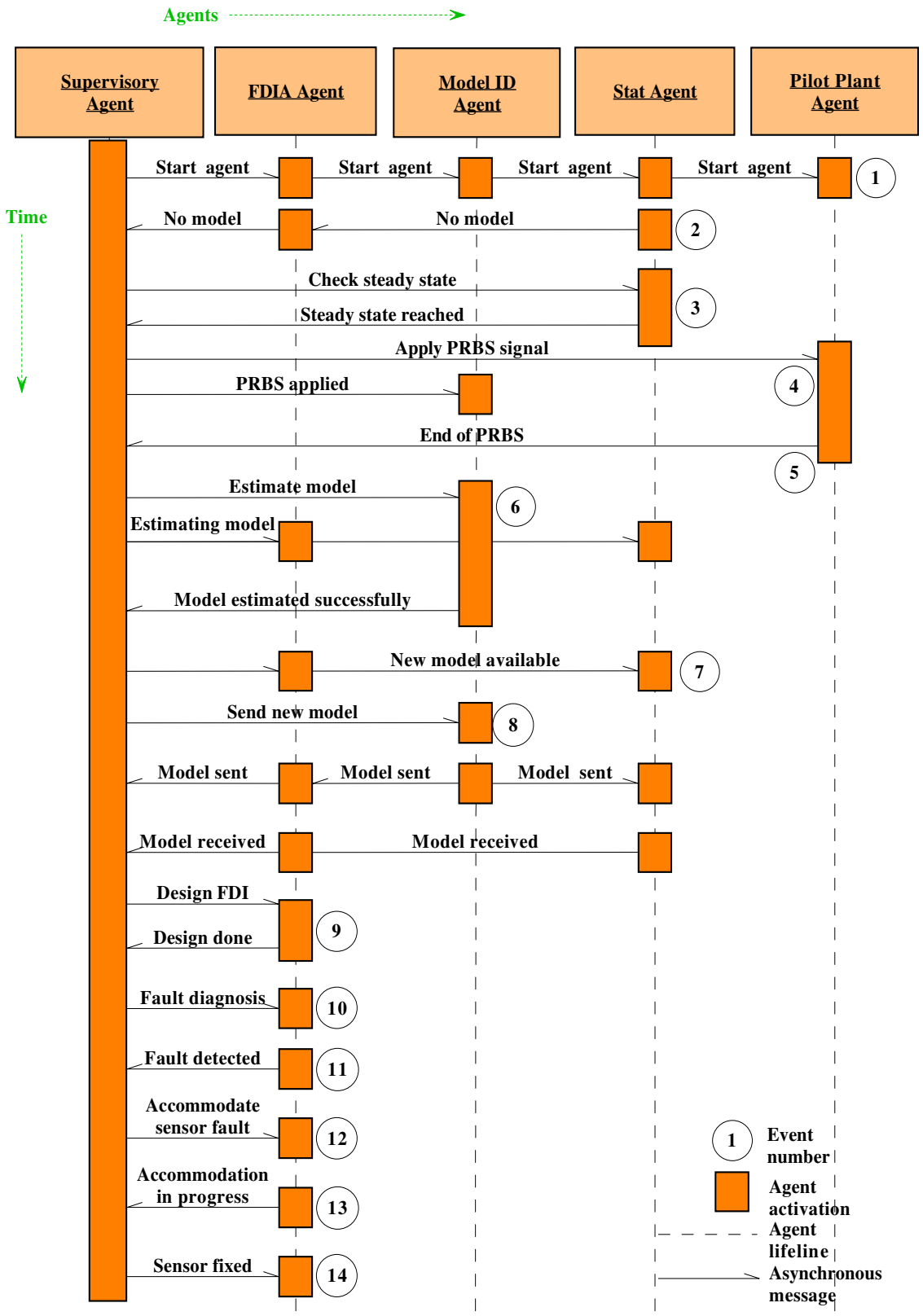

Figure 5.6: ICAM system prototype event sequence

The estimated model is then sent to the appropriate agents (event # 8), which, in turn, report the model reception status back to the supervisory agent. The supervisor then requests that the FDIA agent design the FDI filter based on the received process model (event # 9). The FDIA agent starts monitoring the external process for sensor and actuator faults (event # 10) [71, 72]. If the FDIA agent detects a fault in the plant, the fault location, type, time, size and other information are reported back to the supervisor for further processing (event # 11). In the case of a sensor fault, the FDIA agent will also recommend the appropriate accommodation (correction) (event # 12) [94]. The sensor accommodation status is reported continuously to the supervisory agent (event # 13), which terminates the sensor accommodation task if the sensor has been fixed (event # 14).

5.4 Design of Reactive Agents

The simplified ICAM system prototype consists of four reactive agents, namely, the pilot plant simulation agent, the data statistical preprocessing agent, the model identification agent, and the FDIA agent, whose design is discussed in the following sections. The interactions of the supervisory agent with the reactive agents are indicated on their flow charts by the event numbers, which have been discussed in the previous section (refer to figure 5.6).

5.4.1 The pilot plant agent design

The pilot plant model represents an oil production facility, which separates oil well fluids into crude oil, sales gas, and water. The simulation model basically consists of two processes, as illustrated in figure 5.7. The first is a two-phase separator in which hydrocarbon fluids from oil wells are separated into two phases to remove as much light hydrocarbon gases as possible. The produced liquid is then pumped to the three-phase separator (i.e., the second process), where water and solids are separated from oil. The produced oil is then pumped out and sold to refineries and

petrochemical plants if it meets the required specifications. Gas is processed further and sent as sales gas.

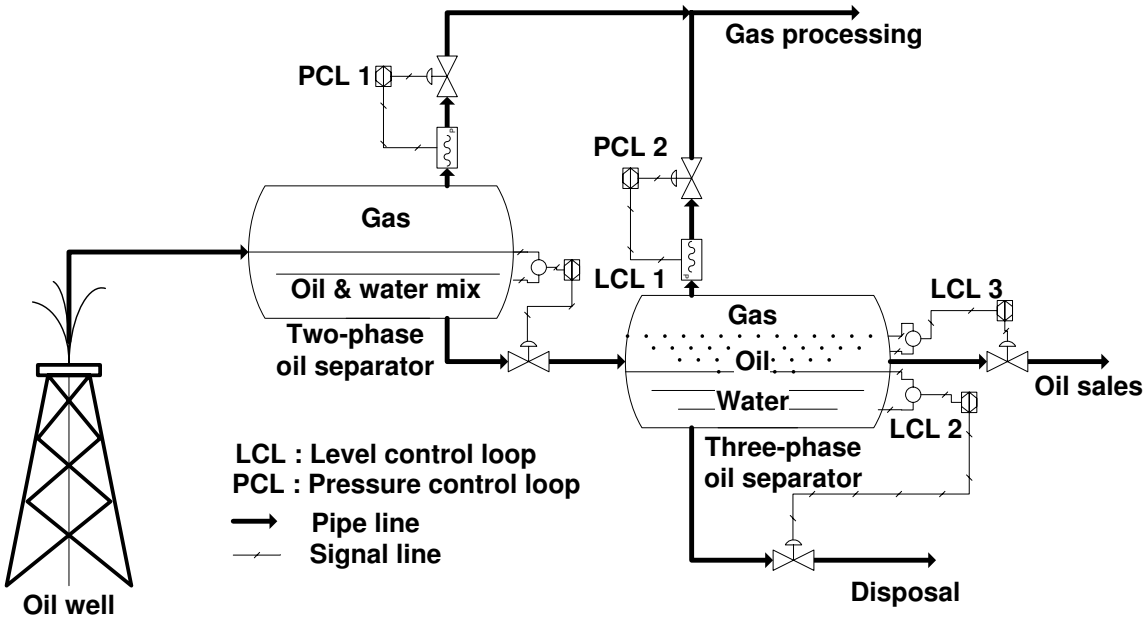

Figure 5.7: Oil production facility P&ID

The two separation processes of the simulation model are controlled to maintain the operating point at its nominal value, and to minimize the effect of disturbances on the produced oil quality. As shown in figure 5.7, the first separation process is controlled by two PI controller loops. In the first loop, LCL1, the liquid level is maintained by manipulating the liquid outflow valve. The second loop, PCL1, controls the pressure inside the two-phase separator by manipulating the amount of the gas discharge. The second separation process has three PI controller loops. An interface level PI controller, LCL2, maintains the height of the oil/water interface by manipulating the water dump valve. The oil level is controlled by the second PI controller, LCL3, through the oil discharge valve, and the vessel pressure is maintained constant by the third PI loop, PCL2 [80] (refer to appendix A).

The plant ordinary differential equation (ODE) model was simulated in real time every $\delta t = 100$ milliseconds using the fixed-step first-order Euler ODE solver. Ten

sensor and actuator faults were embedded in the plant model to validate the ICAM system performance and logical behavior during faulty situations. Figure 5.8 illustrates the flow chart of the pilot plant simulation model agent. After the supervisory agent starts up the plant agent, the plant agent initializes its MPI and G2 communication links, and it enters in a wait loop till it receives a valid scenario to run. Four scenarios are embedded in the plant agent: the first is the default scenario which runs the plant at its nominal operating point; the second scenario allows the set points of the plant to be changed from the nominal operating point; the third scenario applies a disturbance to the plant; and the final scenario simulates the plant during sensor and actuator faults.

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

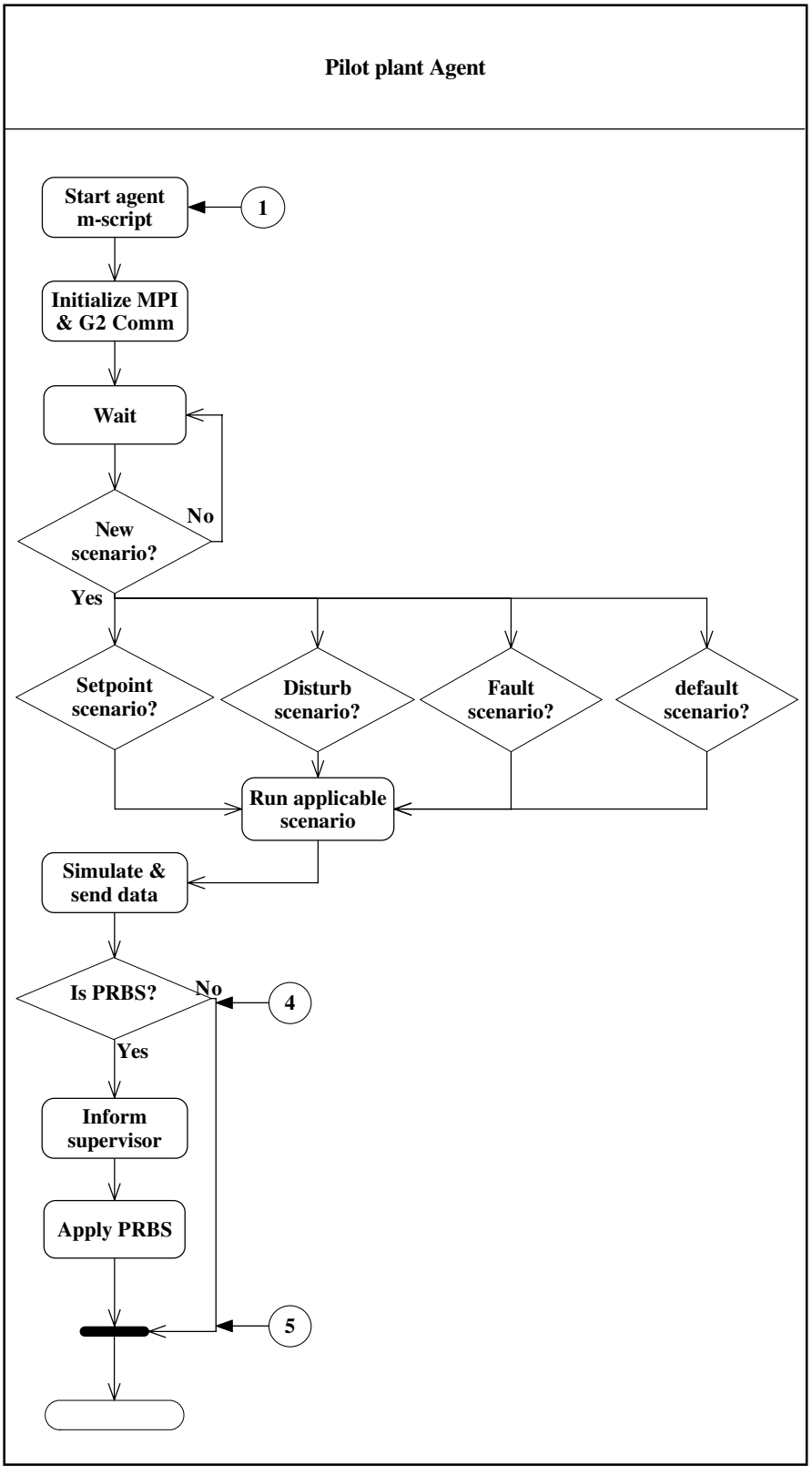

Figure 5.8: The pilot plant simulation agent flow chart

5.4.2 The statistical pre-processing agent design

Figure 5.9 illustrates the flow chart of the data statistical pre-processing agent, where the agent is started up by the supervisory agent (event # 1), and its G2 and MPI communication links are initialized. Once the statistical preprocessor agent receives raw data from the plant agent, it stores them in a data window for further processing. If the agent receives a request from the supervisory agent to check the steady state status of the plant agent (event # 3), it checks the steady state and report the result back to the supervisory agent. The statistical pre-processing agent informs the supervisory agent that it does not have any plant model after its startup (event # 2).

If there is a new plant model available, the supervisory agent informs the statistical pre-processing agent to update the plant model (event # 7). The agent then removes missing data and outliers by exploiting the median absolute deviation algorithm [60]. The data are then reconciled according to a pre-specified material balance for quality control (this functionality may be implemented in future work), and are sent to other agents for further processing. The processed data are also sent to a graphic user interface (GUI) agent to allow plant operators to interact with processed data (e.g., zoom, store, and plot the data) in a friendly manner. Should the statistical pre-processing agent fail internally, it reports its failure mode status to the supervisory agent for further actions. Internal failure could happen during sensor/actuator faults, which leads to an ill-posed optimization problem to solve in the data reconciliation task (to be implemented in future work).

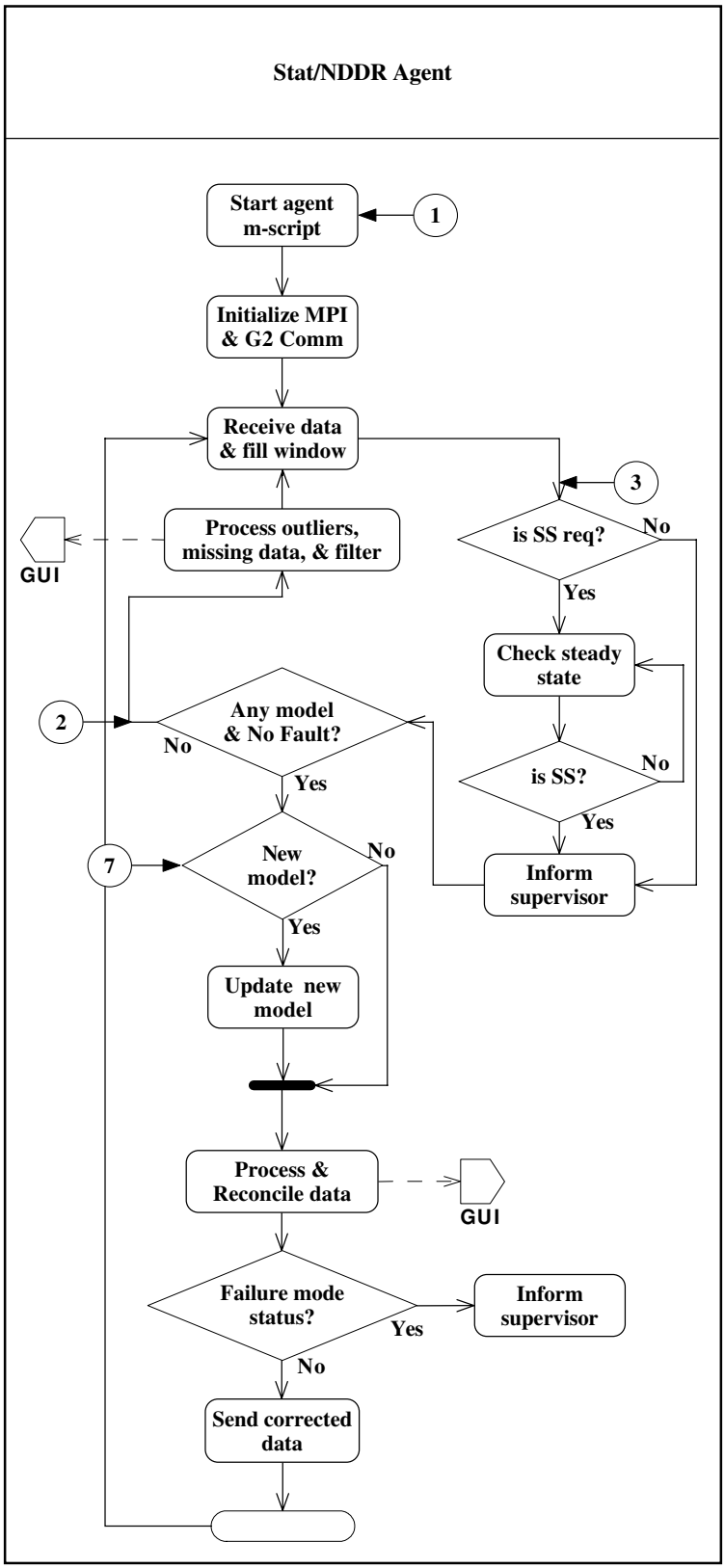

Figure 5.9: The statistical pre-processing agent flow chart

5.4.3 The model identification agent design

The model ID agent estimates the multivariable plant model by using the subspace method, which uses the canonical variable algorithm (CVA) in its singular value decomposition stage [99, 51, 56]. Figure 5.10 illustrates the flow chart of the model identification agent, where the agent is started up by the supervisory agent (event # 1). The model ID agent then initializes its own MPI and G2 communication links. The agent stays in an idle state, unless the supervisory agent informs it that a PRBS signal is being applied by the pilot plant simulation agent (event # 4). Consequently, the agent starts receiving processed data from the statistical processing agent.

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

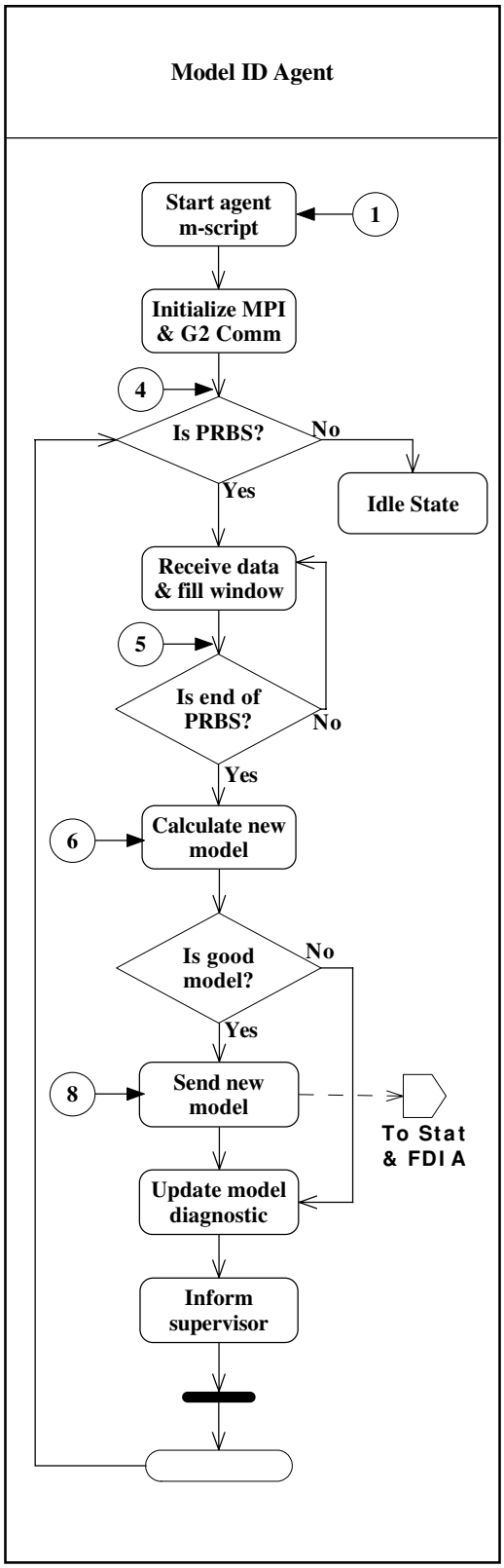

Figure 5.10: The model identification agent flow chart

5.4.4 The fault detection, isolation, and accommodation agent design

The FDI agent exploits the generalized parity space (GPS) to generate a set of directional residuals, from which process faults can be determined [104, 70, 71, 69, 72, 94]. After the supervisory agent starts up the FDIA agent (event # 1), it initializes its G2 and MPI communication links [94], as shown in figure 5.11. If the FDIA agent has no plant model, it reports that back to the supervisory agent for further actions (event # 2). When a new plant model is available (event # 7), the FDIA agent updates its knowledge about the plant. The supervisory agent then instructs the FDIA agent to design its FDI filter based on the newly updated plant model (event # 9). When that task is completed, the FDIA agent then starts to monitor the plant data to detect any sensor/actuator faults (event # 10).

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

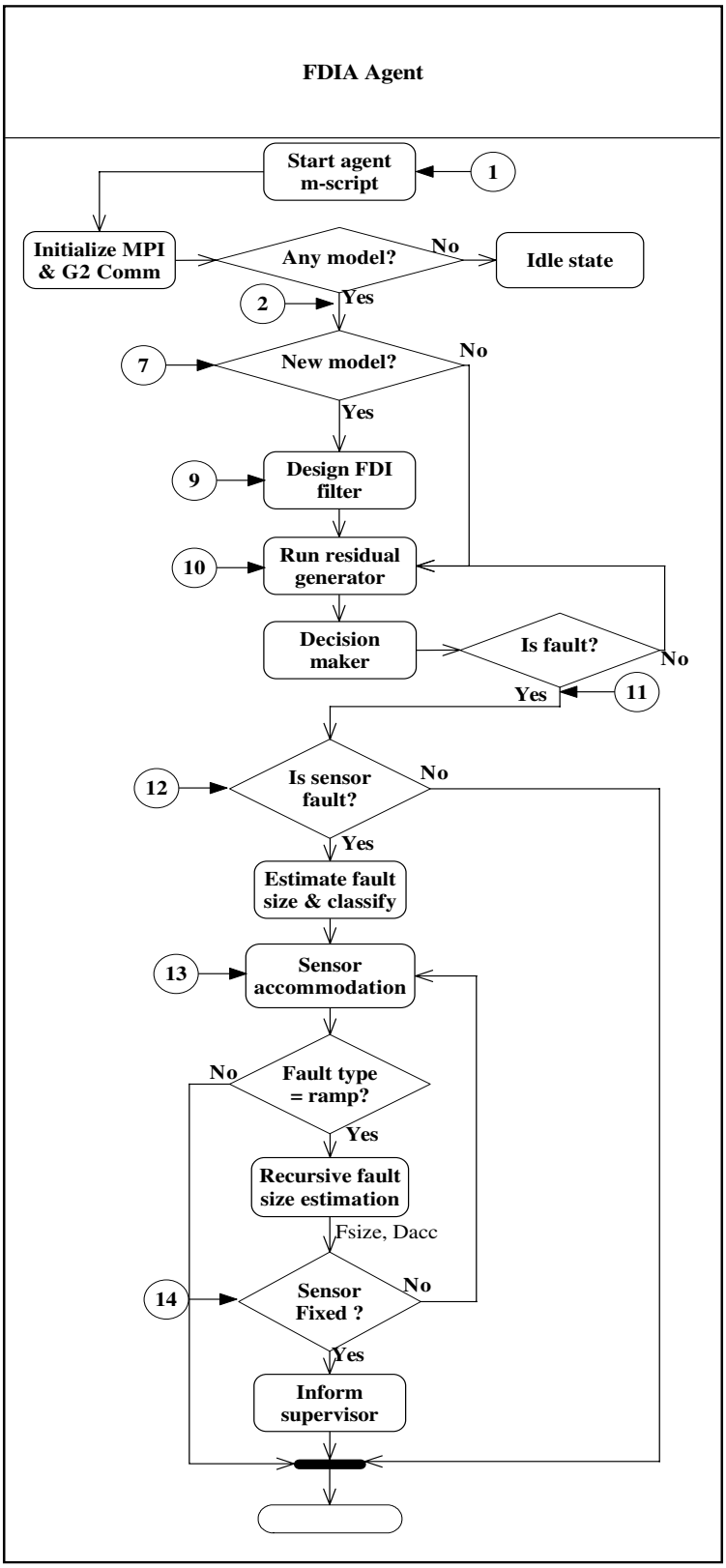

Figure 5.11: The FDIA agent flow chart

5.5 ICAM System Prototype Deployment Scheme

Having discussed the ICAM system prototype design with respect to the artificial intelligence, the middleware, and its reactive agents' requirements, it is crucial to design the prototype deployment scheme to verify its performance and logical behavior. Figure 5.12 illustrates the ICAM system prototype deployment scheme, which is simplified from the general ICAM system deployment scheme (refer to figure 4.6 in the previous chapter). The ICAM system prototype contains two Windows 2003 servers, which provide the local network infrastructure of the system. The prototype also consists of two PC nodes, in which the five agents of the ICAM system prototype are deployed. The first node runs three agents, namely, the pilot plant simulation agent, the model ID agent, and the G2 supervisory agent. The second node runs the data statistical processing agent and the FDIA agent. Each reactive agent consists of its m-script, its middleware task, and its associated MATLAB session.

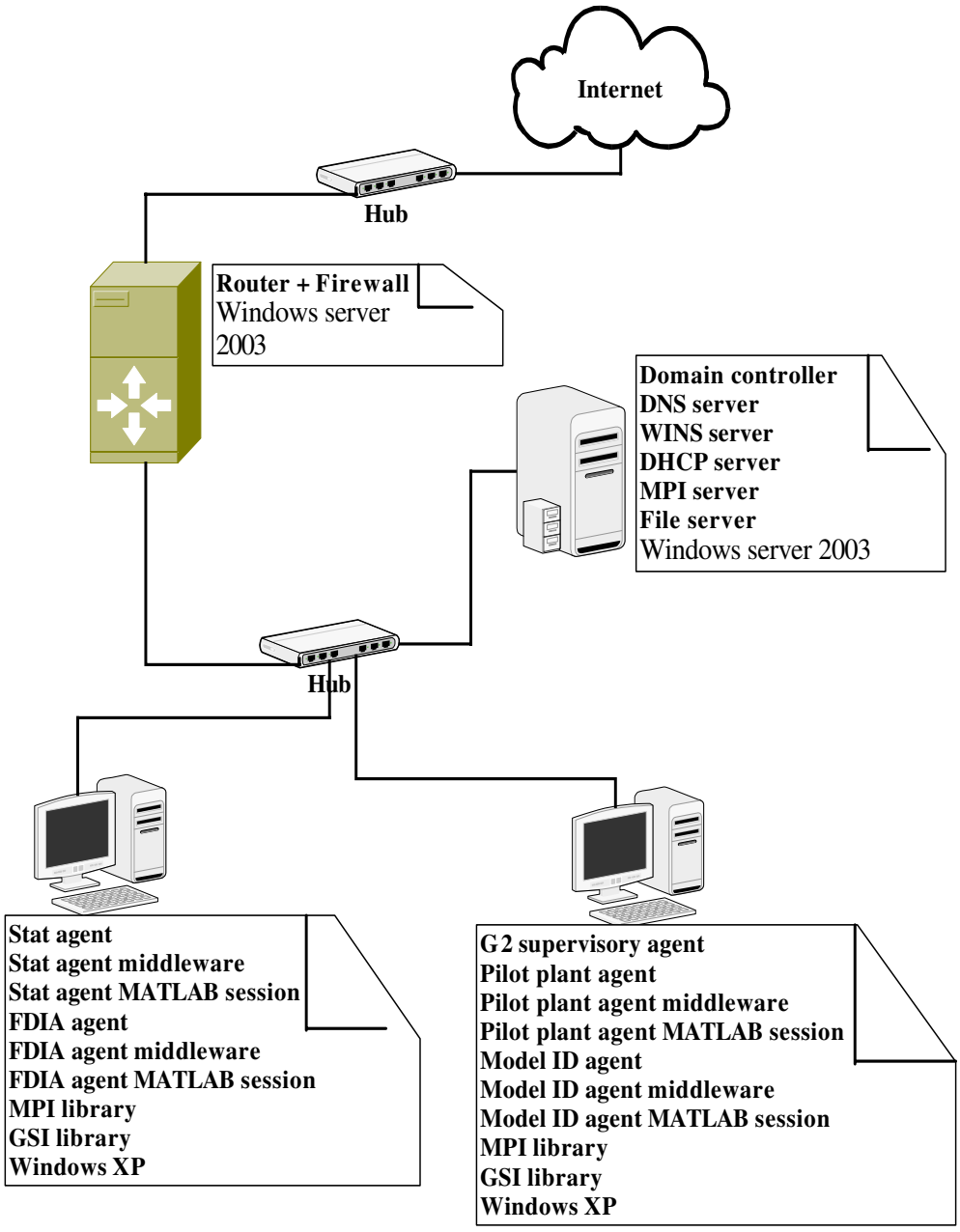

Figure 5.12: ICAM system prototype deployment scheme

Chapter 6

ICAM System Prototype Verification and Validation

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. The ICAM system prototype is deployed in a Windows 2003 network, which has two nodes (i.e., workstations). The first node has three running agents, namely the pilot plant agent, the model ID agent, and the supervisory agent. The second node has the remaining agents, namely the statistical preprocessing agent and the FDIA agent. The pilot plant simulation model corresponds to figure 6.1; it consists of 10 states, 5 manipulated variables, 5 controlled variables, and 17 auxiliary measured inputs and outputs (e.g., disturbances, product quality variables, etc.). Ten sensor/actuator faults are embedded in the pilot plant simulation agent to emulate faulty instrumentation in real-world oil production plants, as indicated in table 6.1. Five different simulation scenarios are applied during the real-time simulation experiments, three showing successful behavior and two revealing ICAM system prototype limitations.

Three simulation scenarios will demonstrate the ICAM system logical behavior and performance. In the first scenario (i.e., indicated by $\boxed{1}$ in figure 6.1), we will apply a bias fault in the three-phase separator water volume sensor. The second scenario will demonstrate the system behavior when a bias fault is applied in the

Fault number	Instrumentation name
F1	Faulty two-phase liquid volume sensor
F2	Faulty two-phase pressure sensor
F3	Faulty three-phase water volume sensor
F4	Faulty three-phase oil volume sensor
F5	Faulty three-phase pressure sensor
F6	Faulty two-phase separator liquid outflow valve
F7	Faulty two-phase separator gas outflow valve
F8	Faulty three-phase separator water outflow valve
F9	Faulty three-phase separator oil outflow valve
F10	Faulty three-phase separator gas outflow valve

Table 6.1: Oil production facility instrumentation faults

two-phase separator gas outflow valve (i.e., indicated by 2 in figure 6.1). The third scenario is a more sophisticated one, in which a ramp fault is applied in the two-phase separator liquid volume sensor (i.e., indicated by 3 in figure 6.1). In order to diagnose the system limitations, two additional simulation scenarios are applied. The first one is the application of a bias fault in the three-phase separator pressure sensor (i.e., indicated by A in figure 6.1). A decrease in oil production is emulated in the second scenario (i.e., indicated by B in figure 6.1).

We will discuss the behavior of each agent of the system in terms of its results and decisions in each simulation scenario. Furthermore, we will discuss the decisions made by the supervisory agent in each scenario. The system performance is analyzed in terms of the real-time performance during the computation and communication tasks of each agent. The network activity also is analyzed to see if it is consistent with the decisions made by the supervisory and the reactive agents of the system, and that no deadlocks are encountered when the agents communicate among each other.

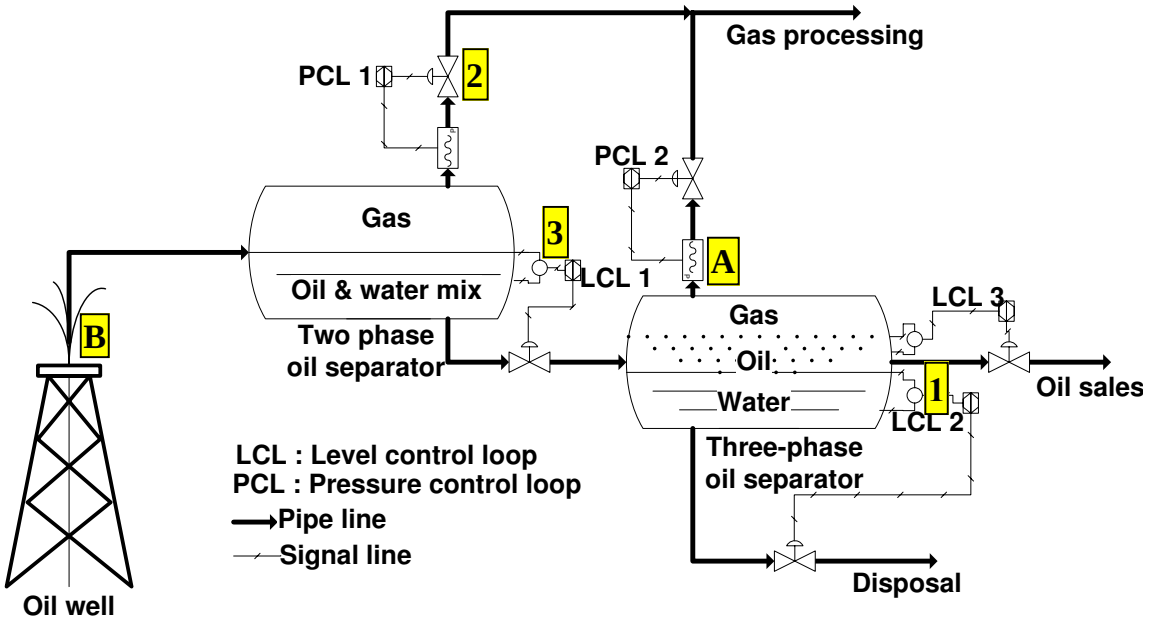

Figure 6.1: Oil production facility P&ID

6.1 Scenario 1: Faulty Water Volume Sensor in The Three-Phase Separator Sub-Process

The first simulation scenario is done by applying a +15% bias fault in the water volume sensor of the three-phase separator (F3; refer to control loop LCL2 in figure 6.1). After the ICAM system supervisory agent starts up executing its rule base, other reactive agents are started and initialized. The pilot plant agent starts its simulation at a nominal value of V = 146 ft^3, P = 625 PSI for the two-phase separation sub-process and V_{wat} = 77.5 ft^3, V_{oil} = 46.5 ft^3, P = 200 PSI for the three-phase separation sub-process. Outliers and missing data are applied to the two-phase separator measurements to emulate real-world data. Since the ICAM system has no knowledge about the pilot plant agent (i.e., no dynamic model), it sends a message to the statistical pre-processor to check if the pilot plant is in steady state.

Once it is in steady state, the supervisor then commands the control system of the pilot plant to apply a sufficiently exciting pseudo random binary (PRBS) signal with an amplitude of 2% about the nominal operating point. This allows the model

ID agent to collect enough data to identify the pilot plant model, after which the FDIA agent designs its FDI filter. Having gained new knowledge about the current dynamic behavior of the pilot plant, the ICAM system now can start monitoring the pilot plant for any instrumentation failure. If a sensor/actuator fault occurs the FDIA agent reports its decision to the supervisory agent. The supervisory agent in turn commands the FDIA agent to start the fault accommodation task, if applicable. The fault accommodation task is stopped if the sensor is fixed. The behavior of each agent during this scenario is discussed in the following sections.

6.1.1 The pilot plant agent behavior

The process variables are logged at the pilot plant agent during the first simulation scenario as indicated by figures 6.2, 6.3, 6.4, 6.5, and 6.6. Positive fixed-size outliers and missing data are applied to the two-phase separator measurements at random time instants (i.e., the liquid volume and the pressure measurement as shown by the top plots of figures 6.2, and 6.3). The pilot plant at first runs at its nominal operating point. Independent PRBS signals are applied to all the plant inputs to identify its model. Subsequently, a +15% bias fault is applied to the three-phase separator water volume sensor at time T_{fault} = 9:47:32, and the accommodation task starts at time T_{accom} = 9:48:44, as shown in figure 6.4. The PI controller in loop LCL2 (refer to figure 6.1) rejects the fault, as it is considered as a constant disturbance applied to the water volume sensor. However, the sensor measurement does not reflect the actual state of the water volume, as shown in the FDIA agent results. The effect of the faulty volume sensor on the oil volume and the gas pressure in the three-phase separator is also shown in figures 6.5 and 6.6.

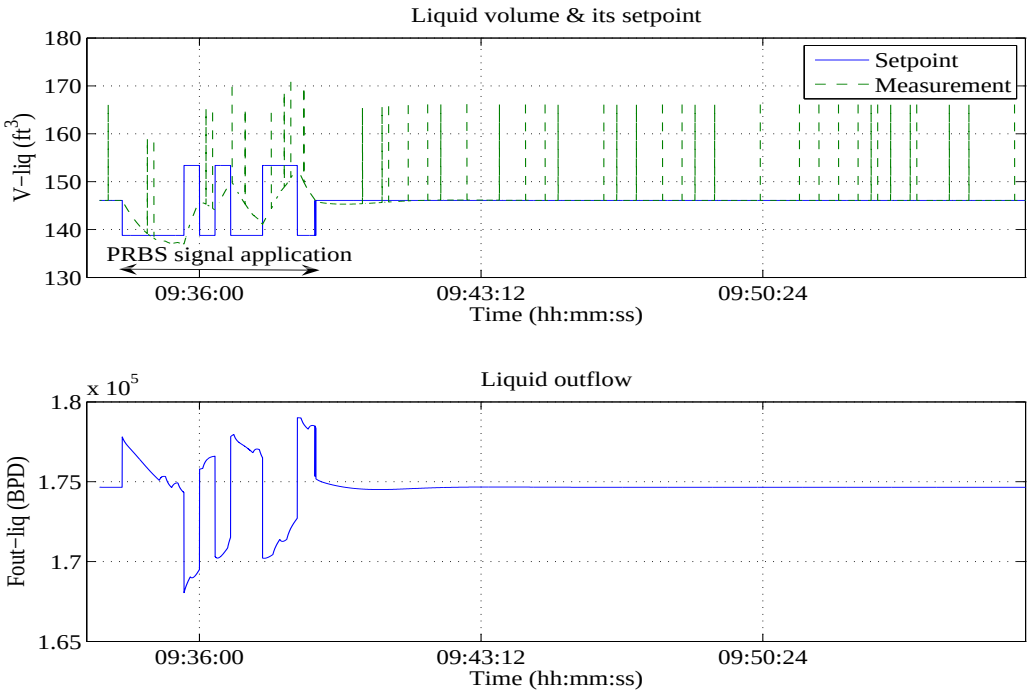

Figure 6.2: Scenario 1: Two-phase separator liquid volume logged by the pilot plant agent

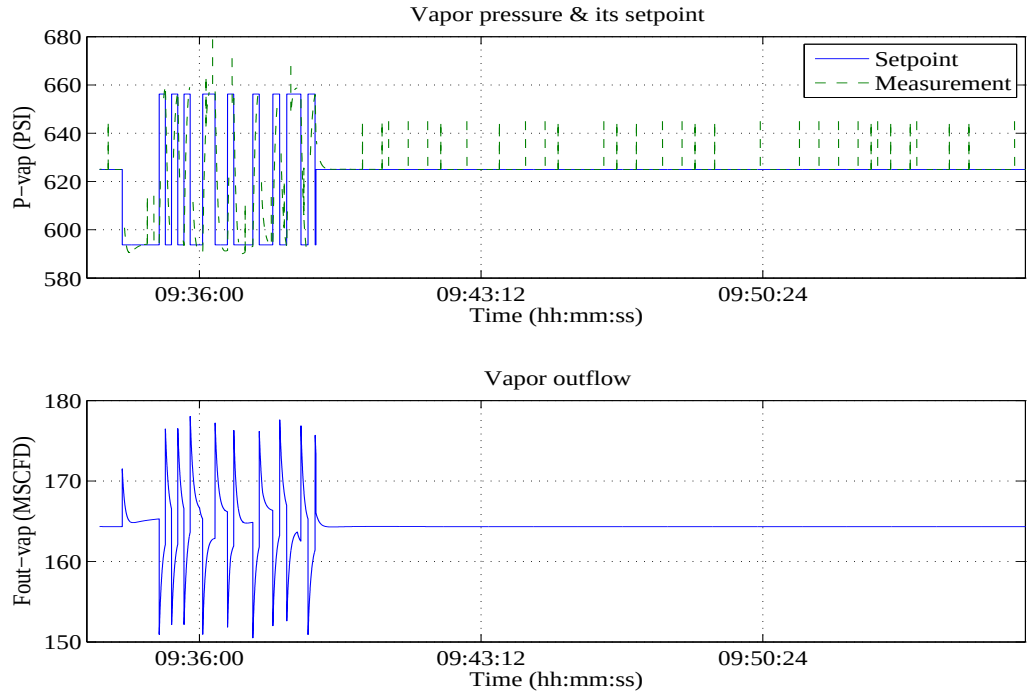

Figure 6.3: Scenario 1: Two-phase separator pressure logged by the pilot plant agent

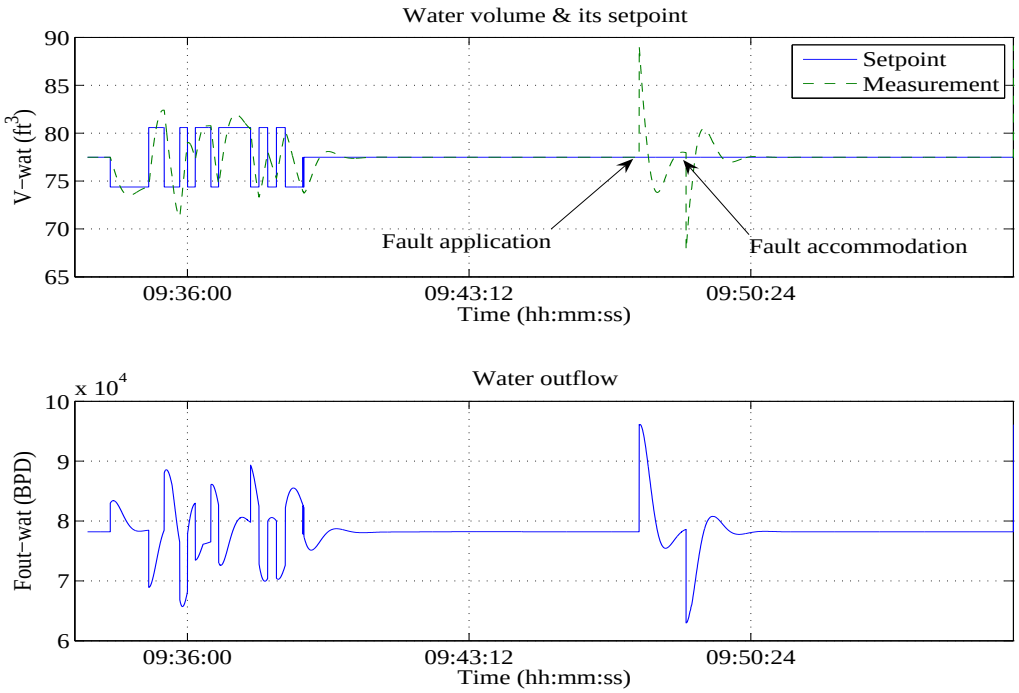

Figure 6.4: Scenario 1: Three-phase separator water volume logged by the pilot plant agent

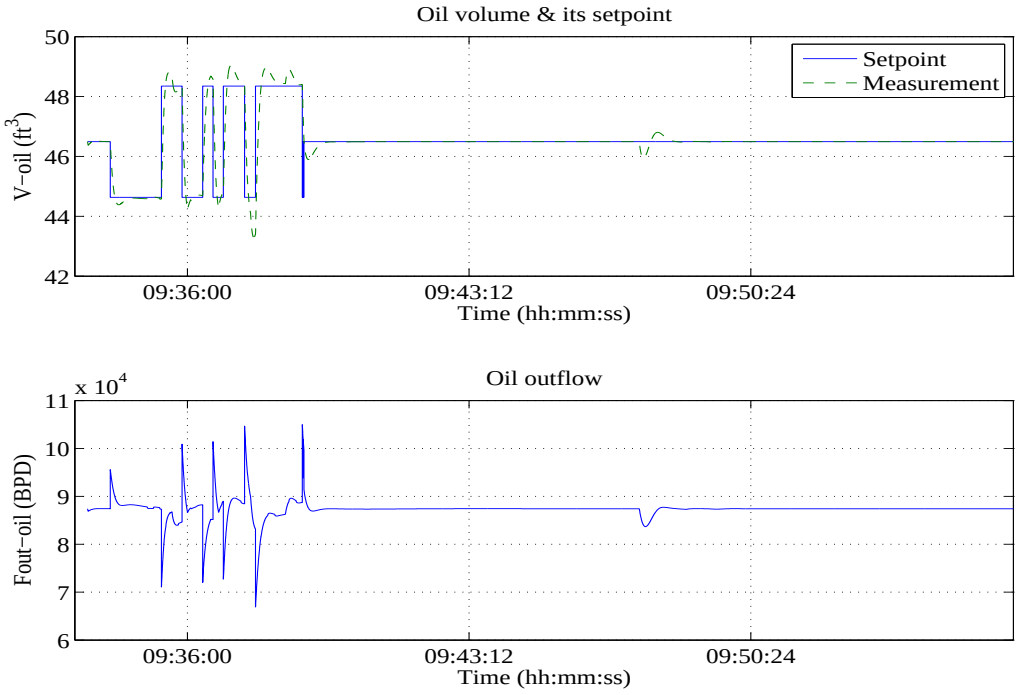

Figure 6.5: Scenario 1: Three-phase separator oil volume logged by the pilot plant agent

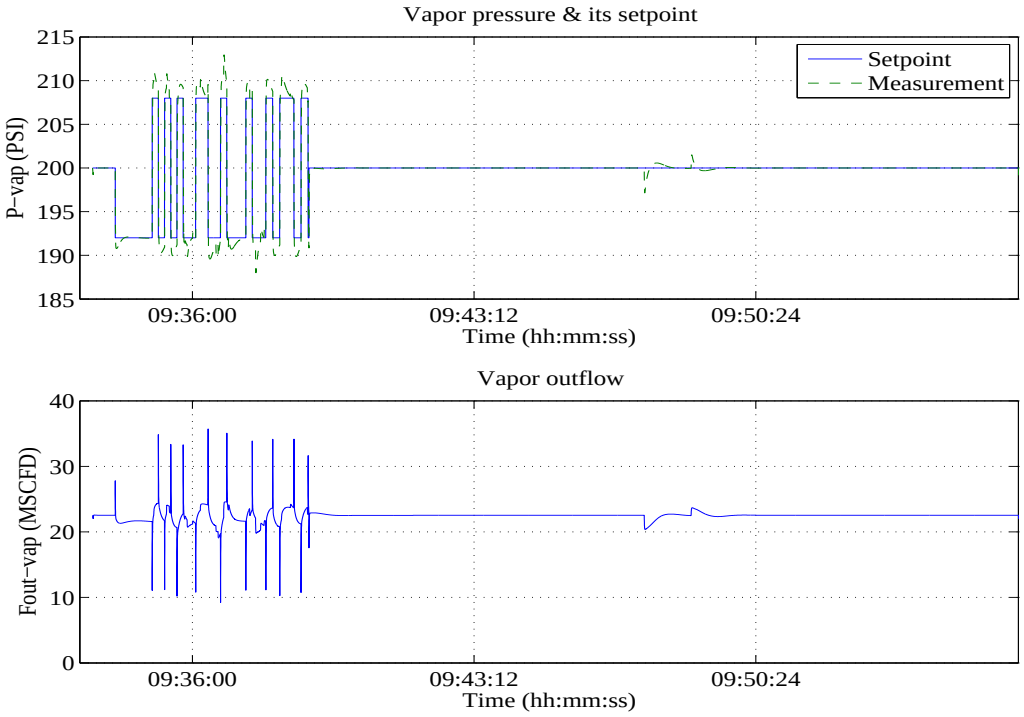

Figure 6.6: Scenario 1: Three-phase separator pressure logged by the pilot plant agent

6.1.2 Behaviors of the statistical preprocessing and model ID agents

Raw data are received by the statistical pre-processing agent, which removes any outliers and corrects missing data by replacing them with the previous data value, as demonstrated by the clean two-phase separator liquid volume and pressure data record in figures 6.7 and 6.8. The statistical agent first checks if the pilot plant is in steady state to prevent applying the PRBS signal in a transient state. Apparently the pilot plant takes a time period of $T_{SS} = 37.204\,s$ to reach steady state due to the plant small initial conditions, as shown in figures 6.7 and 6.8. Processed data are sent to the model ID agent during the PRBS signal application, after which a new process model can be estimated.

Figure 6.9 shows measured plant outputs along with their simulated counterparts using the newly identified plant model. Each process variable data record has a length of 300 seconds, which was the pre-specified PRBS signal application time. It is interesting to notice that although missing data have been corrected, they still affect the identified model quality, as indicated by the two-phase separator pressure data record (refer to the second plot in figure 6.9 with a model fit of 66%). Figure 6.10 shows the plant inputs during the PRBS signal application task.

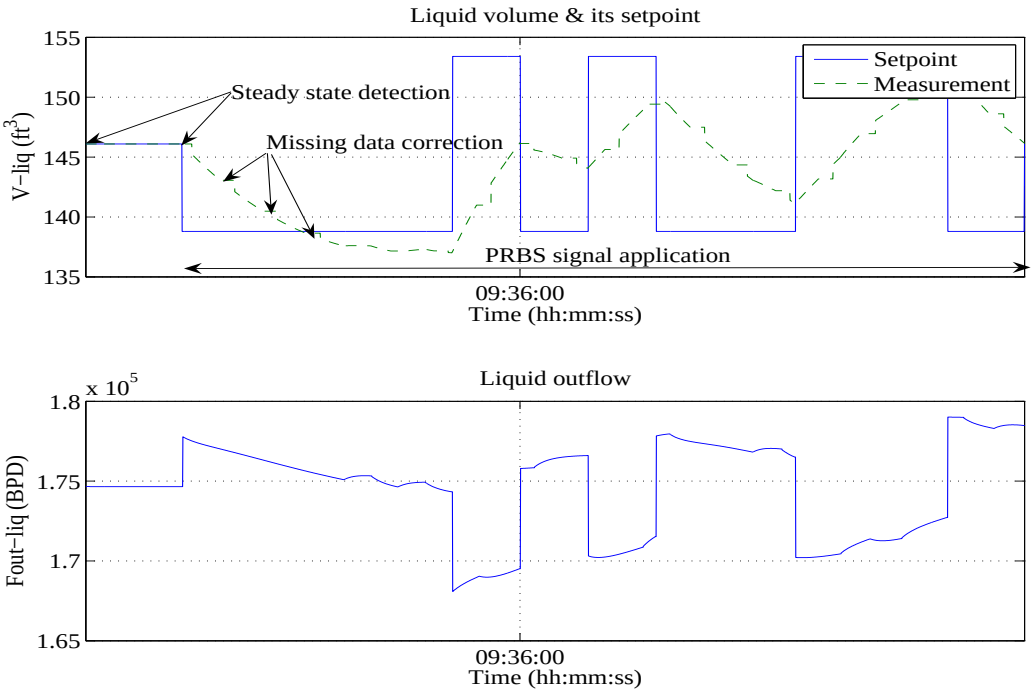

Figure 6.7: Scenario 1: Two-phase separator liquid volume logged by the statistical pre-processing agent

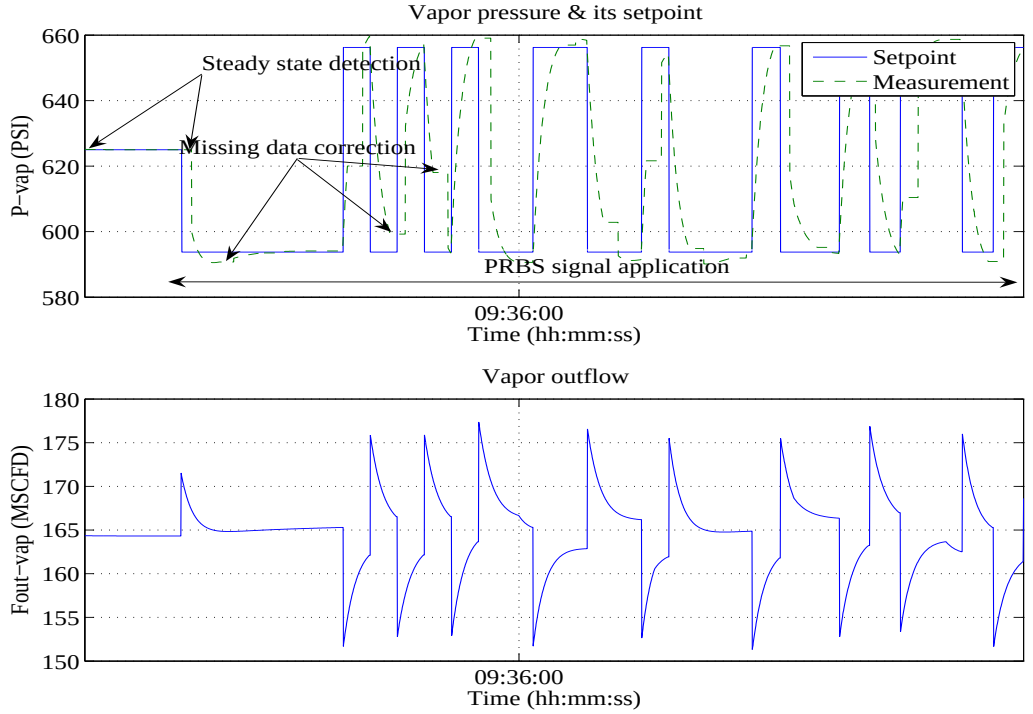

Figure 6.8: Scenario 1: Two-phase separator pressure logged by the statistical pre-processing agent

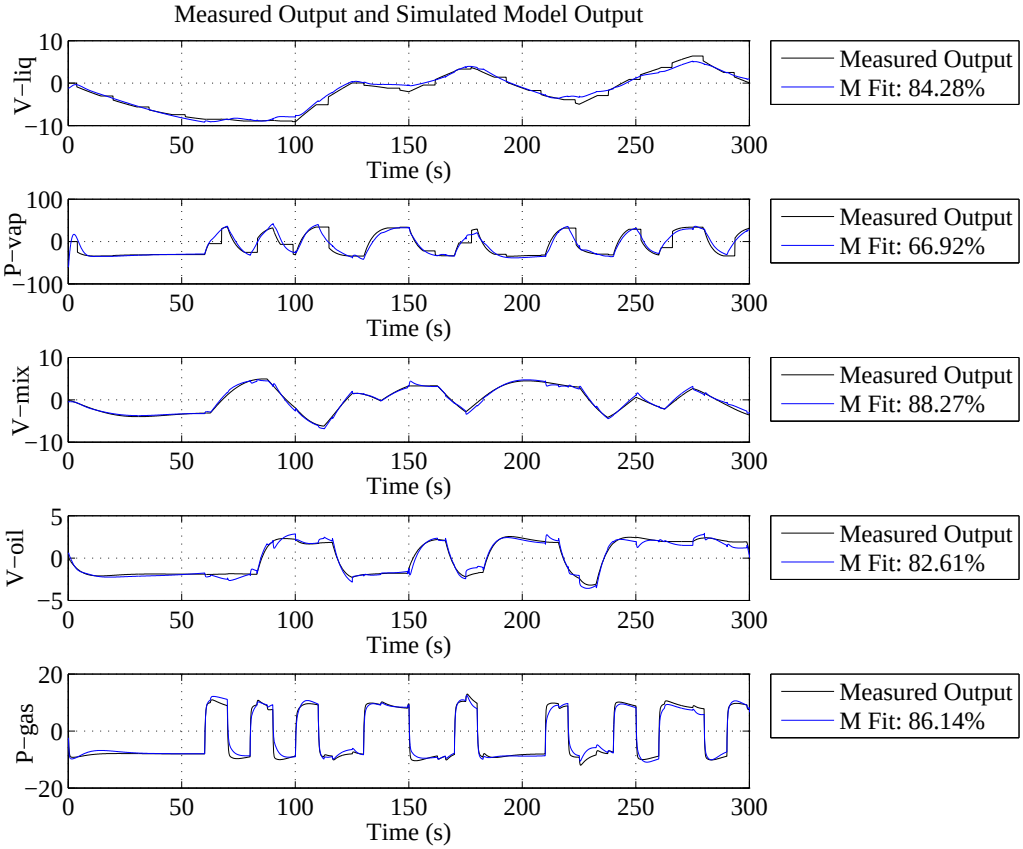

Figure 6.9: Scenario 1: Measured plant outputs and simulated model outputs logged by the model ID agent

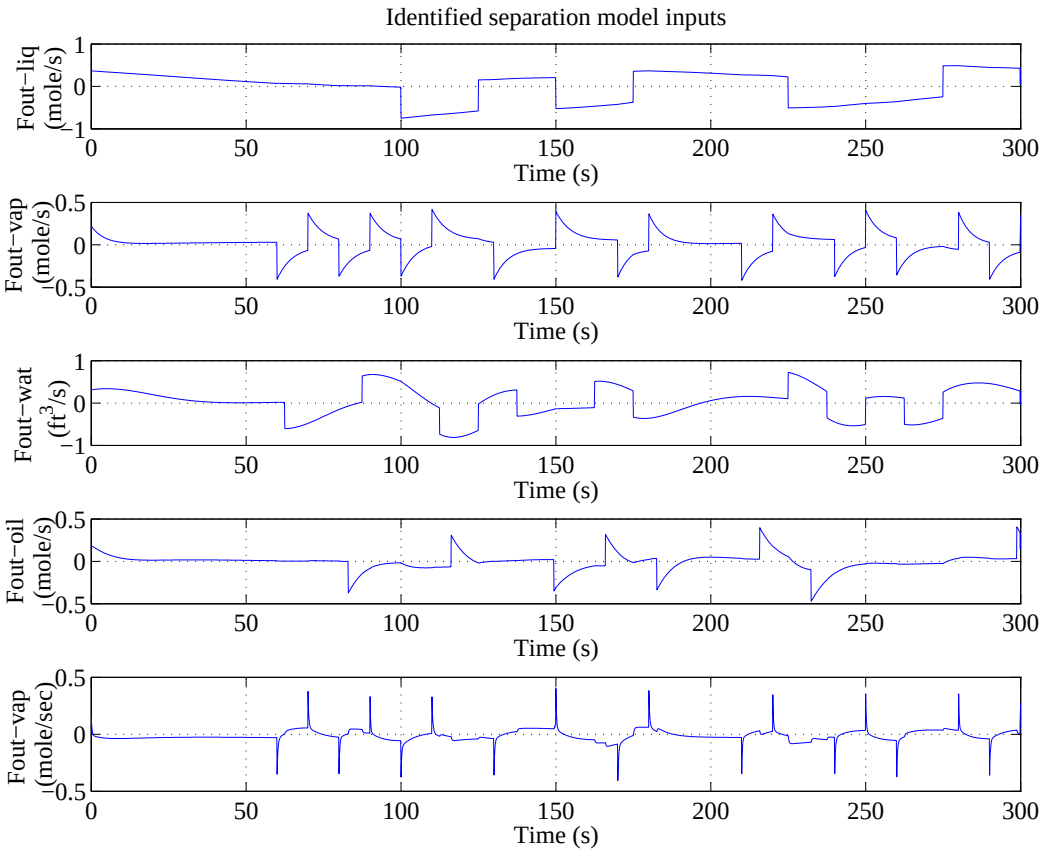

Figure 6.10: Scenario 1: Plant inputs logged at the model ID agent

6.1.3 The FDIA agent behavior

Once the new process model is received by the FDIA agent, then it can design its FDI filter and deploy it to diagnose faulty plant instrumentation. Figure 6.11 shows the three-phase water volume data record collected after the FDI filter is deployed. When the water volume sensor fault occurs, its effect can be noticed not only in the local control loop of the faulty instrumentation but also downstream, which is seen as disturbance in the three-phase oil volume control loop, as shown in figure 6.5. Figure 6.11 shows that the actual process variable has a different response from its corresponding measurement, i.e., the +15% error starts to drive the actual water volume to a lower setpoint in an effort to make the sensed setpoint approach the desired value; once the fault is accommodated the actual water volume returns to the correct setpoint.

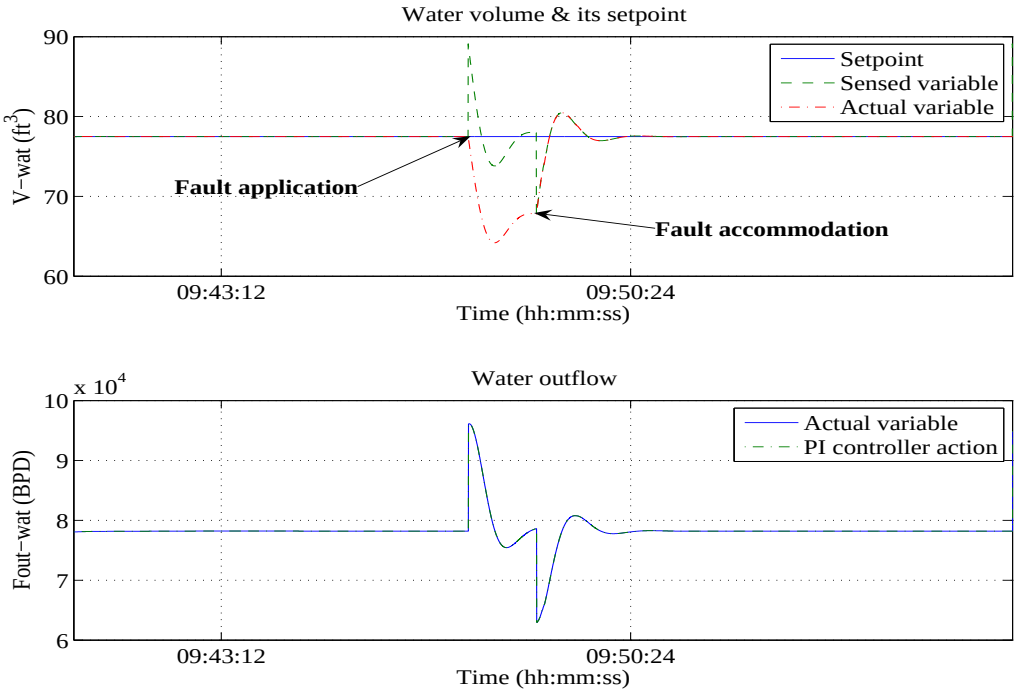

Figure 6.11: Scenario 1: Three-phase separator water volume logged by the FDIA agent

The FDIA agent generates a general parity vector whose abnormal magnitude can

detect faulty instrumentation, and generates the angles between the parity vector and the reference directions of the process variables. When there is a fault, then the smallest angle indicates the approximate alignment of the parity vector with the reference direction of a specific instrumentation fault. Hence the fault can be isolated based on the smallest angle after the fault detection. It is clear from the top plot in figure 6.12 (produced by FDI routines documented in [104, 70, 71, 69, 72, 94]) that the general parity vector (GPV) magnitude increased significantly, which indicates that a fault occurred. Furthermore, the smallest angle after the fault detection instant is the one that corresponds to the water volume sensor in the three-phase separator, as indicated by the blue dash-dotted trace in the middle plot of figure 6.12. The other GPV angles are higher than the faulty volume sensor GPV angles, as indicated by the other traces in the middle and bottom plots of figure 6.12.

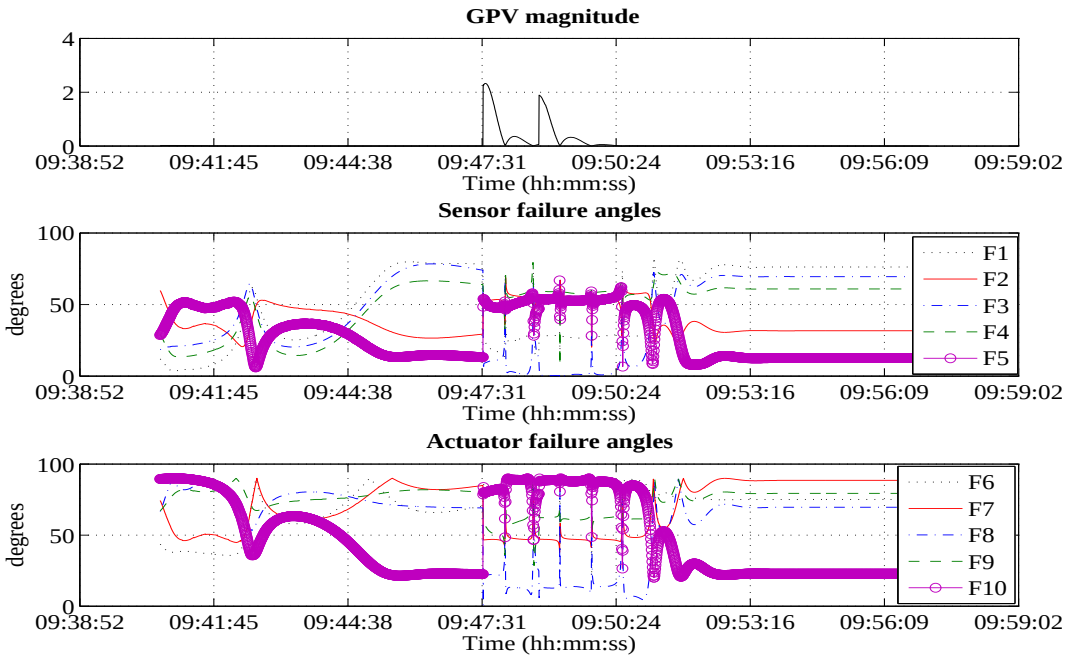

Figure 6.12: Scenario 1: FDIA agent diagnostic signals

The parity vector-based FDI angles are highly sensitive to process variable changes when there is no fault. This is because of the small size of the GPV vector in the

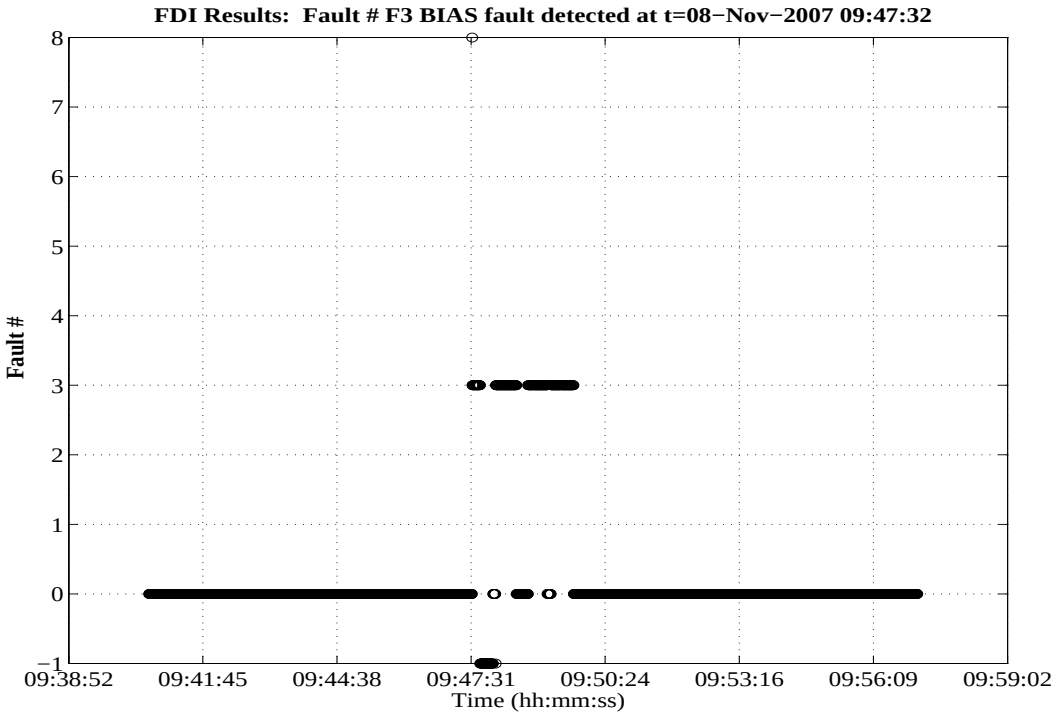

Figure 6.13: Scenario 1: FDIA agent fault display

no-fault situation, which can change its angle widely even in the case of very small process variable changes, as indicated by the large variation of the GPV angles before fault occurrence in figure 6.12. The local decision-making logic of the FDIA agent ignores the angles until a large GPV magnitude signals fault detection, then it isolates the fault after its occurrence as demonstrated in the FDI GUI [94], as shown in figure 6.13 . It interesting to notice a fault # of -1 occurred at the beginning of fault isolation task (-1 indicates an unknown fault). The FDIA agent isolates faults when the process variables have reached an acceptable steady state level, so isolation is ineffective during the transient part of the fault dynamics. As soon as the supervisory agent receives the fault information from the FDIA agent, including the fact that it is a sensor fault, it alerts the FDIA agent to start the fault accommodation task. The FDIA agent then estimates the fault size, which is used to accommodate the fault (correct the sensor reading).

Once the fault has been accommodated the actual water volume process variable

returns to its nominal setpoint, which matches its corresponding corrected measurement, as indicated by figures 6.11 and 6.12. The FDIA agent logic then indicates a no-fault situation during the fault accommodation task, as indicated by figure 6.13. Figure 6.14 shows the accommodation parameters in terms of the estimated fault size and the recursive fault size estimation error, which is only effective during faults of ramp type. The estimated fault size is +15%, which matches the original fault size value.

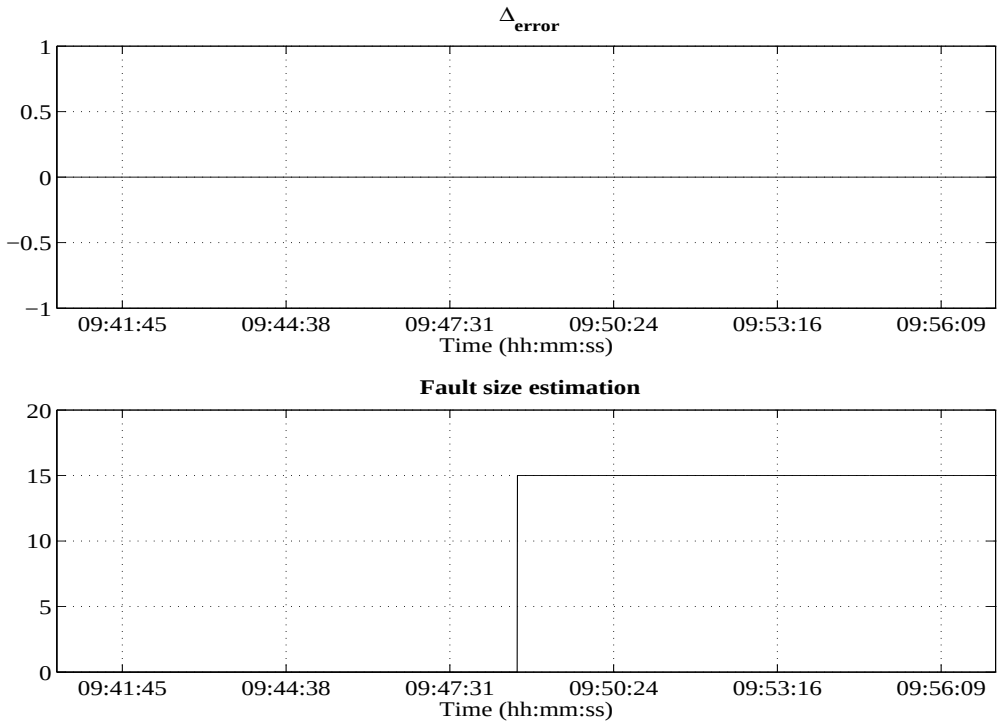

Figure 6.14: Scenario 1: FDIA agent fault accommodation parameters

6.1.4 The supervisory agent behavior

The supervisory agent monitors the state of the reactive agents and reasons about their current state according to its knowledge base. Each reactive agent is represented by an object with a set of attributes that represents its own state, as discussed in chapters 4 and 5 (refer to figure 5.4 in chapter 5). Table 6.2 demonstrates the pilot plant supervisory frame during the fault accommodation task. The pilot

plant agent frame shares some common attributes with the other reactive agents, which represent the agent's internal state, MPI, and G2 communication channels' states. For example, the pilot plant frame is in the simulation state and executing its functionality as indicated by the simulation status attribute. Its MPI and G2 links are connected, and the pilot plant agent has a rank (i.e., the software process number) of 0 in the MPI environment defined by the MPI communicator attribute. The agent's decision attribute indicates that a fault simulation scenario is applied. The decision attribute is the decision made by the supervisory agent depending on the current state of the agent. The MPI channel decision attribute indicates that the accommodated data MPI channel is opened, through which the pilot plant agent receives the accommodation parameters from the FDIA agent.

Names	ENV-OBJECT
Rank	0
State	simulate
Mpi comm	comm-world
Mpi comm size	4
G2 link status	connected
Mpi link status	connected
Decision	fault
Simulation status	on
Mpi channel decision	open-accom-channel
Prbs status	none

Table 6.2: Scenario 1: Pilot plant agent supervisory frame

Tables 6.3 and 6.4 show the statistical preprocessing and model ID supervisory frames, which have the same ICAM system common attributes (i.e., rank, G2 link, MPI link status, decision etc.). Both agents have a common model status attribute, which indicate that the two agents have updated their knowledge about the current dynamics of the pilot plant. The statistical preprocessing agent has a steady state detection attribute to indicate if the pilot plant is in a steady or transient state. The

decision attributes in these agents' supervisory frames have the value no-decision, which indicates that the supervisory agent does not require these agents to do any task. Likewise, the MPI channel decision attributes of the statistical preprocessing and model ID supervisory frames have a no-decision value, which indicates that the supervisory agent does not require these agents to close any of their MPI data channels.

Names	STAT-OBJECT
Rank	1
State	simulate
Mpi comm	comm-world
Mpi comm size	4
G2 link status	connected
Mpi link status	connected
Decision	no-decision
Simulation status	on
Mpi channel decision	no-decision
Model status	model-is-identified
Steady state	none

Table 6.3: Scenario 1: Statistical preprocessing agent supervisory frame

The FDIA agent supervisory frame has the same common attributes, which indicate that the agent is in the simulation state and is executing the fault diagnosis task, as shown in table 6.5. It also has attributes about the fault information such as the fault size, sign, type, time, and location. The model status and the FDI design status attributes indicate that the FDIA agent has received the process model and has deployed the designed FDI filter. The FDIA agent supervisory frame also has attributes to represent the accommodation task status and the recursive fault estimation in case of ramp faults. For example, the FDIA agent has reported the fault information back to the supervisor for further processing and actions. In this case the FDIA agent successfully detected, isolated, and identified the faulty instrumenta-

Names	MODELID-OBJECT
Rank	2
State	simulate
Mpi comm	comm-world
Mpi comm size	4
G2 link status	connected
Mpi link status	connected
Decision	no-decision
Simulation status	on
Mpi channel decision	no-decision
Model status	model-is-identified

Table 6.4: Scenario 1: Model ID agent supervisory frame

tion, which is the three-phase separator water volume sensor (F3; refer to table 6.1). The fault has occurred at time T_{fault} = 9:47:32, which is nearly the exact fault application time. The fault has a type bias with an estimated size of +15%. The fault accommodation task is in progress and the accommodation parameters are sent to the pilot plant agent, as indicated by the MPI channel decision attribute. Since the fault type is bias and not of a ramp type, then recursive fault size estimation is not required as indicated by the corresponding attribute of the table.

Names	FDIA-OBJECT
Rank	3
State	simulate
Mpi comm	comm-world
Mpi comm size	4
G2 link status	connected
Mpi link status	connected
Decision	diagnose-faults
Simulation status	on
Mpi channel decision	open-accom-channel
Model status	model-is-identified
Fdi design status	fdi-filter-is-designed
Fault	f3
Fault sign	plus
Fault size	15.0
Fault type	bias
Fault time	"08-Nov-2007 09:47:32"
Accommodation status	acc-in-progress
Recursive fault size estimation status	none
Acknowledge fault	off

Table 6.5: Scenario 1: FDIA agent supervisory frame

6.1.5 Network activity

The ICAM system prototype is deployed in a Windows 2003 network, which has two nodes (i.e., workstations). The first node has the statistical preprocessing agent and the FDIA agent running. The second node has three running agents, namely, the pilot plant agent, the model ID agent, and the supervisory agent, as shown in figure 6.15. The total communication throughput between the two nodes (indicated by green solid arrows in figure 6.15) is composed of five channels; one asynchronous supervisory channel (indicated by black dashed arrows in figure 6.15), and four synchronous MPI data channels. The first MPI data channel is the raw data channel

which connects the pilot plant agent with the statistical preprocessing agent (indicated by a green solid arrow). The statistical preprocessing agent transfers the processed data on the second MPI data channel (indicated by dark-blue solid arrows) to the model ID and FDIA agents. Once the plant model is identified, it is transferred to the statistical preprocessing and FDIA agents through the model MPI channel (indicated by magenta dash-double-dotted arrows). Finally the accommodation parameters are transferred from the FDIA agent to the pilot plant agent through the accommodated data MPI channel (indicated by a purple dash-dotted arrow).

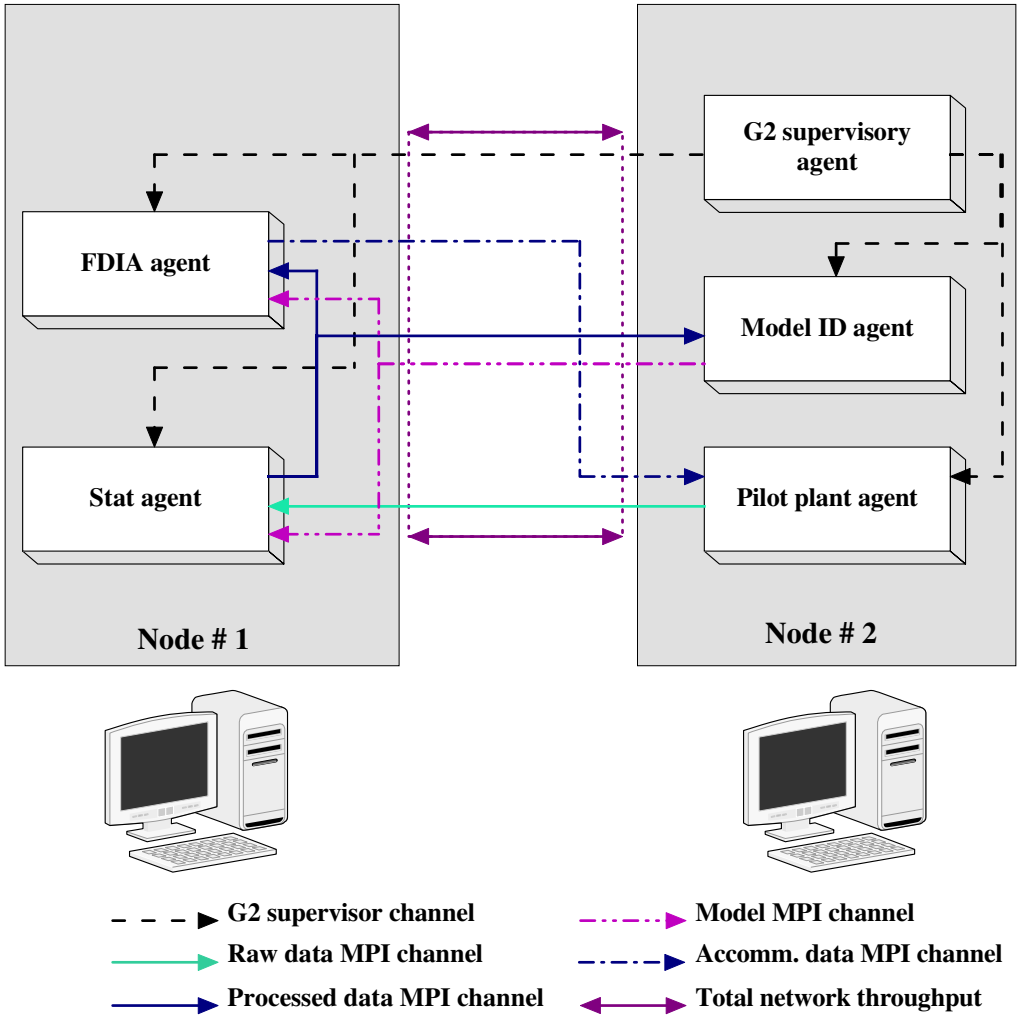

Figure 6.15: ICAM system prototype network architecture

Figure 6.16 depicts the ICAM system prototype network activity during the first simulation scenario. After the ICAM system prototype starts up (event 1), the raw data and processed data MPI channels start to transfer data at a rate of 330 Kbps (i.e., 0.33% of the 100 Mbps network transfer rate) for each channel, as indicated by the event 1; note that the green and dark-blue traces have nearly the same rate. The total network throughput is represented by the purple trace. The transfer rates of the MPI channels dip prior to event 2 because of increasing memory consumption and computations resulted from increasing data storage in some agents (refer to section 6.4.1 for more detailed analysis). The processed data channel (i.e., the dark-blue trace) is closed during the plant model identification task, as indicated by event 2. Once the plant model is transferred to the corresponding agents, the processed data channel is opened again and the fault diagnosis task is started, as indicated by event 3. When the three-phase water volume sensor fault is detected and the fault accommodation task is started, the accommodated data channel is opened, as indicated by event 4; the dark-blue trace is stepped up to its twice rate (i.e., 580 kbps), and the total network transfer rate is at 870 kbps. When the system shuts down at the end of the first scenario, the MPI channels are closed sequentially starting with accommodated data channel, followed by the processed data channel, and finally the raw data channel, as indicated by event 5.

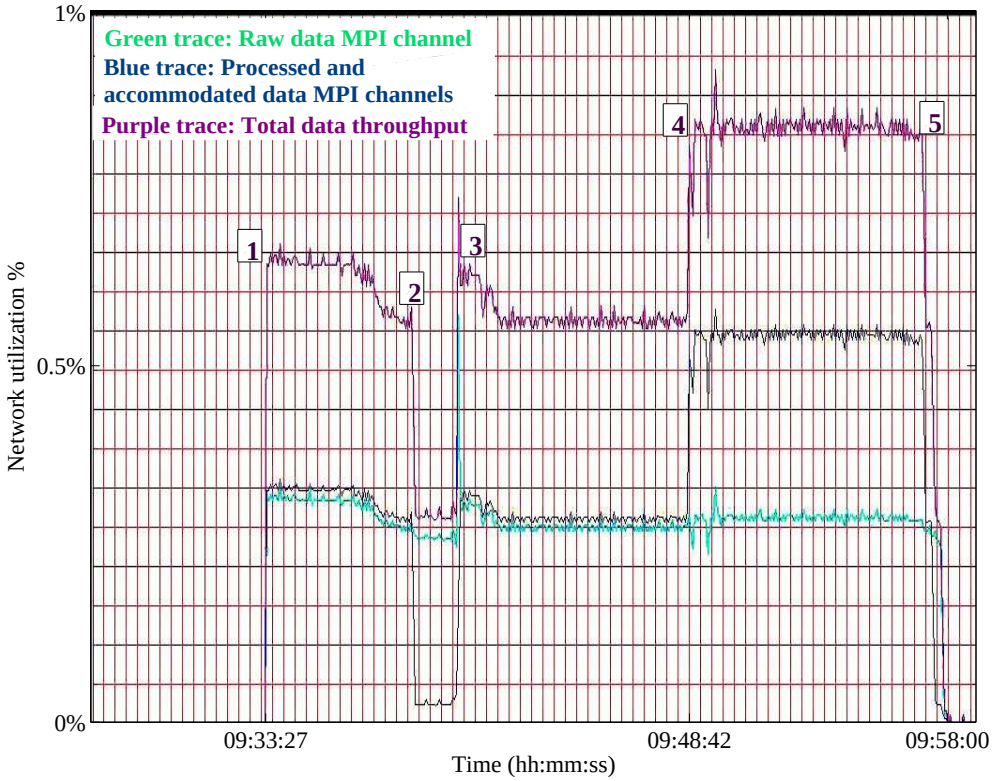

Figure 6.16: Scenario 1: ICAM system prototype network activity

6.2 Scenario 2: Faulty Gas Outflow Valve in the Two-Phase Separator Sub-Process

The second simulation scenario is done by applying a fault in the two-phase separator gas outflow valve, which is stuck at 15% higher than its nominal operating point (F7; refer to control loop PCL1 in figure 6.1). The fault is applied at time T_{fault} = 14:33:40, which is after the ICAM system has started the fault diagnosis task. Figure 6.17 shows the two-phase separator pressure data and its associated gas outflow data logged at the FDIA agent. It is clear that the measured and actual pressure of the two phase separator goes to zero as a result of the flow (i.e., top plot of figure 6.17). The bottom plot of figure 6.17 shows the faulty flow measurement (i.e., solid blue trace) and the control action generated by the PI controller of loop PCL1, which rapidly decreases (indicating the valve should be closing) until

it reaches the lower bound of zero (i.e., green dash-dotted trace). The FDIA agent detects the fault when the GPV magnitude is increased significantly, as indicated by the top plot of figure 6.18. Furthermore, the lowest GPV angle compared to the other angles indicates the faulty instrumentation (i.e., F7), as shown in bottom plot of figure 6.18. The FDIA agent local logic isolates in this case fault # 7, as shown in figure 6.19. This fault corresponds to the two-phase separator gas outflow valve being stuck (refer to table 6.1). Note that actuator faults cannot be accommodated (a stuck valve cannot be compensated); hence, no accommodation parameters are generated.

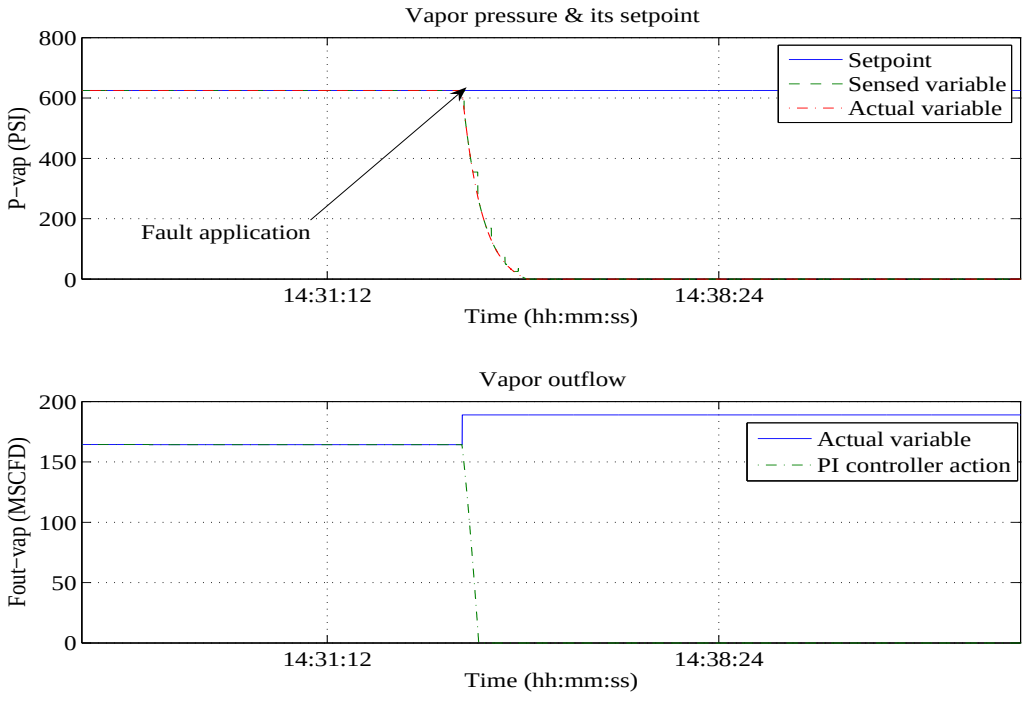

Figure 6.17: Scenario 2: Two-phase separator pressure logged by the FDIA agent

Once the FDIA agent isolates the fault, it sends the fault information to the supervisory agent for further processing, as shown in table 6.6. The fault attribute of the FDIA supervisory frame shows that fault F7 has occurred at time at time T_{fault} = 14:33:40. The accommodation status attribute indicates that accommodation is not possible in case of actuator faults. The fault sign and size attributes do not provide

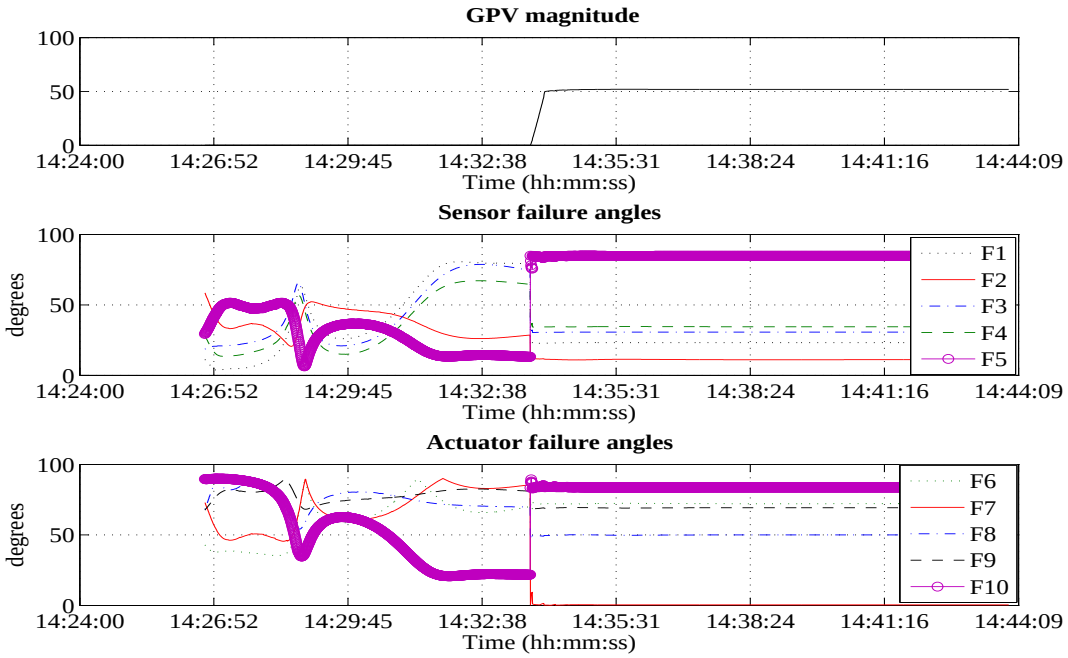

Figure 6.18: Scenario 2: FDIA agent diagnostic signals

any fault knowledge, as the FDIA agent is not designed to estimate actuator fault size. Figure 6.20 illustrates the ICAM system network activity during this scenario, which is nearly similar to the first simulation scenario. The only difference is that the accommodated data channel is not active in this case.

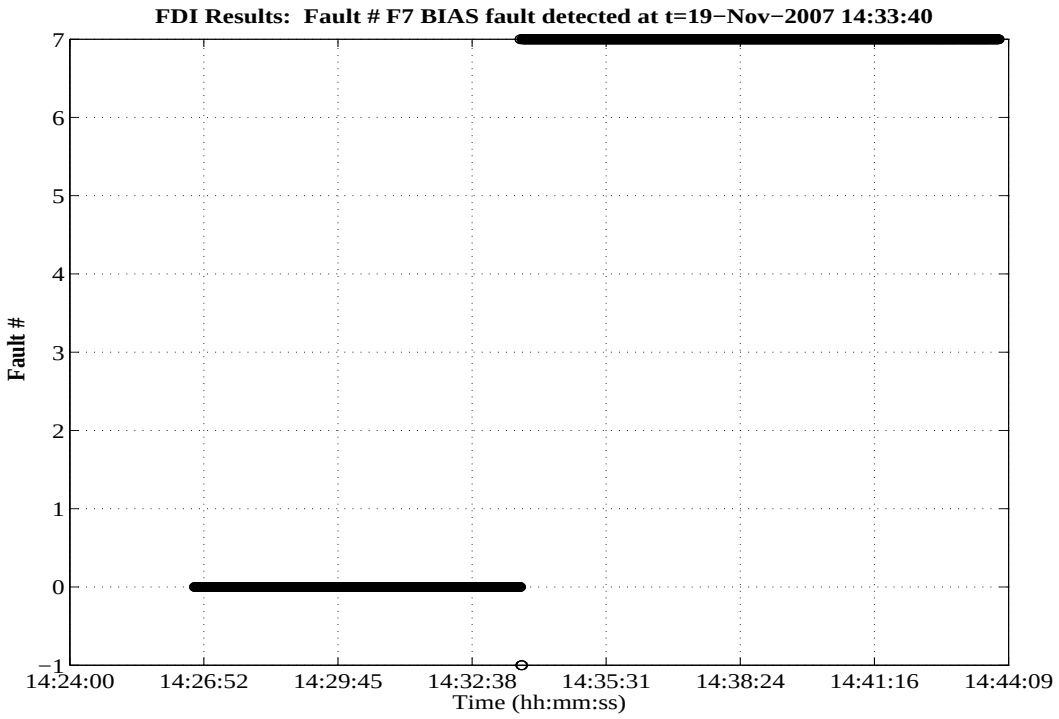

Figure 6.19: Scenario 2: FDIA agent fault display

Names	FDIA-OBJECT
Rank	3
State	simulate
Mpi comm	comm-world
Mpi comm size	4
G2 link status	connected
Mpi link status	connected
Decision	diagnose-faults
Simulation status	on
Mpi channel decision	no-decision
Model status	model-is-identified
Fdi design status	fdi-filter-is-designed
Fault	f7
Fault sign	none
Fault size	NaN
Fault type	bias
Fault time	"19-Nov-2007 14:33:40"
Accommodation status	acc-not-possible
Recursive fault size estimation status	recalculation-not-required
Acknowledge fault	off

Table 6.6: Scenario 2: FDIA agent supervisory frame

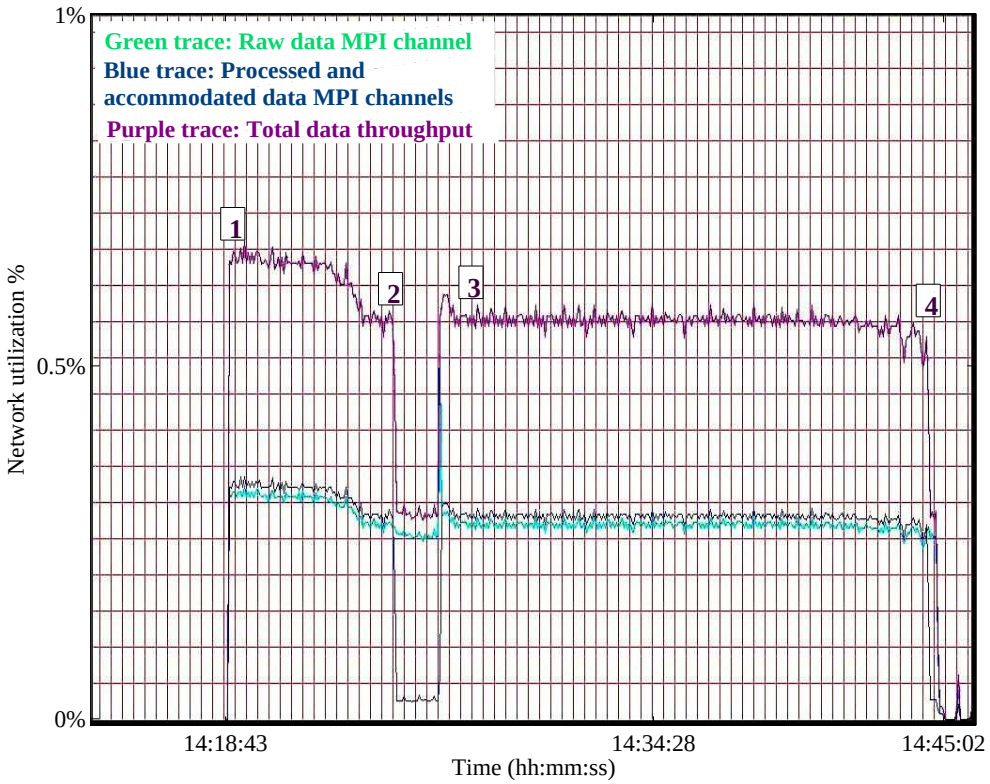

Figure 6.20: Scenario 2: ICAM system prototype network activity

6.3 Scenario 3: Drift Fault in the Two-Phase Separator Liquid Level Sensor

The third simulation scenario investigates the system behavior where a ramp fault is applied to the two-phase separator liquid level sensor (F1; refer to control loop LCL1 in figure 6.1). The drift fault was applied at a time of T_{fault} = 10:44:15 with a slope of $S = 10\ \%/100s$. Figure 6.21 depicts the fault application effect on the liquid level in the two-phase separator sub-process. Clearly, the drifting sensor reading causes a fault in both the measurement and the actual liquid volume. The PI controller of the LCL1 loop generates the wrong control action, trying to correct the erroneous drifting volume measurement. This results in a big decrease in the actual liquid volume of the two-phase separator. The FDIA agent detects the fault when the GPV magnitude increases significantly, as shown in the top plot of figure 6.22. Furthermore, the lowest GPV angle corresponds to fault F1 compared to other angles, as indicated by the black dotted trace in the middle plot of figure 6.22. The internal logic of the FDIA agent isolates fault F1 (refer to table 6.1), which corresponds to a faulty two-phase separator liquid volume sensor, as shown in figure 6.23.

The FDIA agent sends the fault information to the supervisory agent, as depicted in table 6.7. The supervisory agent alerts the FDIA agent to start the fault accommodation task, which is started at time T_{accomm} = 10:46:07. The FDIA agent flags back that the accommodation task is in progress along with the recursive fault estimation task (refer to the accommodation attributes in the FDIA agent supervisory frame in table 6.7) because of the ramp nature of the fault.

It is interesting to notice that the fault slope size is initially estimated with a little error, because the accommodation task was activated during the transient time of the fault as opposed to the steady state. This results in an incomplete fault accommodation, as shown in figure 6.21 (top plot), which produces a growing error. Figure 6.24 show the accommodation parameters in terms of the estimated fault

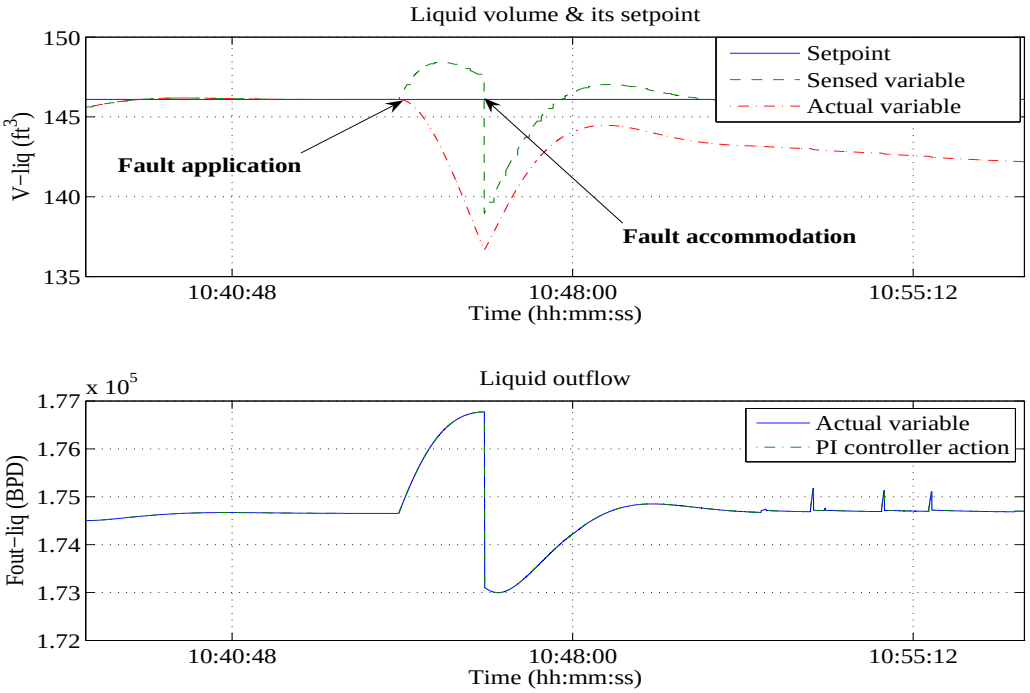

Figure 6.21: Scenario 3: Two-phase separator liquid volume logged by the FDIA agent

and the recursive fault estimation error. It is evident that there is an error in the fault size, as indicated in the bottom plot of figure 6.24. As far as the network activity during this simulation scenario, it is similar to the network activity of the first simulation scenario, where the accommodated data channel is opened.

Names	FDIA-OBJECT
Rank	3
State	simulate
Mpi comm	comm-world
Mpi comm size	4
G2 link status	connected
Mpi link status	connected
Decision	diagnose-faults
Simulation status	on
Mpi channel decision	open-accom-channel
Model status	model-is-identified
Fdi design status	fdi-filter-is-designed
Fault	f1
Fault sign	plus
Fault size	9.852
Fault type	ramp
Fault time	"07-Nov-2007 10:44:41"
Accommodation status	acc-in-progress
Recursive fault size estimation status	recalculation-in-progress
Acknowledge fault	off

Table 6.7: Scenario 3: FDIA agent supervisory frame

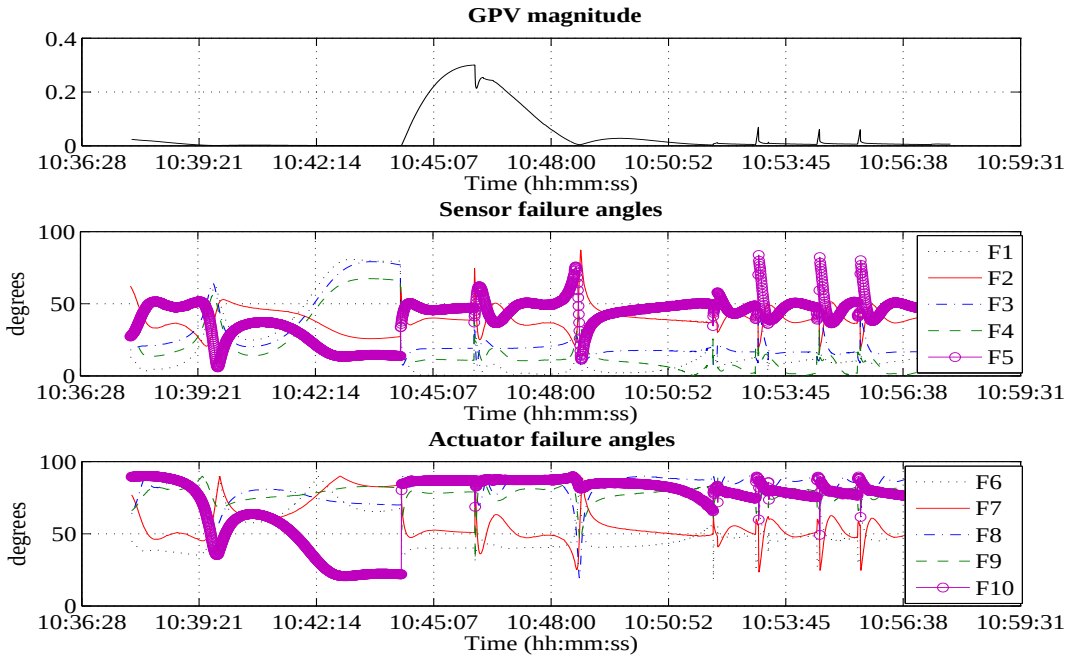

Figure 6.22: Scenario 3: FDIA agent diagnostic signals

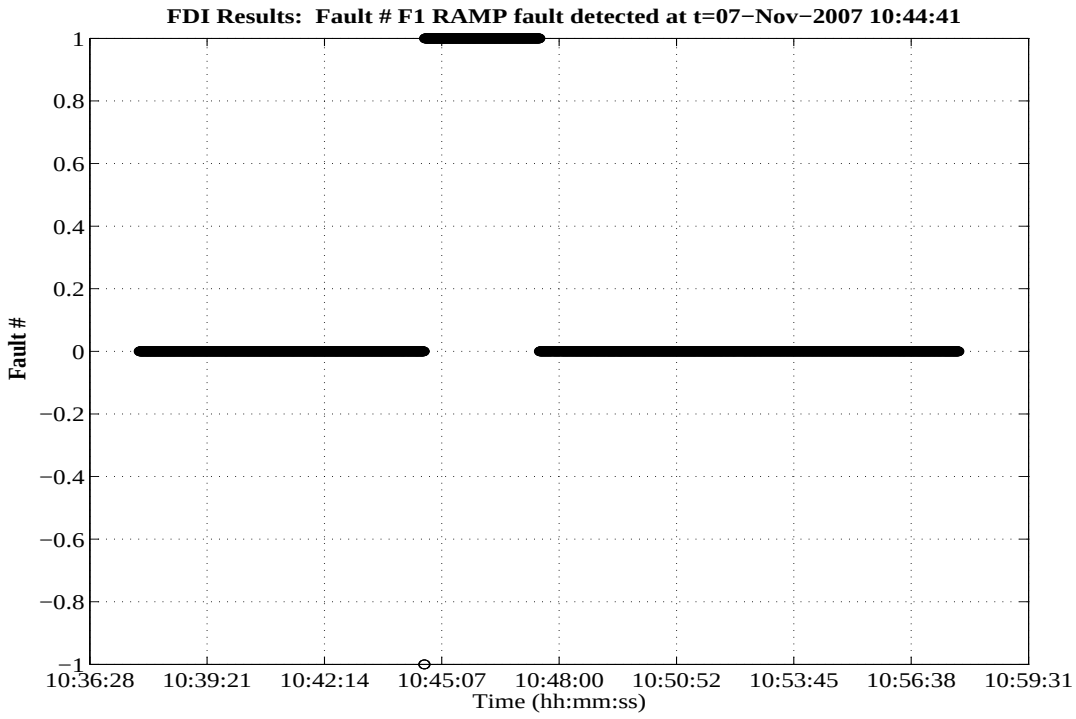

Figure 6.23: Scenario 3: FDIA agent fault display

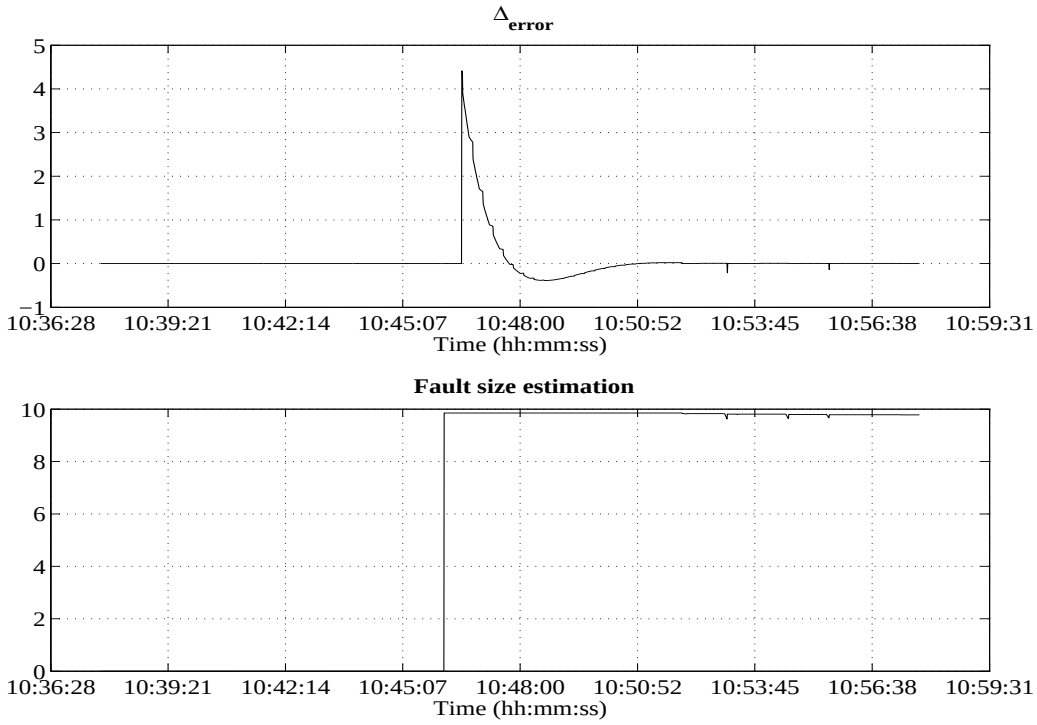

Figure 6.24: Scenario 3: FDIA agent fault accommodation parameters

6.4 Performance Analysis

Having verified and validated the ICAM system prototype functionality during the previous simulation scenarios, it is crucial to analyze its performance to pinpoint any computational bottlenecks and to verify the correctness of the computation/communication overlap (i.e., the correct order of communication and computation tasks in each agent's code). The performance analysis is done during the fault accommodation system mode of the first simulation scenario. Table 6.8 illustrates the profile of the pilot plant agent, whose execution cycle took 108.543 milliseconds.

It is evident that the real-time clock functionality took the biggest execution time slot (i.e., about 70.49%). The evaluation of the oil separator ordinary differential equation (ODE) model took 24.12% of the total execution time, due to the nonlinear problem being solved every sampling period [80]. Raw data storage consumed around 1.77% of the agent's execution cycle. Communicating data to other agents and messages to the supervisory agent did not have a significant effect on the agent's performance, which indicates a good communication/computation overlap.

While computational functionalities dominated the pilot plant agent, data communications with other agents took the largest execution time slot in the statistical preprocessing agent. That is, the raw data reception task took 69.64% of the agent's execution cycle, as demonstrated in table 6.9. This is due to synchronization with other agents during data reception, i.e., waiting. However, this is less significant on the agent performance when sending processed data to other agents (i.e., about 11.99% of the execution time), as specified by the system design requirements [78, 79]. The total execution cycle of this agent was 106.98 milliseconds. It is evident that there is a performance bottleneck in this agent due to raw and processed data storage (around 16.8%). This can be rectified by adding a database agent to the system which stores the different data types across the ICAM system. Again, the communication part with the supervisor had a minimum effect on performance.

Functionality name	Total Time Per Execution Cycle (ms)	% Time
Real time clock	76.512	70.49%
Separator ODE model evaluation	26.18	24.12%
Accommodation data reception	3.387	3.12%
Raw data storage	1.925	1.77%
Raw data sending	0.336	0.31%
Communication with supervisor	0.203	0.18%
Totals	108.543	100%

Table 6.8: The pilot plant agent performance profile

Functionality name	Total Time Per Execution Cycle (ms)	% Time
Raw data reception from pilot plant	74.50	69.64%
Processed data sending	12.837	11.99%
Raw data storage	9.451	8.83%
Processed data storage	8.54	7.98%
Outlier removal	1.404	1.31%
Communication with supervisor	0.248	0.23%
Totals	106.98	100%

Table 6.9: The statistical agent performance profile

When it comes to the model ID agent, the reception of processed data from the statistical preprocessing agent had the biggest effect on performance (i.e., about 99.48% of the agent's execution cycle time). While the execution cycle of this agent took 105.69 milliseconds, communications with the supervisory agent had the least effect on performance, as shown in table 6.10. The FDIA agent had a similar profile of the model ID agent, in which data communications took 90.89% of the agent's cycle execution time. We do notice here that data storage has a fairly undesirable effect of 13.24% on the FDIA agent performance, as illustrated in table 6.11. Table 6.12 demonstrates the performance of the supervisory agent during the real-time system

simulation. The G2 supervisory agent was in an idle state for almost 99.18% of the total simulation time, whereas communications with other agents had almost no impact on the agent's performance, as specified in the design requirements.

Functionality name	Total Time Per Execution Cycle (ms)	% Time
Processed data reception	105.15	99.48%
Communication with supervisor	0.54	0.52%
Totals	105.69	100%

Table 6.10: The model ID agent performance profile

Functionality name	Total Time Per Execution Cycle (ms)	% Time
Accommodation data sending	68.601	45.67%
Processed data reception	67.916	45.22%
Processed data storage	7.45	4.96%
FDI variables storage	5.763	3.84%
Communication with supervisor	0.464	0.31%
Totals	150.194	100%

Table 6.11: The FDI agent performance profile

Functionality name	Total Time	% Time
Idle time	1575 s	99.18%
Scheduling time	0.982 s	0.06%
Communication with agents	4.505 s	0.28%
Other functionalities	7.565 s	0.47%
Totals	1588.052 s	100%

Table 6.12: The supervisory agent performance profile

6.4.1 Complete ICAM system performance analysis

Although the ICAM system prototype performance analysis showed good results, the performance analysis was a snapshot done during the fault accommodation mode of the system. It is crucial to conduct a complete system performance analysis throughout the complete life-time of the ICAM system. The ICAM system goes through six different modes during its life-cycle, namely, system startup and steady state detection mode, PRBS signal application mode, model identification mode, fault diagnosis mode, fault accommodation mode, and system shutdown mode. A real-time simulation scenario was set up to measure the execution cycles of the different reactive agents and compare it against the ICAM system network activity during the six modes of the ICAM system. We studied scenario 3, i.e., we applied a bias fault in the liquid volume sensor of the two-phase separator to make the system execute the fault diagnosis and accommodation modes.

Figure 6.25 shows the measured execution cycles of the ICAM system reactive agents, where the overlapped execution cycles traces show remarkable synchronization among the reactive agents during the six system modes, since the agents have nearly the same execution cycle traces. The ICAM system agents start up with an execution cycle of 94 milliseconds, which increases to 95 milliseconds during the steady state detection mode (i.e., mode 1). It is interesting to notice that the execution cycle increases to a level of 110.7 milliseconds during mode 1. The execution

cycle of the model ID agent increases to around 92 milliseconds because of the time-consuming plant model estimation task. The processed data MPI channel is closed during this mode and the FDIA agent enters a waiting state till the end of mode 2, as shown in figure 6.25.

The pilot plant and statistical preprocessing agents continue executing their functionalities at a cycle level of 110.7 milliseconds. The agents' execution cycle decreases to a level of 98 milliseconds and then increases to a level of 109 milliseconds during the fault diagnosis mode (i.e, mode 4) and the beginning of the fault accommodation mode (i.e, mode 5). The execution cycle then increases to a level of 124 milliseconds during the fault accommodation and system shutdown modes. The gradual increases of the execution cycles of the agents are accompanied with matching gradual increases in agents' memory consumption and matching gradual decreases in network activity (i.e., less communications among agents). This interesting phenomenon is attributed to the fact that some agents store their local data in large matrices, whose growing size requires more computational effort and more memory consumption. This reflects on the communications and network activity of the ICAM system as demonstrated by figure 6.27.

Figure 6.26 shows the individual execution cycles of the ICAM system reactive agents, which again demonstrates the agents' remarkable synchronization in spite of the semi-autonomous nature of the ICAM system agents. We measured the network activity of the ICAM system prototype in this simulation scenario, as illustrated in figure 6.27. It is interesting to notice that the communication activity of the agents is a mirror of the computation activity. The gradual decreases in network activity match the gradual increases in computational activity of the agents, as observed in figures 6.25 and 6.27. It is also observed that the processed data MPI channel is closed during the model ID mode (i.e., mode 2) because of the time-consuming task of plant model estimation (as indicated by the yellow trace in figure 6.27).

This simulation scenario demonstrated an excellent ICAM system performance in terms of good computation/communications activities overlap, as specified by design requirements. It also highlighted the need to embed a data-base management agent in the ICAM system to relax the execution cycle of the reactive agents, and hence, to improve the ICAM system performance.

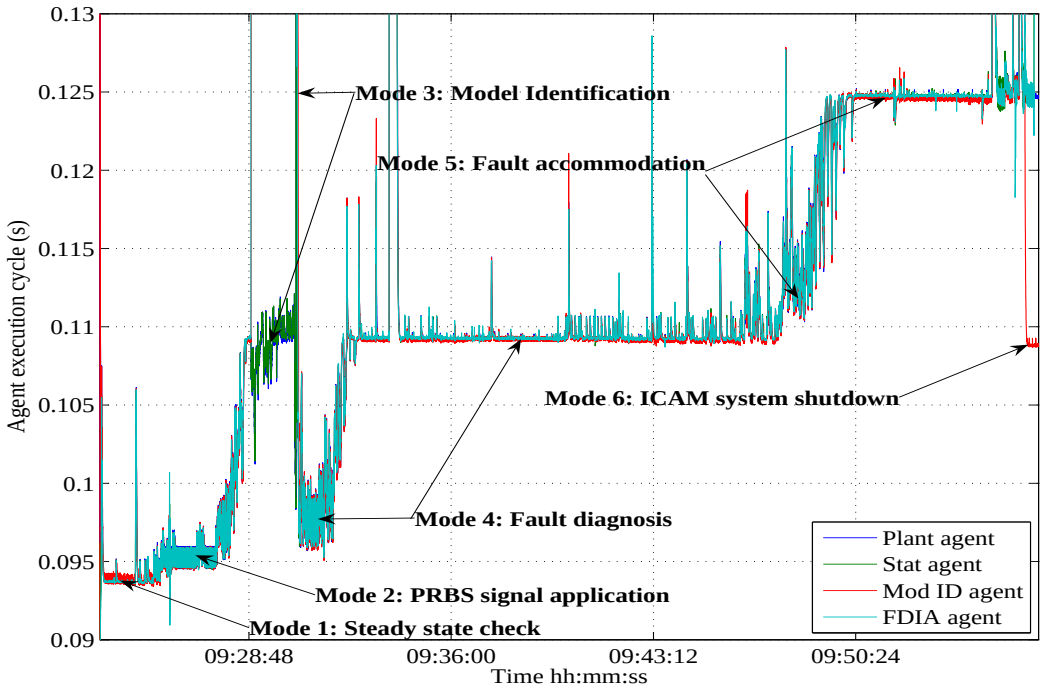

Figure 6.25: Execution cycles of ICAM system agents (overlapped)

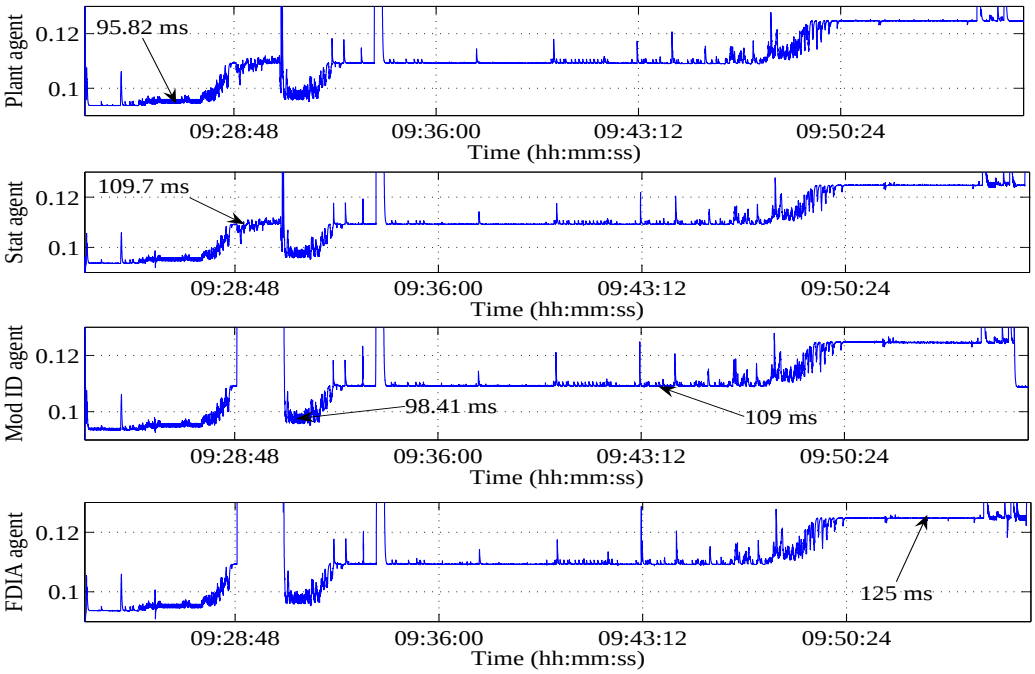

Figure 6.26: Execution cycles of ICAM system agents (non overlapped)

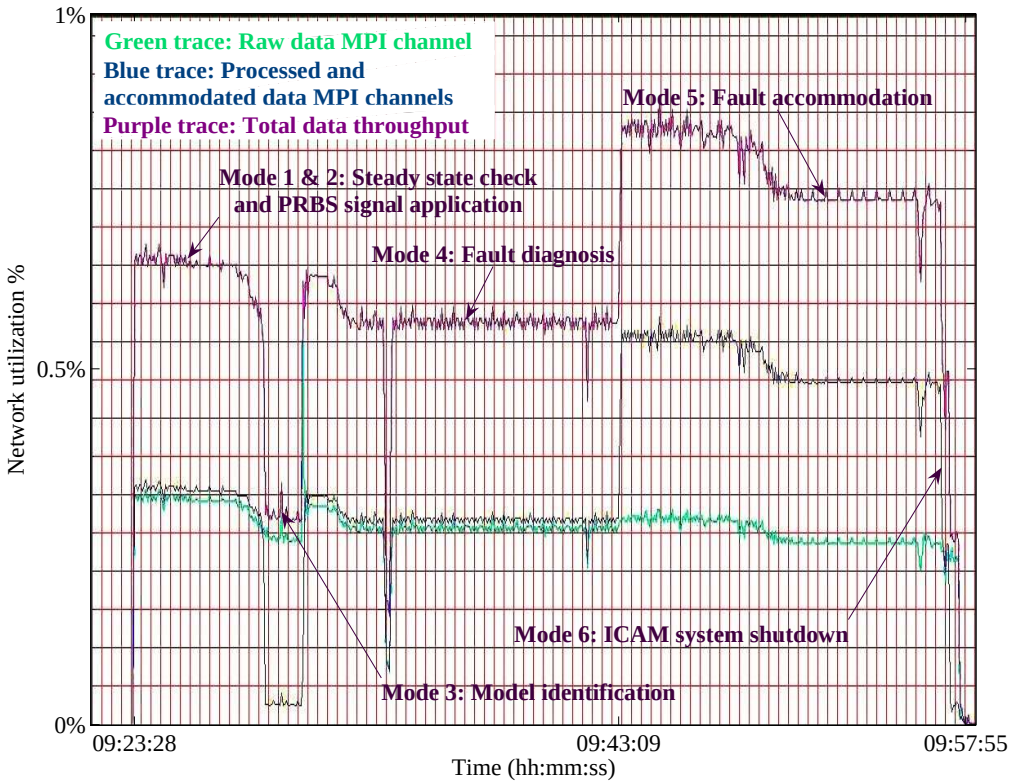

Figure 6.27: ICAM system prototype network activity

6.5 ICAM System Prototype Limitations

The ICAM system prototype showed excellent logical behavior in response to simple faulty sensors/actuator scenarios. However, the system has limitations that must be identified and analyzed carefully. This will reveal if the system will respond consistently against unexpected disturbances, and show a coherent performance and acceptable logical behavior. Two simulation scenarios have been applied to study the ICAM system prototype limitations, as discussed in the following sections.

6.5.1 Scenario A: ICAM system behavior during faults with fast dynamics

To demonstrate the system behavior during fault with fast dynamics, a +15% bias fault is applied to the three-phase separator pressure sensor (F5; refer to control loop PCL2 in figure 6.1) at time T_{fault} = 09:09:55. Figure 6.28 shows the measured and actual pressure of the three-phase separator along with its associated gas outflow. The figure obviously shows that there is a significant mismatch between the measured and actual pressure. The FDIA agent successfully detects a fault at time T_{fault} = 09:09:55 (refer to table 6.13), as the GPV magnitude spikes up sharply, as shown in the top plot of figure 6.29. The lowest GPV angle compared to other angles corresponds to fault F10, where it decreases sharply for a short time period, as shown in the bottom plot of figure 6.29. Accordingly the FDIA agent internal logic declares that a fault F10 has been isolated, as shown in figure 6.30. Interestingly enough, the isolation decision lasts for a very short period of time. Fault F10 corresponds to a fault in the three-phase separator gas outflow valve (refer to table 6.1), which is in the same control loop as F5. The pressure sensor fault is almost immediately masked by a quick spike in outflow, allowing the pressure to adjust to the erroneous setpoint so fast that correct isolation is impossible.

The FDIA agent sends the fault information to the supervisory agent to reason

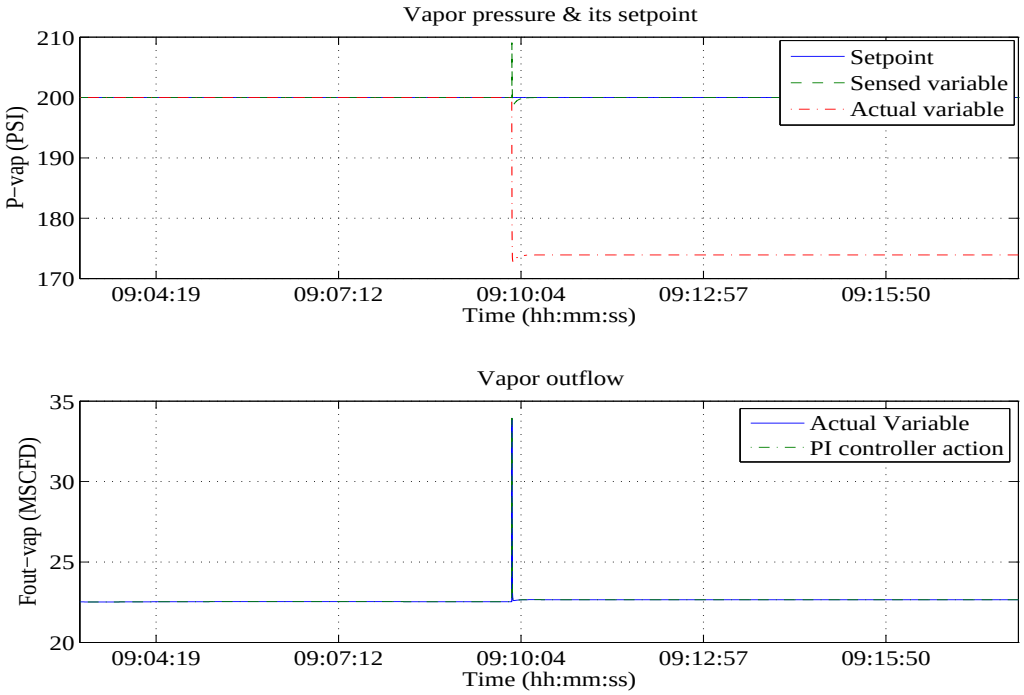

Figure 6.28: Limitation scenario A: Three-phase separator pressure logged by the FDIA agent

about this abnormal plant situation, as shown in the FDIA agent supervisory frame in table 6.13. The supervisory agent decides not to activate the fault accommodation task as the fault is identified an actuator fault (refer to the accommodation attributes in table 6.13). As a result, the fault accommodation parameters have a value of zero and the fault size and sign attributes have no facts, as the identified actuator fault size can not be estimated by the FDIA agent. It is very evident that something went wrong during this simulation scenario, as the system isolated the wrong fault (i.e., fault F10 instead of fault F5).

To analyze this situation, we compare the data record of the three-phase separator pressure measurement logged at the FDIA agent (refer to figure 6.28) and the same data record logged at the pilot plant agent (refer to figure 6.31). The comparison reveals that there is a significant difference between the two pressure measurement data records at the fault application instant. The pressure measurement logged at

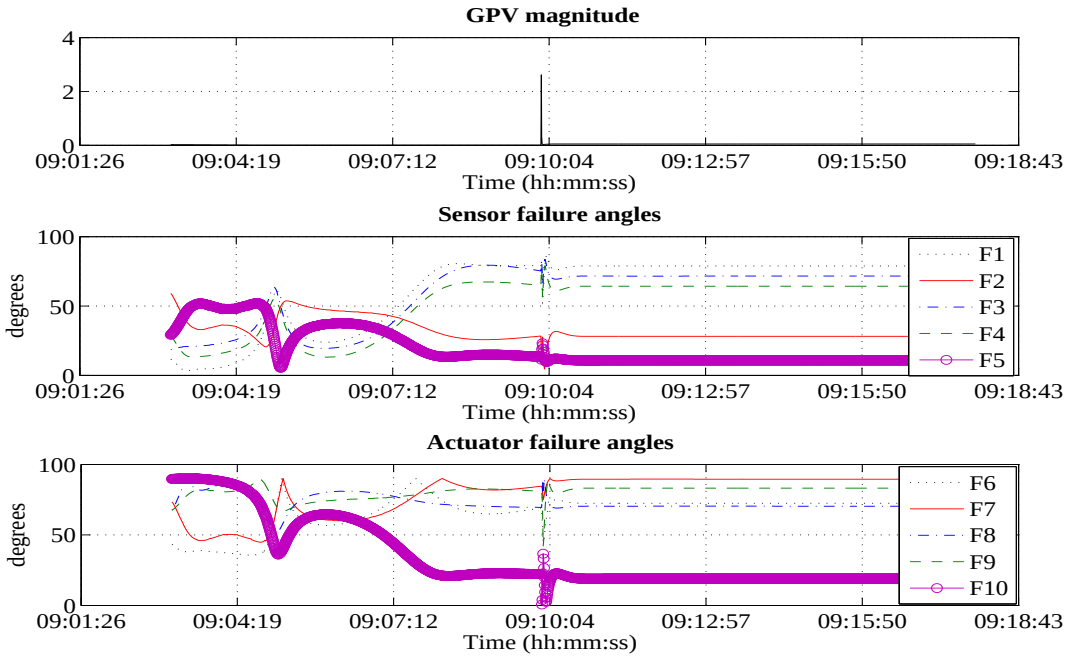

Figure 6.29: Limitation scenario A: FDIA agent diagnostic signals

the FDIA agent shows that the pressure spikes up to $P = 209$ PSI compared to a value of $P = 230$ PSI logged at the pilot plant agent during fault application. The same difference can be noticed in the three-phase separator gas outflow measurement data records. This measurement mismatch is due to the fast dynamics nature of the three-phase separator pressure, which caused the outlier removing task in the statistical preprocessing agent to clip the pressure and gas outflow measurements before being sent to the FDIA agent. This led to the wrong fault isolation decision made by the FDIA agent, and the wrong decisions made by the supervisory agent accordingly. This highlights the importance of embedding a safety net in the ICAM system prototype to compensate for the limitations of the system agents; for example, if the statistical preprocessing agent notified the other agents whenever it clipped a measurement they could be able to make allowances for that fact. Refining the statistical preprocessing agent so that pressure spikes were not treated as outliers would also be effective.

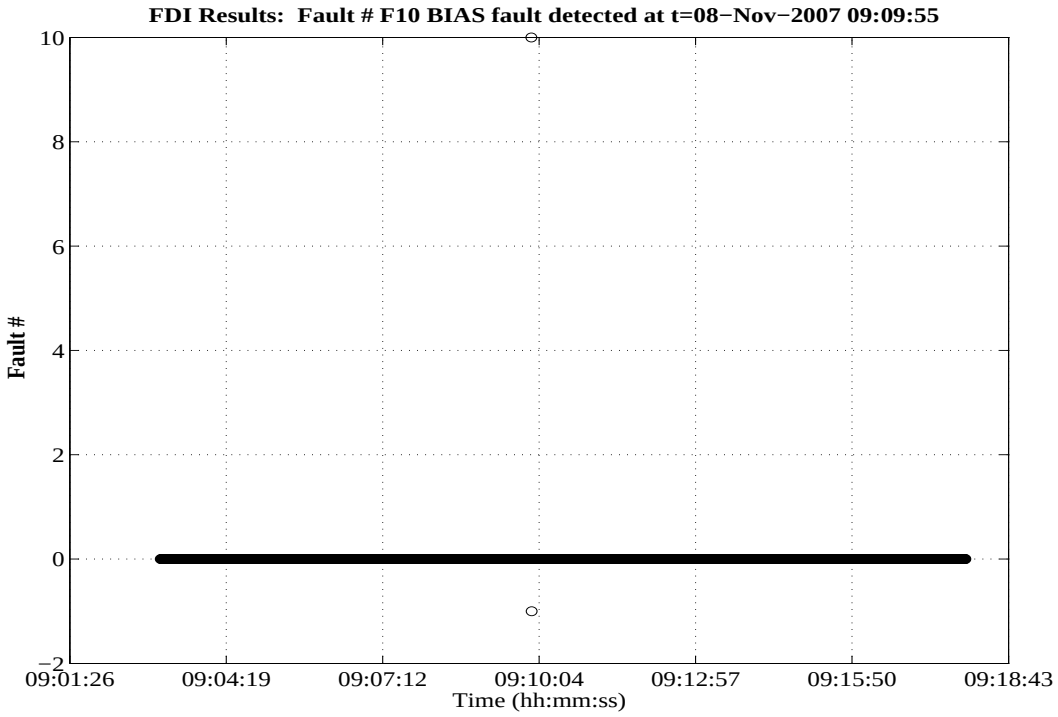

Figure 6.30: Limitation scenario A: FDIA agent fault display

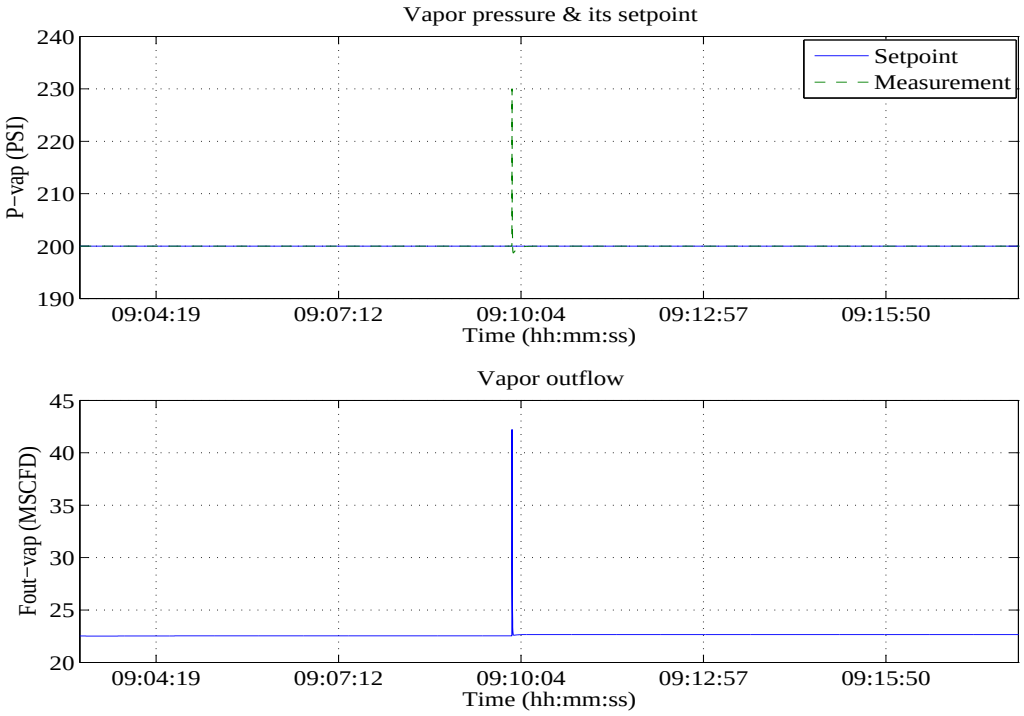

Figure 6.31: Limitation scenario A: Three-phase separator pressure logged by the pilot plant agent

Names	FDIA-OBJECT
Rank	3
State	simulate
Mpi comm	comm-world
Mpi comm size	4
G2 link status	connected
Mpi link status	connected
Decision	diagnose-faults
Simulation status	on
Mpi channel decision	no-decision
Model status	model-is-identified
Fdi design status	fdi-filter-is-designed
Fault	f10
Fault sign	none
Fault size	NaN
Fault type	bias
Fault time	"08-Nov-2007 09:09:55"
Accommodation status	none
Recursive fault size estimation status	none
Acknowledge fault	off

Table 6.13: Limitation scenario A: FDIA agent supervisory frame

6.5.2 Scenario B: Oil-well production decrease

The productivity of offshore oil wells and fields may decrease with time and demand, which leads to a decrease in the oil flow to the production facility. In order to analyze the impact of such change on the logical behavior of the ICAM system prototype, we introduce a 20% sudden decrease in the oil component of the oil-well incoming flow at time T_{dist} = 14:58:00 (refer to the symbol B in figure 6.1).

Figure 6.32 shows the two-phase liquid volume measurement logged at the FDIA agent, where the disturbance effect on the liquid volume is corrected by the PI controller by adjusting the liquid outflow valve accordingly. The disturbance is rejected in a time period of about seven minutes, during which the GPV magnitude increases and stays at value of 0.4, as shown in the top plot of figure 6.33. Interestingly enough, none of the GPV angles go to a low level except for the angle of fault F7 for a very short time period, as illustrated in the middle and bottom plots of figure 6.33. The local decision making logic of the FDIA agent declares that a fault F7 is isolated for a short time period, as shown in figure 6.34. The decision of the FDIA agent then takes a value of -2, which corresponds to an undefined fault decision during steady state. The FDIA decision then changes to -1 for a longer time period, which represents an undefined fault during transient. This strange behavior can also be noticed in the GPV magnitude (refer to the top plot of figure 6.33), where the GPV magnitude seems to reach a steady state of about 0.2 for a very short time period and then increases for a longer time period before it is in a true steady state. Finally, the FDIA fault isolation decision settles down on a fault F7 (i.e., the last detected fault) till the end of the simulation scenario, although the lowest GPV angle is the one associated with fault F1.

In order to verify the FDIA fault isolation decision, we examine the FDIA agent supervisory frame, depicted in table 6.14. It is evident that the FDIA agent isolates a fault F7 of type ramp at time T_{fault} = 14:58:04. Fault F7 corresponds to a faulty

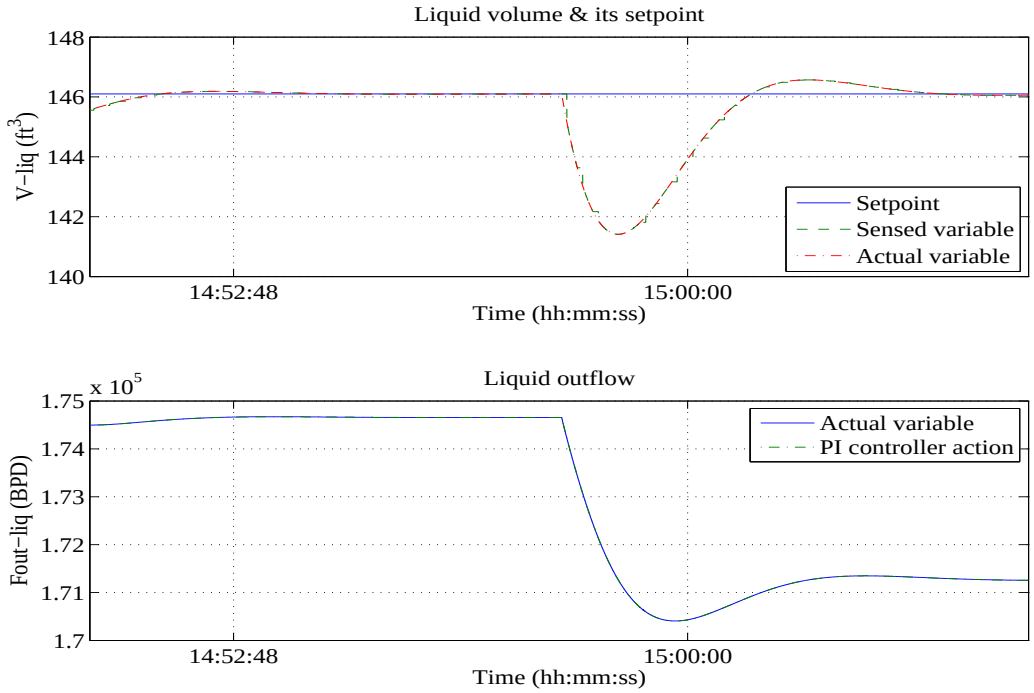

Figure 6.32: Limitation scenario B: Two-phase separator liquid volume logged by the FDIA agent

two-phase gas outflow valve (refer to table 6.1). The supervisory agent reasons about this situation and decides that no fault accommodation action should be taken, as indicated by the accommodation attributes in table 6.14. The fault sign and size attributes provide no new facts, as the identified fault is an actuator fault, whose size can not estimated by the FDIA agent. The FDIA agent can only estimate sensor faults by design.

To further analyze the results, we plot the two-phase separator pressure measurement and its associated gas outflow data records logged at the FDIA agent, as shown by figure 6.35. The gas pressure and outflow measurements show the effect of the oil-well incoming flow disturbance, which is rejected by the PI control loop PCL1 (refer to figure 6.1). Furthermore, there is no mismatch between the measured and actual data records for both process variables. To add to the situation complexity, the FDIA agent was not designed to isolate ramp actuator faults. Yet,

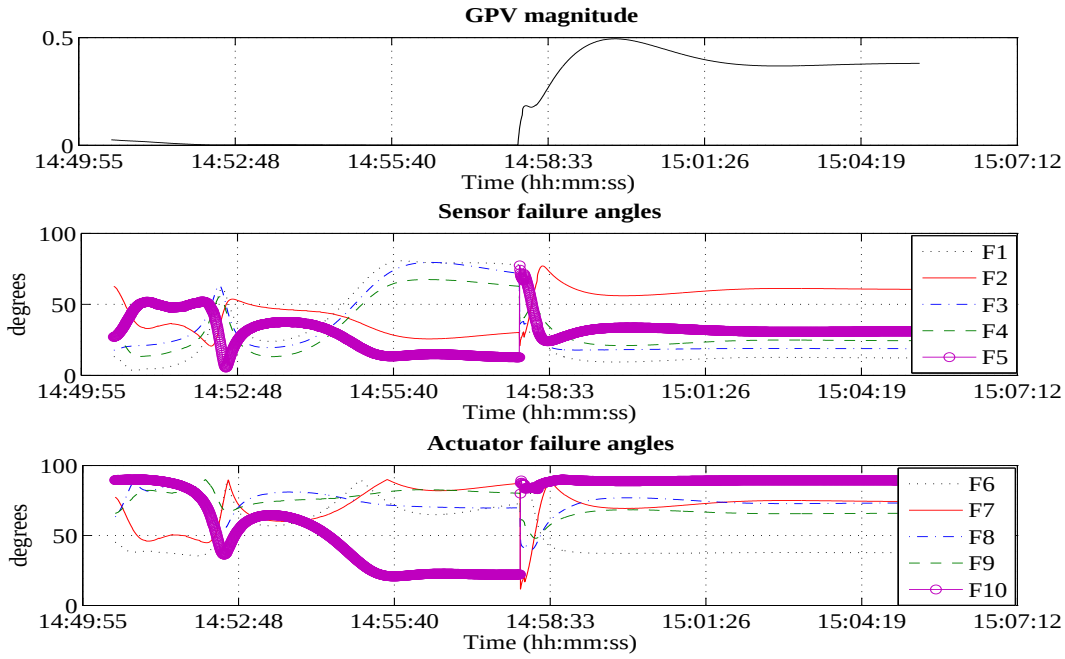

Figure 6.33: Limitation scenario B: FDIA agent diagnostic signals

it did declare that a ramp actuator (i.e., gas outflow valve) fault has occurred. This complex simulation situation, in which the FDIA agent generated confusing decisions, is attributed to the fact that the FDIA agent was not designed to decouple disturbances from sensor/actuator faults. This limitation is due to a lack of an analytic model that includes disturbance inputs; studies have demonstrated that known disturbances can be decoupled so they do not interfere with FDI [71]. Again, the necessity of embedding limitations of the ICAM system reactive agents in the knowledge base of the supervisory agent becomes crucial to guarantee robust and logically coherent system behavior.

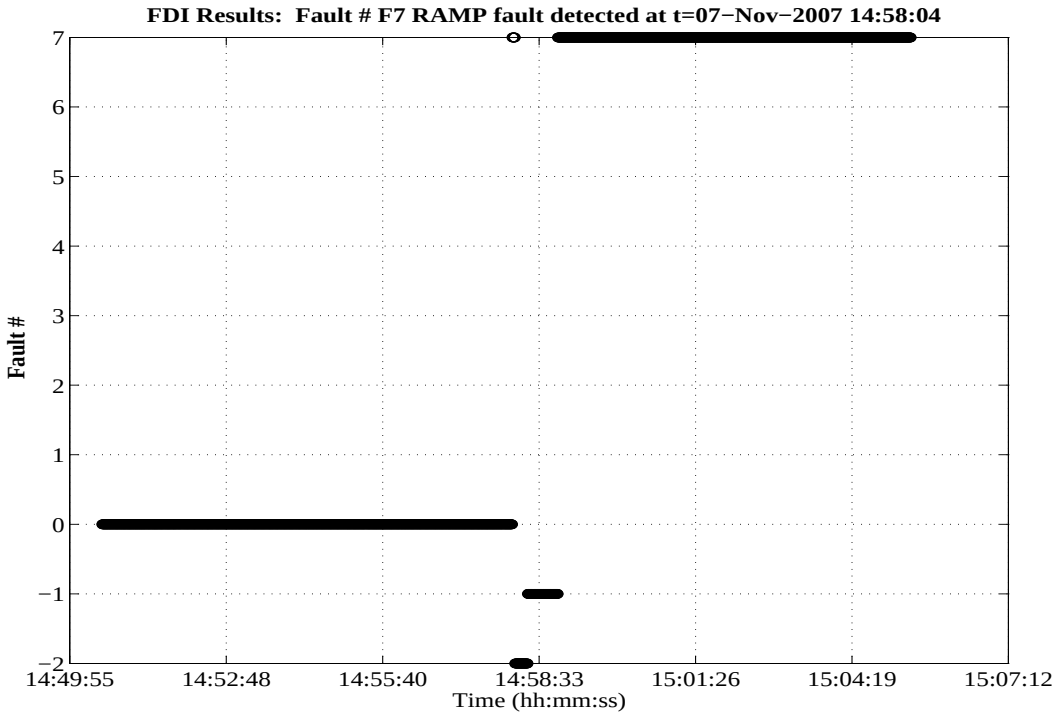

Figure 6.34: Limitation scenario B: FDIA agent fault display

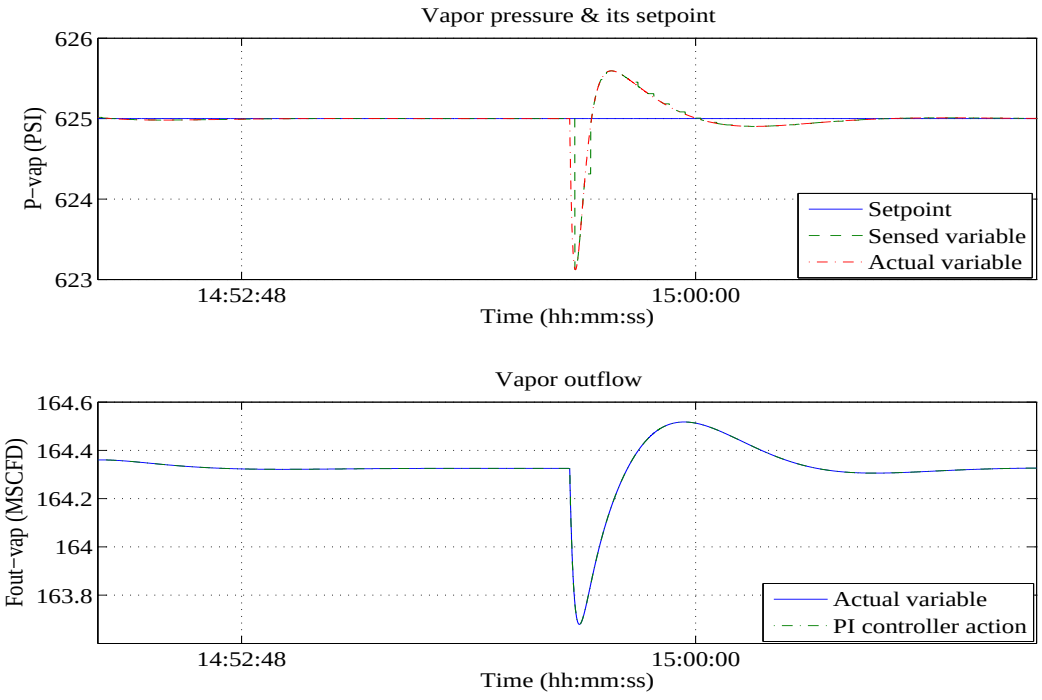

Figure 6.35: Limitation scenario B: Two-phase separator pressure logged by the FDIA agent

Names	FDIA-OBJECT
Rank	3
State	simulate
Mpi comm	comm-world
Mpi comm size	4
G2 link status	connected
Mpi link status	connected
Decision	diagnose-faults
Simulation status	on
Mpi channel decision	no-decision
Model status	model-is-identified
Fdi design status	fdi-filter-is-designed
Fault	f7
Fault sign	none
Fault size	NaN
Fault type	ramp
Fault time	"07-Nov-2007 14:58:04"
Accommodation status	acc-not-possible
Recursive fault size estimation status	recalculation-not-required
Acknowledge fault	off

Table 6.14: Limitation scenario B: FDIA agent supervisory frame

Chapter 7

Conclusions and Future Work

The automation of asset management for industrial process plants proved to be a very challenging problem, which we tried to investigate and to develop an innovative framework to solve in this thesis. The research revealed that the implementation of the solution has its own challenges, some of which have been tackled in this thesis. The rest of the challenges have been suggested as future work.

7.1 Summary and Conclusions

- A thorough literature review of intelligent asset management systems in industry and academia was conducted, which showed that this research area has been active for the past two decades. Most early research projects focused on the qualitative approach to diagnose the industrial plant during abnormal situations. Later, research focused on quantitative approaches, which focused on embedding plant models and more advanced data statistical processing algorithms. They also developed a basic framework to integrate the different system modules. However, most of the projects operated the different modules in a semi-automated manner and did not efficiently exploit artificial intelligence (AI) techniques to better manage the different modules of the system. The intelligent control and asset management (ICAM) system research project was introduced to build on previous research and to specifically address the

full automation and high performance aspects of such systems.

- The conceptual model of the ICAM system was defined based on the human cognition-affect (H-Cogaff) architecture, which we determined to represent a combination of the cognitive architecture and the multi-agent system (MAS) conceptual frameworks. The different agents of the ICAM system were properly distributed among the layers of the H-Cogaff architecture. A detailed functional description of the ICAM system agents was discussed. We also described the general logical behavior of the ICAM system based on Durffee's informal theory of coordination, which tackles the full automated interactions among the different agents of the ICAM system. We found that the H-Cogaff architecture addressed some aspects of Durffee's coordination theory. A development plan along with the appropriate analysis tools were suggested for designing such a complex system.

- The implementation requirements of the ICAM system were analyzed in terms of communications, AI supervision, and the reactive agent structure. The communication requirements of the ICAM system were analyzed and discussed after having conducted a comprehensive review of middleware (i.e., communications) technologies. Among the middleware technologies the Message Passing Interface (MPI) technology was chosen to meet ICAM's high performance requirements. The MPI communication requirements were further analyzed and refined, where the remote memory access (RMA) one-sided protocol was chosen due to its higher performance compared to other protocols. The artificial intelligence (AI) requirements of the ICAM system were analyzed in terms of knowledge representation and processing, and the appropriate AI paradigm. The rule-based reasoning AI paradigm was chosen, in particular the G2 rule-based expert system shell. The structure, implementation and deployment of the system agents were designed based on the MATLAB simulation environment.

As a result of the implementation requirements analysis, we found that MATLAB, the MPI communication library, and the G2 expert system shell form the best development platform for designing and prototyping complex intelligent multi-agent systems. In fact, the design space formed by these development tools allows designers to design and prototype complex systems that meet any degree of complexity.

- A simple prototype of the ICAM system was designed and developed in terms of the middleware layer, the intelligent supervisory layer, and the reactive agents layer. Three data communication channels were embedded in the middleware layer, namely the raw data channel, the processed data channel, and the plant model channel. A remote procedure call (RPC) based communication channel was incorporated in the middleware layer to connect the supervisory agent with the reactive agents of the system. The intelligent supervisory agent was designed, which supported a representative subset of the actual ICAM system functionality. This representation was developed from the ontology of the ICAM system, which also was designed as a class hierarchy. The supervisory agent rule-base, which runs the ICAM system prototype, was designed to meet the ICAM system pre-specified behavior. The design of the reactive agents of the ICAM system prototype was described in terms of their functional algorithms and their interaction with the supervisory agent. The deployment scheme of the system was also thoroughly discussed, where we defined the hardware, software development tools and network servers required to run the ICAM system prototype.

- The oil production facility simulation model, upon which the system's verification and validation were demonstrated, was described and discussed. We modeled the main three-phase separation process in the production facility. In fact, two main physical phenomena were modeled, namely, the hydrodynamics

of oil droplets separation in an aqueous phase, and the thermodynamics of light hydrocarbon gas flashing out of the oil liquid phase. A two-phase separator model was derived from the three-phase separator model. The two separation models were combined together to form a model of an oil production facility. A control system was also designed for the oil production facility model to control it around a specified operating point [80]. In order to emulate instrumentation failures, five actuator and five sensor faults were embedded in the facility model. The production facility simulation model can be used to design complex systems such as the ICAM system in both normal and abnormal operating situations.

- The verification and validation of the system were demonstrated, where several simulation scenarios were applied to the system to analyze its performance in real-time and its logical behavior. The first simulation scenario dealt with a bias sensor fault situation, in which the system showed good logical behavior as illustrated by the simulation results of the system reactive agents and the network activity analysis. The ICAM system prototype also behaved logically during the bias actuator fault application scenario, which was the second simulation scenario. A more complex simulation scenario was applied during the application of sensor fault of ramp type (i.e, a drifting sensor fault). The FDIA agent was required to recursively estimate the fault size and to generate the accommodation date in real time. The ICAM system behavior was satisfactory in general during this simulation scenario. However, the fault accommodation task was incomplete, as there was still a minor drift in the accommodated sensor measurement. This is due to the real-time distributed nature of the ICAM system prototype.

- To analyze the ICAM system prototype behavior and to identify its limitations during unexpected situations, two simulation scenarios were set up. The first

simulation scenario was designed to simulate a faulty sensor with fast dynamics. The FDIA agent decision making logic isolated a wrong fault, which led to wrong supervisory decisions. A further simulation results analysis revealed a mismatch between the faulty sensor measurement data records logged at the pilot plant and the FDIA agents. The data records mismatch was because of the fast dynamics nature of the sensor fault, which caused the outlier removal task in the statistical preprocessor agent to clip the sensor data during the fault application. The second simulation scenario was set up to emulate a decrease in the oil-well productivity. This scenario was more complex than the previous one, in which the FDIA agent isolated a fault that was not even specified in the design requirements. Again the supervisory agent generated the wrong decision. This highlights the importance of embedding a safety net in the ICAM system prototype to compensate for the limitations of the system agents.

- A thorough performance analysis of the ICAM system prototype was conducted during the fault accommodation mode, which demonstrated an excellent system performance in terms of computation/communication overlap. The execution cycles of the reactive agents nearly met the design specification of an agent execution cycle of 100 milliseconds. However, there was a minor computation bottleneck due to local data storage, which had a minor effect on the ICAM system performance. The communications between the reactive agents and the supervisory agent took an insignificant part of the agent execution cycle, which verifies the system design specifications of the G2 communication link. The overall performance of the ICAM system was excellent, but more research has to be done to further enhance it in future work.

7.2 System Limitations, Design Challenges, and Future Work

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 [78], 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 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 general parity vector (GPV) based 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.

- In order to address the complete asset management solution in process plants, several agents have to be embedded in the ICAM system prototype to manage

the process plants during normal situations. An optimization agent is essential to generate optimal material recipes and process variable set-points to guarantee higher product quality. Planning and scheduling agents are also essential to schedule operation plans in accordance with long term production plans. Furthermore, the addition of a real-time database management agent is vital for both high system performance and future scalability. Finally, a graphical user interface (GUI) agent must be added to the ICAM system to meet process operator interaction requirements.

- 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.

- During abnormal situations hundreds of alarms are initiated, leading to alarm flooding. This results in the operator missing important alarms. Proper asset management requires proper alarm and event management techniques in addition to good operator decision support. The incorporation of alarm management techniques that can dynamically prioritize important alarms and suppress unnecessary alarms would definitely enhance the ICAM system performance. The alarm management agent would interact with the FDIA agent to better identify the most important alarms that have to be dealt with.

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

Bibliography

[1] *Asset management primer*, Tech. report, US Department of Transportation, December 1999.

[2] *Middleware: Software technology roadmap*, `http://www.sei.cmu.edu/str/descriptions/middleware.html`, January 2007.

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

[4] K. D. Althoff, E. Auriol, R. Barletta, and M. Manago, *A review of industrial case-based reasoning tools*, Tech. report, Al lntelligence, Oxford, UK, 1995.

[5] K. Arnold and M. Stewart, *Surface production operation: Design of oil-handling systems and facilities*, 2nd ed., vol. 1, Butterworth-Heinemann, Woburn, MA, 1999.

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

[7] P. A. Bernstein, *Middleware: A model for distributed services*, Communications of the ACM **39** (1996), no. 2, 86–97.

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

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

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

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

[12] D. D. Corkill, *Collaborating Software: Blackboard and Multi-Agent Systems & the Future*, Proceedings of the International Lisp Conference (New York, New York), October 2003.

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

[14] M. M. Dionne, *The dynamic simulation of a three phase separator*, Master's thesis, University of Calgary, 1998.

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

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

[17] E. Durfee and T. Montgomery, *Coordination as distributed search in a hierarchical behavior space*, IEEE Transactions on Systems, Man, and Cybernetics **21** (1991), no. 6, 1363–1378.

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

[19] H. Wörn et al, DIAMOND*: Distributed multi-agent architecture for monitoring and diagnosis*, Production Planning and Control **15** (2004), 189–200.

[20] W. Gropp et al, MPI*: The complete reference*, vol. 2, The MIT Press, 1998.

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

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

[23] F. Figueroa, R. Holland, J. Schmalzel, D. Duncavage, A. Crocker, and R. Alena, ISHM *implementation for constellation systems*, 42nd AIAA/ASME/SAE/ASEE Joing Propulsion Conference and Exhibit (Sacramento, CA, USA), 9-12 July 2006.

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

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

[26] R. G. E. Franks, *Modeling and simulation in chemical engineering*, Wiley, 1972.

[27] A. Gaddah and T. Kunz, *A survey of middleware paradigms for mobile computing*, Tech. Report SCE-03-16, Carleton University Systems and Computing Engineering, July 2003.

[28] C. Garcia-Beltran, M. Exel, and S. Gentil, *An interactive toot for causal graph modeling for supervision purposes*, In Proc. IEEE International Symposium on Intelligent Control (ISIC) (Huston, Texas), 5-8 October 2003, pp. 866– 871.

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

[30] L. Geng, Z. Chen, C. W. Chan, and G. H. Huang, *An intelligent decision support system for management of petroleum contaminated sites*, Expert Systems with Applications **20** (2001), 251–260.

[31] GenSym Corporation, Burlington, Massachusetts, G2 *for application developers reference manual*, 8.0 ed., December 2005.

[32] J. Gertler, *Survey of model based failure detction and isolation in complex plants*, IEEE Conteol Systems Magazine (1988).

[33] J. Giarratano and G. Riley, *Expert systems, principles and programming*, 3rd ed., PWS Publishing Company, 1998.

[34] W. Gropp and E. Lusk, *Tuning* MPI *applications for peak performance*, www.mcs.anl.gov/Projects/mpi/tutorials/perf, Argonne National Laborary.

[35] W. Gropp, E. Lusk, and A. Skjellum, *Using* MPI*: portable parallel programming with the message-passing interface*, 2nd ed., Scientific and Engineering Computation, MIT Press, Cambridge, Massachusetts, 1999.

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

[37] B. Hafskjold, H. K. Celius, and O. M. Aamo, *A new mathematical model for oil/water separation in pipes and tanks*, SPE Production & Facilities **14** (1999), no. 1, 30–36.

[38] A. Hallanger, F. Soenstaboe, and T. Knutsen, *A simulation model for three-phase gravity separators*, Proceedings of SPE Annual Technical Conference and Exhibition (Denver, Colorado), SPE, October 1996, pp. 695–706.

[39] J. He, *Neuro-fuzzy based fault diagnosis for nonlinear processes*, Master's thesis, University of New Brunswick, May 2006.

[40] C. D. Holland, *Fundamentals and modeling of separation processes: Absorption, distillation, evaporation, and extraction*, Prentice-Hall, Englewood Cliffs, N.J., 1974.

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

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

[43] S. A. K. Jeelani, R. Hosig, and E. J. Windhab, *Kinetics of low reynolds number creaming and coalescence in droplet dispersions*, AIChE Journal **51** (2005), no. 1, 149–161.

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

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

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

[47] 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*, Proceedings of the 5th IFAC Symposium Fault Detection, Supervision and Safety of Technical Processes - SAFEPROCESS05 (Washington D.C.), 2003.

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

[49] L. M. Kristensen, J. B. Jrgensen, and K. Jensen, *Lectures on concurrency and petri nets: advances in petri nets*, ch. Application of coloured petri nets in system development, pp. 626–685, Springer-Verlag, 2004.

[50] W. Larimore, Multivariable System Identification Workshop (Fredericton, New Brunswick), University of New Brunswick, 31 October – 2 November 2005.

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

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

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

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

[55] J. Liebowitz, *The handbook of applied expert systems*, CRC Press, Boca Raton, FL, 1998.

[56] L. Ljung, *System identification - theory for the user*, 2nd ed ed., PTR Prentice Hall, Upper Saddle River, N.J., 1999.

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

[58] R. Matania1, *Interoperability and integration oil and gas science and technologyof industrial software tools*, Oil and Gas Science and Technology **60** (2005), no. 4, 617–627.

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

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

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

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

[63] D. Mylaraswamy, DKit*: a blackboard-based, distributed, multi-expert environment for abnormal situation management*, Ph.D. thesis, Purdue University, 1996.

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

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

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

[67] A. Norvilas, A. Negiz, J. DeCicco, and A. Cinar, *Intelligent process monitoring by interfacing knowledge-based systems and multivariate statistical monitoring.*, Journal of Process Control (2000), no. 10, 341–350.

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

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

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

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

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

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

[74] H. Pinus, *Middleware: Past and present a comparison*, `http://www.research.umbc.edu/~dgorin1/451/middleware/middleware.pdf`, June 2004.

[75] M. L. Powers, *Analysis of gravity separation in freewater knockouts*, SPE Production Engineering **5** (1990), no. 1, 52–58.

[76] H. E. Rauch, *Fault diagnosis and control reconfiguation*, IEEE Control Systems Magazine (1994).

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

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

[79] A. F. Sayda and J. H. Taylor, *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), 18-20 June 2007, pp. 851–858.

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

[81] A. F. Sayda and J. H. Taylor, *Toward a practical multi-agent system for integrated control and asset management of petroleum production facilities*, IEEE International Symposium on Intelligent Control (ISIC) (Singapore), 1–3 October 2007.

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

[83] M. J. H. Simmons, E. Komonibo, B. J. Azzopardi, and D. R. Dick, *Residence time distribution and flow behavior within primary crude oil-water separators*

treating well-head fluids, Chemical Engineering Research & Design **82** (2004), no. A10, 1383–1390.

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

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

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

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

[88] E. Tarifa and N. Scenna, *Fault diagnosis, directed graphs, and fuzzy logic*, Computer and Chemical Engineering **21** (1977), S649–S654.

[89] E. Tatara and A. Cinar, *An intelligent system for multivariate statistical process monitoring and diagnosis.*, ISA Transactions **41** (2002), 255–270.

[90] J. H. Taylor, *Expert systems for computer-aided control system design and engineering*, Proc. Chemical Process Control III (Asilomar, CA), January 1986, pp. 807–838.

[91] J. H. Taylor, *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, 2004.

[92] 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*, Advanced Sensing, Modelling, and Control of Materials Processing (Warrendale, PA), Ed. by E. F. Matthys and B. Kushner, TMS Publications, 1991.

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

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

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

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

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

[98] R. Taylor and A. Lucia, *Modeling and analysis of multicomponent separation processes*, Separation Systems and Design (1995), 19–28.

[99] P. vanOverschee and B. DeMoor, *Subspace identification of linear systems: Theory, implementation, applications*, Kluwer Academic Publishers, 1996.

[100] H. Vedam, Op-Aide*: an intelligent operator decision support system for diagnosis and assessment of abnormal situations in process plants.*, Ph.D. thesis, Purdue University, 1999.

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

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

[103] 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 **27** (2003), no. 3, 293–346.

[104] 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 **3** (1987), no. 1, 45–72.

[105] H. Wang and C. Wang, APACS*: A multi-agent system with repository support*, Knowledge-Based Systems **9** (1996), no. 5, 329–337.

[106] H. Wang and C. Wang, *Intelligent agents in the nuclear industry*, Computer **30** (1997), no. 11, 28–34.

[107] L. Wang, *On-line fault diagnosis using signed digraphs*, Master's thesis, University of New Brunswick, May 2006.

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

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

[110] S. Y. Yim, H. G. Ananthakumar, L. Benabbas, A. Horch, R. Drath, and N. F. Thornhill, *Using process topology in plant-wide control loop performance assessment.*, Computers and Chemical Engineering (2006), no. 31, 86–99.

Appendix A

Modeling and Control of Three-Phase Gravity Separators in Oil Production Facilities

A.1 INTRODUCTION

The function of an oil production facility is to separate the oil well stream into three components or "phases" (oil, gas, and water), and process these phases into some marketable products or dispose of them in an environmentally acceptable manner. In mechanical devices called "separators", gas is flashed from the liquids and "free water" is separated from the oil. These steps remove enough light hydrocarbons to produce a stable crude oil with the volatility (i.e., vapor pressure) to meet sales criteria. Separators are classified as "two-phase" if they separate gas from the total liquid stream and "three-phase" if they also separate the liquid stream into its crude oil and water components. The gas that is separated is compressed and treated for sales [5]. Modeling such facilities has become very crucial for controller design, fault detection and isolation, process optimization, and dynamic simulation. In this appendix, we focus on three-phase gravity separators as they form the main processes in the upstream petroleum industry, and have a significant economic impact on produced oil quality.

Three-phase separators have rich and complex dynamics, which span from hydrodynamics to thermodynamics and conservation laws. Many modeling techniques and

approaches have been used to model three-phase separators. As far as the thermodynamics aspects of the separator are concerned (i.e., the oil and gas phases), many modeling approaches have been suggested in the literature. The phase equilibrium modeling approach has been used for 50 years and has provided satisfactory results for such equipment as flash tanks and distillation columns [98]. The basic equations in this approach are used to describe the material balances, equilibrium relations, the composition summation equations, and the enthalpy equations. Non-equilibrium models have been developed to describe real physical separation processes; other modeling approaches were also considered such as the computational model, the collocation model, and the bubble residence contact time model [14].

Historically, the hydrodynamics of the separator's aqueous part have been modeled using complex mathematical-numerical models, which describe the coalescence and settling of oil droplets in oil-water dispersions. Such models take into account separator dimensions, flow rates, fluid physical properties, fluid quality and drop size distribution. The output of these models is the quality of the output oil [37]. Other models, which describe the kinetics of low Reynolds number coalescence of oil droplets in water-oil dispersions, have been developed to give the volumes of separated continuous-phase and coalesced drops [43]. A computational fluid dynamic (CFD) model was developed to model the hydrodynamics of a three-phase separator, based on the time averaging of Navier-Stokes equations for three phases; it takes into consideration the non-ideal flow due to inlet /outlets and internal equipment for separation enhancement [38]. The "alternative path model approach", which exploits the residence time distribution (RTD) of both oil and aqueous phases in three-phase separators, was developed to give a quantitative description of hydrodynamics and mixing in the aqueous phase [83]. Powers [75] extended the American Petroleum Institute (API) gravity separator design criteria to design free-water knockout vessels for better capacity and performance. Powers showed that the API design criteria

can handle non-ideal flow by performing practical RTD experiments.

We extend the API static design criteria to model the hydrodynamics of three phase separators, which results in a simpler modeling approach. Furthermore, a simple phase equilibrium model is developed to model the thermodynamic aspects of the separator. We describe the operation of gravity three-phase separators in section A.2. A dynamic model of the separator is developed for each phase in section A.3, where it can be used to estimate the steady state flows and to study the separator behavior during other operating conditions. An oil production facility simulation model is designed, implemented and tested to validate and demonstrate the separator behavior during normal operation and upsets in section A.4. Finally, simulation results are discussed and summarized in section A.5.

A.2 Three-phase gravity separation process description

Three-phase separators are designed to separate and remove the free water from the mixture of crude oil and water. Figure A.1 is a schematic of a three phase horizontal separator. The fluid enters the separator and hits an inlet diverter. This sudden change in momentum does the initial gross separation of liquid and vapor. In most designs, the inlet diverter contains a downcomer that directs the liquid flow below the oil/water interface. This forces the inlet mixture of oil and water to mix with the water continuous phase (i.e., aqueous phase) in the bottom of the vessel and rise to the oil/water interface. This process is called "water-washing"; it promotes the coalescence of water droplets which are entrained in the oil continuous phase. The inlet diverter assures that little gas is carried with the liquid and assures that the liquid is not injected above the gas/oil or oil/water interface, which would mix the liquid retained in the vessel and make control of the oil/water interface difficult.

Some of the gas flows over the inlet diverter and then horizontally through the

gravity settling section above the liquid. As the gas flows through this section, small drops of liquid that were entrained in the gas and not separated by the inlet diverter are separated out by gravity and fall to the gas-liquid interface. Some of the drops are of such a small diameter that they are not easily separated in the gravity settling section. Before the gas leaves the vessel it passes through a coalescing section or mist extractor to coalesce and remove them before the gas leaves the vessel.

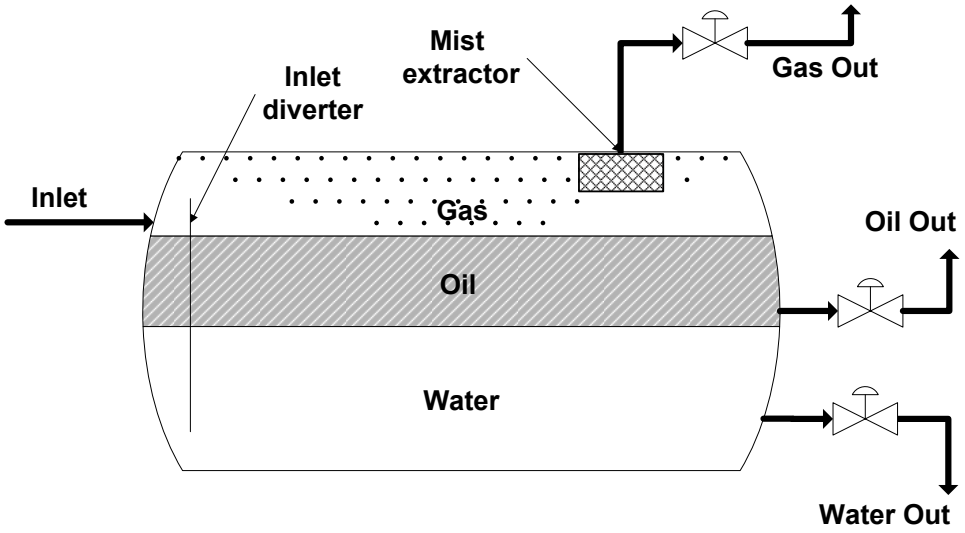

Figure A.1: Three phase horizontal separator schematic.

A.3 Three phase gravity separator mathematical modeling

When the hydrocarbon fluid stream enters a three-phase separator, two distinctive phenomena take place. The first phenomenon is fluid dynamic, which is characterized by the gravity separation of oil and water droplets entrained in the aqueous and the oil phases respectively, the gravity separation of gas bubbles entrained in the stream, and the gravity separation of liquid droplets which are dispersed in the gas phase. The second phenomenon is thermodynamic, in the sense that some light hydrocarbons and gas solution flash out the oil phase and reach a state of equilibrium due to the pressure drop in the separator. Due to the complexity of such phenomena,

we are going to focus on the hydrodynamic separation of oil droplets entrained in the aqueous phase and the thermodynamic separation of gas and light hydrocarbons from the oil phase. This decision is justified by the fact that the water washing process minimizes the water entrained in the oil phase. Furthermore, preceding gravity separation processes minimize the amounts of gas entrained in the main stream.

Figure A.2 illustrates the simplified separation process, where an oil-well fluid with molar flow F_{in} and gas, oil, and water molar fractions Z_g, Z_o, Z_w respectively enters the separator. The hydrocarbon component of the fluid separates into two parts; the first stream F_{h1} separates by gravity and enters the oil phase, and the second stream F_{h2} stays in the aqueous phase due to incomplete separation. The liquid discharge from the aqueous phase $F_{W_{out}}$ is a combination of the dumped water stream F_W plus the unseparated hydrocarbon stream F_{h2}. The gas component in the separated hydrocarbon stream, which enters the oil phase, separates into two parts; the first gas stream F_{g1} flashes out of the oil phase due to the pressure drop in the separator, and the second gas stream F_{g2} stays dissolved in the oil phase. The oil discharge $F_{o_{out}}$ from the separator contains the oil component of the separated hydrocarbon F_o and the dissolved gas component F_{g2}. The flashed gas $F_{g_{out}}$ flows out of the separator for further processing.

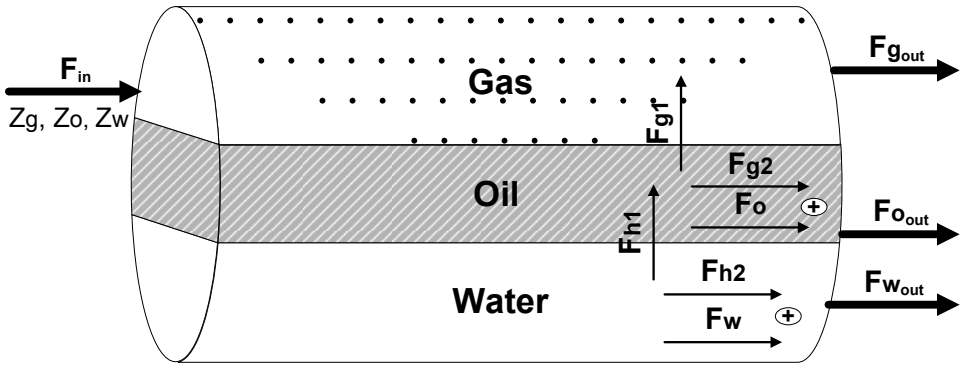

Figure A.2: Main separated component streams in three-phase gravity separator

We model the dynamics of each phase of the separator in the subsequent sections,

to simplify the modeling process. Additionally, some simplifying assumptions have to be made. The separation processes are assumed to be isothermal in all phases of the separator at $100\,^oF$. We also assume that the flow pattern in the liquid phases is plug flow, especially in the aqueous phase. Furthermore, the oil droplets in the aqueous phase have a uniform droplet size distribution with a diameter of $d_m = 500$ micron. The oil droplets' rising velocities are assumed to obey Stokes' law. We model the equilibrium thermodynamics phenomenon under the assumption that Raoult's law is valid. We assume that only one light hydrocarbon gas flashes out the oil phase into the gas phase, namely methane. Methane in the vapor phase is also assumed to be an ideal gas (i.e., the ideal gas law applies). Finally, there is liquid-vapor equilibrium at the oil surface and liquid-liquid equilibrium at the water-oil interface.

A.3.1 The aqueous phase

In order to model the aqueous phase of the separator, we will follow the API static design criteria under the usual simplifying assumptions. The API specification permits hydrocarbon droplets (i.e., oil and dissolved light gas) of design diameter to rise from the bottom of the separator to the surface during the water retention period, as illustrated in figure A.3. A hydrocarbon droplet located on the cylinder bottom has the greatest distance to traverse to the oil-water interface. Therefore, modeling the oil separation hydrodynamics based on removal of this droplet would ensure removal of all others of the same or larger diameter. Given the simplifying assumptions, the traversing hydrocarbon droplet on its path to the oil-water interface is subjected to a vertical rising velocity component v_v governed by Stokes' law, and a horizontal velocity component v_h governed by the plug flow pattern of the aqueous phase. The vertical velocity component is estimated from Stokes' law by equation (A.1):

$$v_v = 1.7886 \times 10^{-6} \frac{(SG_h - SG_w) d_m^2}{\mu_w} \tag{A.1}$$

where SG_h, SG_w are the specific gravities of the hydrocarbon droplets and water, respectively, d_m is the droplet diameter in microns, and μ_w is the water viscosity in CP at $100\,^oF$. The horizontal velocity component is estimated from the aqueous phase retention as $v_h = L\,/\,\tau$, where L is the length of the separator and $\tau = V_{wat}\,/\,F_{wat}$ is the retention time of the aqueous phase; V_{wat} is the volume of the aqueous phase, and F_{wat} is the water outflow. The level of the oil-water interface h is determined from equations (A.2):

$$\begin{aligned} A_c &= V_{wat}/L \\ &= R^2\theta \,-\, 0.5R^2\sin(2\theta) \\ h &= R(1-\cos(\theta)) \end{aligned} \tag{A.2}$$

where A_c is the cross-sectional area of the aqueous phase, R is the separator radius and θ is the angle which defines the circle sector of the cross sectional area A_c. The angle Φ of the longest droplet path to the oil-water interface can be estimated from equation (A.3):

$$\Phi = \arctan\frac{v_v}{v_h} \tag{A.3}$$

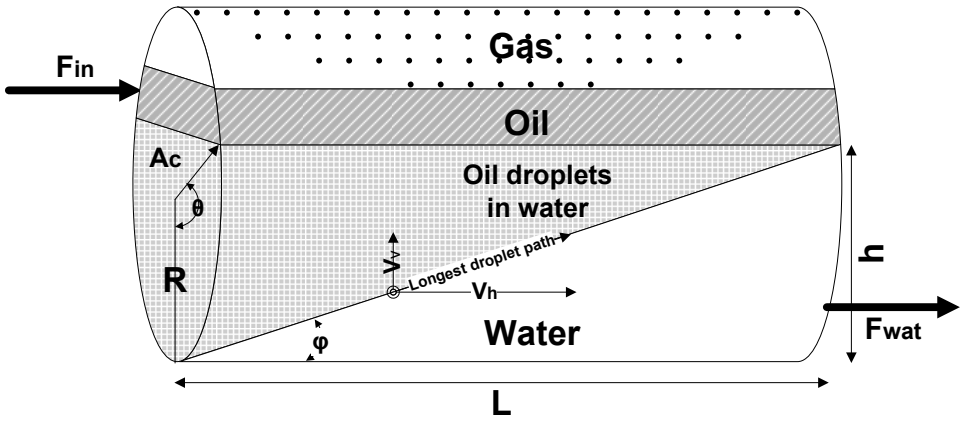

Figure A.3: Oil separation hydrodynamics under normal operation conditions

The design parameters $\{A_c,\, h,\, \theta,\, \Phi\}$ of the aqueous phase will take their nominal values under normal operating conditions of the three-phase separator, i.e., for the nominal value of F_{wat} which leads to complete separation, as shown in figure A.3. However, our model must also be valid for off-nominal values. This is complicated at higher flow values, since we can no longer achieve complete separation. Let us assume that the water outflow F_{wat} has increased by a value of ΔF_{wat} due to a corresponding increase in inflow. This will result in an increase in v_h to $v_h + \Delta v_h$ and an angle change of the longest path of a traversing hydrocarbon droplet from Φ to $\Phi_1 < \Phi$. Figure A.4 illustrates the concept of *virtually extending the tank* so that complete separation would be achieved at $L_1 = L + \Delta L$, although this is fictional. Assuming that the design parameters $\{A_c,\, h,\, \theta\}$ remain the same, as shown in figure A.4, we have:

$$\begin{aligned} \Phi_1 &= \arctan \frac{(v_v + \delta v_v)}{v_h} \\ L_1 &= h \cot(\Phi_1) \end{aligned} \tag{A.4}$$

We have to make a simplifying assumption in order to estimate the volume fraction of unseparated hydrocarbon, ε. As shown in figure A.5 (top), we assume that the unseparated oil droplets in the aqueous phase form a "tail" extending into the virtual separator extension, as also shown by dashed lines in figure A.5 (bottom, labeled S3) that undergoes turbulent flow and exits with the water. The accuracy of this assumption is, of course, dependent upon the geometry of the tank and structure of the water and oil outlets.

Under this assumption, region S3 in the bottom figure represents the volume of the unseparated hydrocarbon fluid V_{S3}. It can be seen from figure A.5 that region S3 is the difference between the hydrocarbon fluid volume in the virtual separator (represented by region S1), V_{S1} and the hydrocarbon fluid volume in the actual separator,

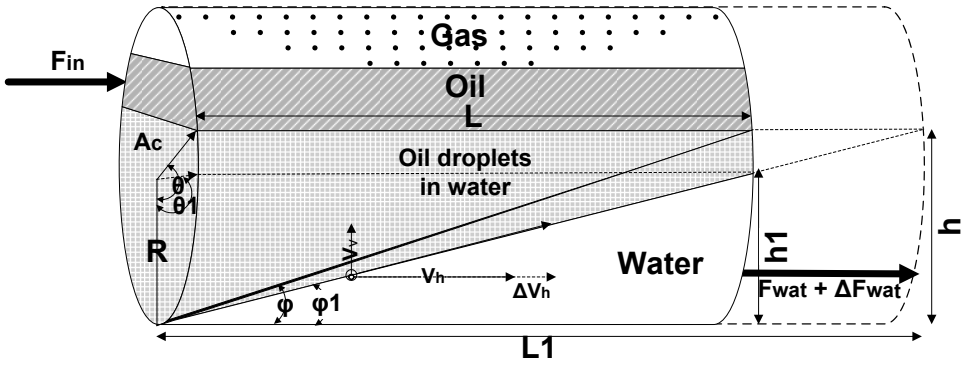

Figure A.4: Oil separation hydrodynamics under high water outflow condition

V_{S2} (represented by region S2). The volume V_{S1} can be calculated as the difference between the volume of the cylindrical segment defined by the parameters $\{h,\, L_1,\, \theta\}$ and the cylindrical wedge parameterized by $\{h,\, L_1,\, \Phi_1\}$, as in equation (A.5):

$$V_{S1} = R^2 L_1 \{\theta - 0.5\sin(2\theta) - \frac{3\sin\theta - 3\theta\cos\theta - \sin^3\theta}{3(1-\cos\theta)}\} \tag{A.5}$$

Furthermore, again referring to figure A.5, the volume V_{S2} can be estimated as the difference between the volume of the cylindrical segment parameterized by $\{h,\, L,\, \theta\}$ and the cylindrical wedge parameterized by $\{h_1,\, \theta_1,\, L,\, \Phi_1\}$, as in equation (A.6):

$$V_{S2} = R^2 L \{\theta - 0.5\sin(2\theta) - \frac{3\sin\theta_1 - 3\theta_1\cos\theta_1 - \sin^3\theta_1}{3(1-\cos\theta_1)}\} \tag{A.6}$$

where the virtual oil-water interface h_1 and angle θ_1 are defined by equations (A.7):

$$\begin{aligned} h_1 &= L\tan(\Phi_1) \\ \theta_1 &= \arccos(1 - \frac{h_1}{R}) \end{aligned} \tag{A.7}$$

Consequently, we can estimate the unseparated hydrocarbon fluid volume fraction ε from equation (A.8):

$$\varepsilon = \begin{cases} 1 - \frac{V_{S2}}{V_{S1}} \ , & L_1 > L \\ 0 & else \end{cases} \tag{A.8}$$

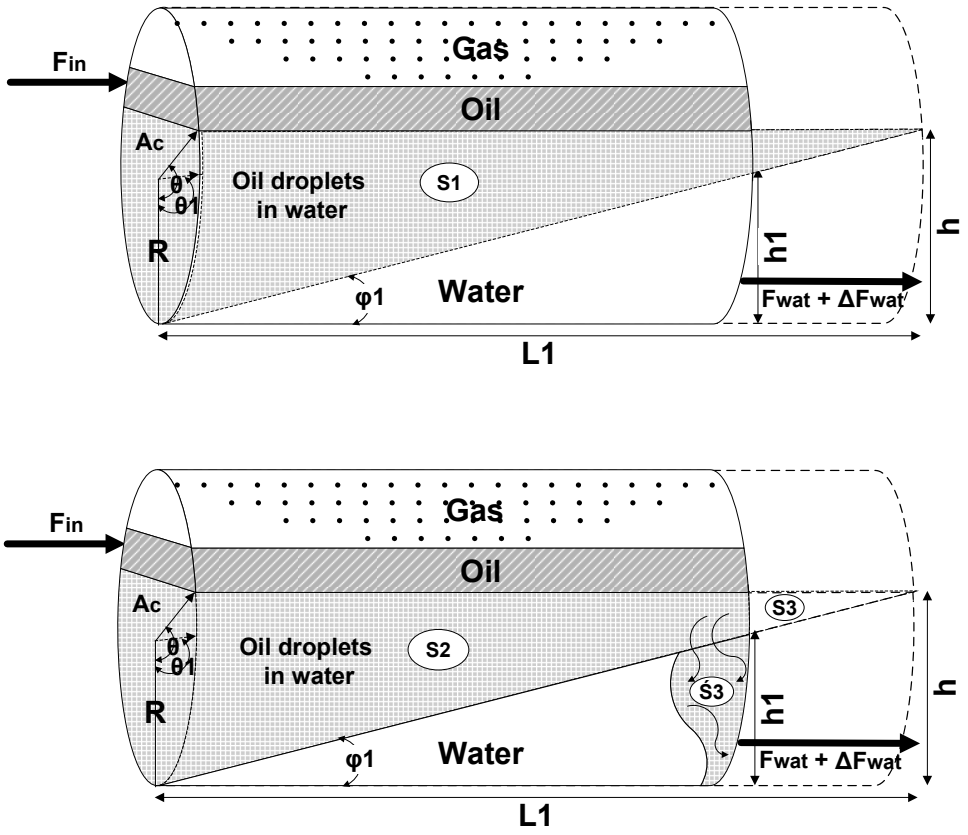

Figure A.5: Unseparated hydrocarbon fluid volume under high water outflow condition

Having estimated the unseparated hydrocarbon fluid volume fraction ε, we can calculate the separated and unseparated volumetric flow components of the hydrocarbon fluid F_{h1_v}, F_{h2_v} respectively. Finally we can write the dynamic material balance of the aqueous phase by using equations (A.9), after we convert the molar flows to volumetric flows:

$$
\begin{aligned}
F_{h1_v} &= \frac{\varepsilon(Z_g + Z_o)F_{in}Mw_h}{62.43SG_h} \\
F_{h2_v} &= \frac{(1-\varepsilon)(Z_g + Z_o)F_{in}Mw_h}{62.43SG_h} \\
F_{W_{out}} &= \frac{Z_w F_{in} Mw_w}{62.43SG_w} + F_{h2_v} \\
\frac{dV_{wat}}{dt} &= \frac{F_{in}Mw_{in}}{62.43SG_{in}} - F_{W_{out}} - F_{h1_v}
\end{aligned} \tag{A.9}
$$

where $\{Mw_h,\, Mw_w,\, Mw_{in}\}$ are the hydrocarbon, water, and incoming mixture molecular weights; $\{SG_h,\, SG_w,\, SG_{in}\}$ are the hydrocarbon, water, and incoming mixture specific gravities; V_{wat} is the aqueous phase volume; and $F_{W_{out}}$ is the water discharge volumetric outflow.

A.3.2 The oil phase

In order to model the thermodynamic phenomenon in the oil phase, we first do the flash calculations to estimate the amounts of gas which will flash out of solution. Since we assumed that ideal phase equilibrium state is valid, then by applying Raoult's law we can tell how much methane will stay entrained in the oil phase. Raoult's law relates the vapor pressure of components to the composition of the solution. This can be formulated mathematically as $y_i P = x_i P_{v_i}$ where y_i is the mole fraction of the component i in the vapor phase, x_i is the mole fraction of component i in the liquid phase, P is the total pressure of the vapor phase (i.e., the separator working pressure), and P_{v_i} is the vapor pressure of component i [26, 40].

Since we have only one flashing light hydrocarbon (i.e., methane), this implies that the mole fraction of methane in the vapor phase is $y = 1$, and that the mole fraction of methane entrained in the liquid phase is $x = P/P_v$. Given the composition $\{Z_{g1},\, Z_{o1}\}$ of the separated hydrocarbon stream F_{h1}, we can estimate the amounts of flashing methane F_{g1} and dissolved methane in the oil phase F_{g2}. The oil discharge flow $F_{o_{out}}$ can also be estimated along with its average molecular weight Mw_{o1} and

its specific gravity SG_{o1}. The complete dynamic model of the oil phase, which is given by equations (A.10), can be then formulated by taking the material balance:

$$\begin{aligned}
F_{g1} &= (1-x)Z_{g1}F_{h1} \\
F_{g2} &= xZ_{g1}F_{h1} \\
F_{o_{out}} &= F_o + F_{g2} \\
\frac{dN_{oil}}{dt} &= F_{h1} - F_{g1} - F_{o_{out}} \\
Mw_{o1} &= xMw_g + (1-x)Mw_o \\
SG_{o1} &= \frac{xMw_gN_{oil} + (1-x)Mw_oN_{oil}}{\frac{xMw_gN_{oil}}{SG_g} + \frac{(1-x)Mw_oN_{oil}}{SG_o}}
\end{aligned} \tag{A.10}$$

where N_{oil} is the number of liquid moles in the oil phase; F_o is the molar oil component in the oil discharge flow $F_{o_{out}}$; $\{Mw_g, Mw_o\}$ are the gas and oil molecular weights; and $\{SG_g, SG_o\}$ are the gas and oil specific gravities.

A.3.3 The gas phase

Given the ideal gas law assumption, the gas phase of the separator is modeled by taking the material balance. We can estimate the gas pressure P by applying the ideal gas law, as described by equations (A.11):

$$\begin{aligned}
\frac{dN_{gas}}{dt} &= F_{g1} - F_{g_{out}} \\
V_{oil} &= \frac{Mw_{o1}N_{oil}}{62.43SG_{o1}} \\
V_{gas} &= V_{sep} - V_{wat} - V_{oil} \\
P &= \frac{N_{gas}RT}{V_{gas}}
\end{aligned} \tag{A.11}$$

where N_{gas} is the number of gas moles in the gas phase; $F_{g_{out}}$ is the gas molar outflow from the separator; $\{V_{oil}, V_{gas}, V_{sep}\}$ are the volumes of the oil phase, gas phase, and separator respectively; R is the universal gas constant; and T is the

absolute separator temperature.

A.4 Separator model validation

Having obtained the dynamic model of the three-phase gravity separator, we design a simulation model which emulates an oil production facility to validate the model behavior under several scenarios. The simulation model basically consists of three processes, as depicted in figure A.6. The first is a two-phase separator in which hydrocarbon fluids from oil wells are separated into two phases (gas, oil + water) to remove as much light hydrocarbon gases as possible. The separator is 15 ft long and has a diameter of 5 ft. The two-phase separator model was developed based on the models of the oil and gas phases of the three-phase separator. That is, we have modeled only the thermodynamic phenomenon of gas flashing out the liquid phase. The liquid produced is then pumped to the three-phase separator (i.e., the second process), where water and solids are separated from oil. The oil produced is then pumped out and sold to refineries and petrochemical plants if it meets the required specifications. The three-phase separator has length of 8.6 ft and a diameter of 4.8 ft. Flashed light and medium gases from the separation processes are sent to a gas scrubber where medium hydrocarbon and other liquid remnants are separated from gas and sent back for further treatment. Produced gas is then compressed by a compressor (i.e., the third process) and pumped out for sales. The third process model was not included in the simulation model for the sake of simplicity.

The two separation processes in the simulation model are controlled to maintain the operating point at its nominal value, and to minimize the effect of disturbances on the produced oil quality. As shown in figure A.6, the first separation process is controlled by two PI controller loops. In the first loop, the liquid level is maintained by manipulating the liquid outflow valve. The second loop controls the pressure inside the two-phase separator by manipulating the amount of gas discharge. The

second separation process has three PI controller loops. An interface level PI controller maintains the height of the oil/water interface by manipulating the water dump valve, while the oil level is controlled by the second PI controller through the oil discharge valve. The vessel pressure is maintained constant by the third PI loop.

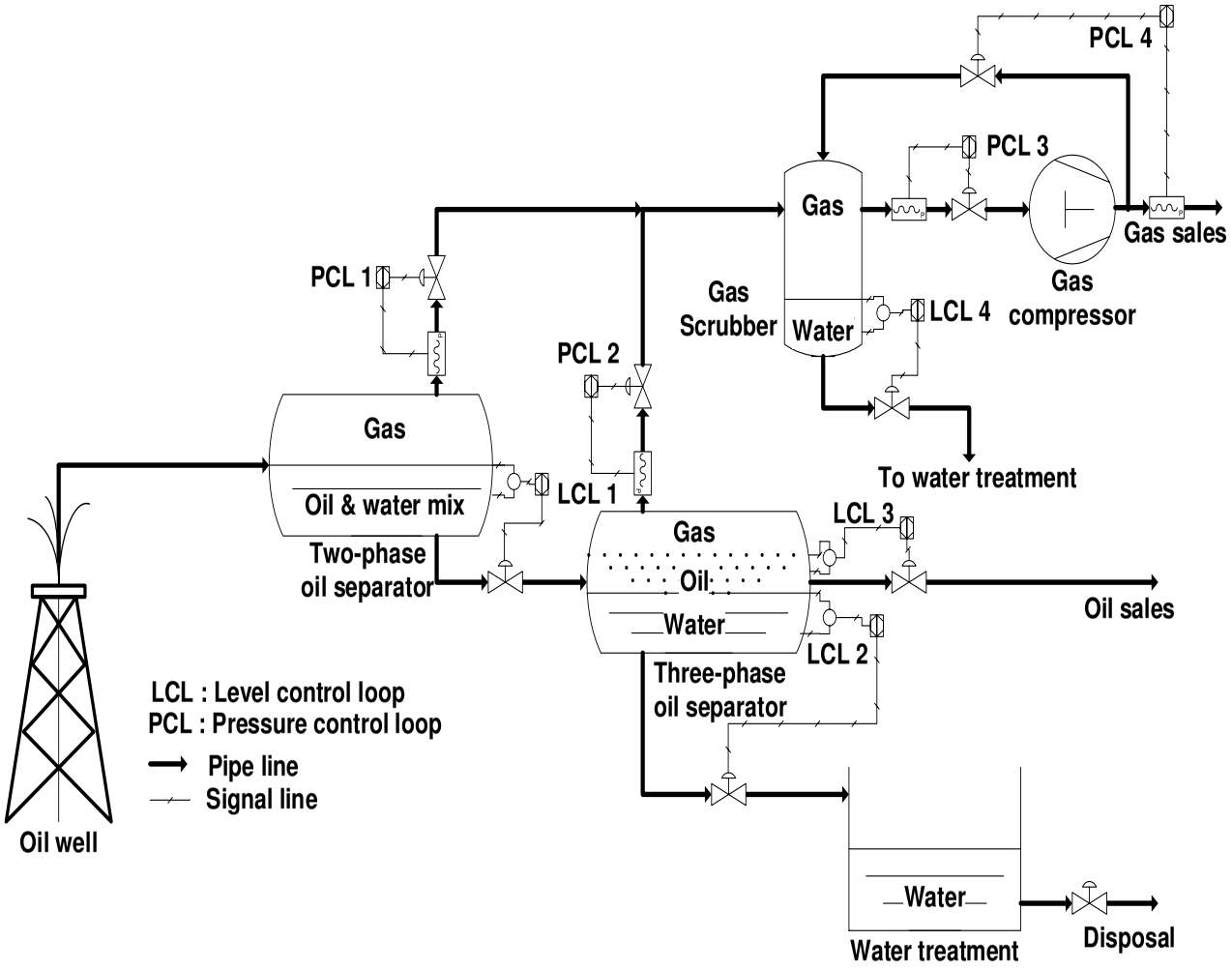

Figure A.6: The oil production facility schematic diagram

A.5 Simulation results

The two-phase separator process operates at a liquid phase volume of 146 ft^3 and working pressure of 625 PSI. In contrast, the three-phase separator operates at water phase volume of 77.5 ft^3, oil phase volume of 46.5 ft^3, and working pressure of 200 PSI. The working temperatures of the two separation processes are 100 ^{o}F. The facility processes hydrocarbon streams of 25.23 moles/s from oil wells under pressure of 1900 PSI. The incoming stream has mole fractions of 22.61% gas, 7.79% oil, and 69.6% water. In order to demonstrate the dynamic behavior of the separators, the oil content of the incoming stream has been increased linearly by 2 moles/s between the time instants $t_1 = 150$ s and $t_2 = 250$ s of the simulation time. Figure A.7 portrays this change in the incoming stream flow and in its molar composition; the oil mole fraction increased while the water and gas mole fractions decreased.

The ramp increase in the oil component of the incoming stream caused the liquid volume and gas pressure in the two-phase separator to peak to 167 ft^3 and 630 PSI respectively, as shown in figure A.8. The two PI control loops of the two-phase separator intervened to correct such operating point errors by manipulating the liquid and gas outflows. This operating point disturbance took approximately 300 s to be totally rejected by the separator control system. Figure A.8 also reveals the difference between the dynamics of the two phases of the separator. The liquid phase has slower dynamics than the gas phase dynamics, i.e., the pressure changes faster than the liquid volume. It is interesting to notice that the liquid molar outflow increased by 2 moles/s, which is the same applied change in the incoming stream. This reflects on the quality of the liquid produced in terms of its specific gravity, which increased from 31.7 ^{o}API to 35.3 ^{o}API. The quality change can be verified by plotting the molar composition of the liquids produced, as shown in figure A.9. The oil mole fraction of the produced liquid increased, while the mole fractions of dissolved gas and water decreased.

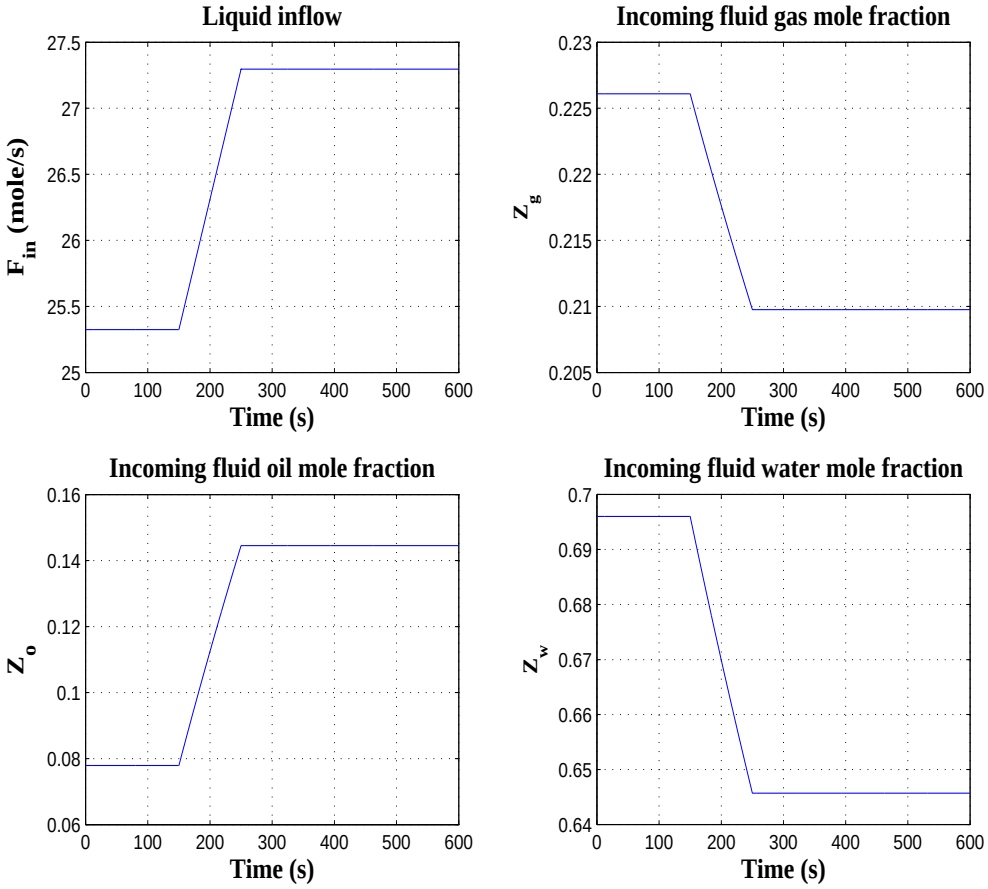

Figure A.7: Incoming hydrocarbon fluid and its molar composition

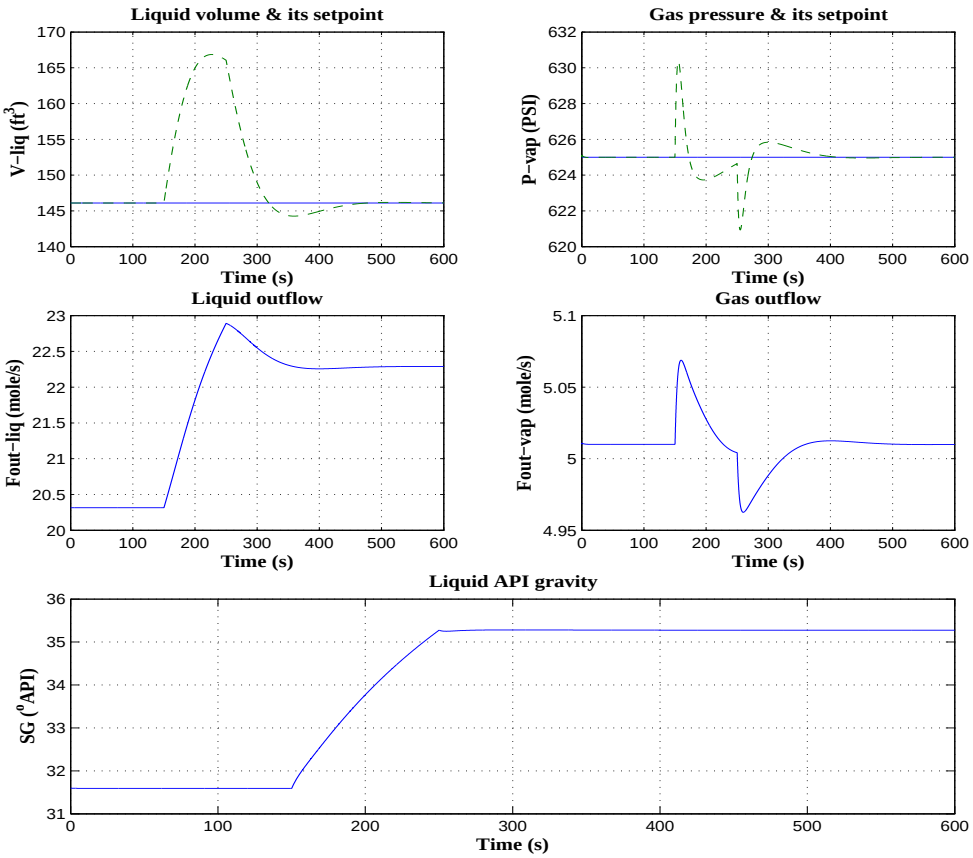

Figure A.8: Two-phase separator process variables change during the incoming stream upset

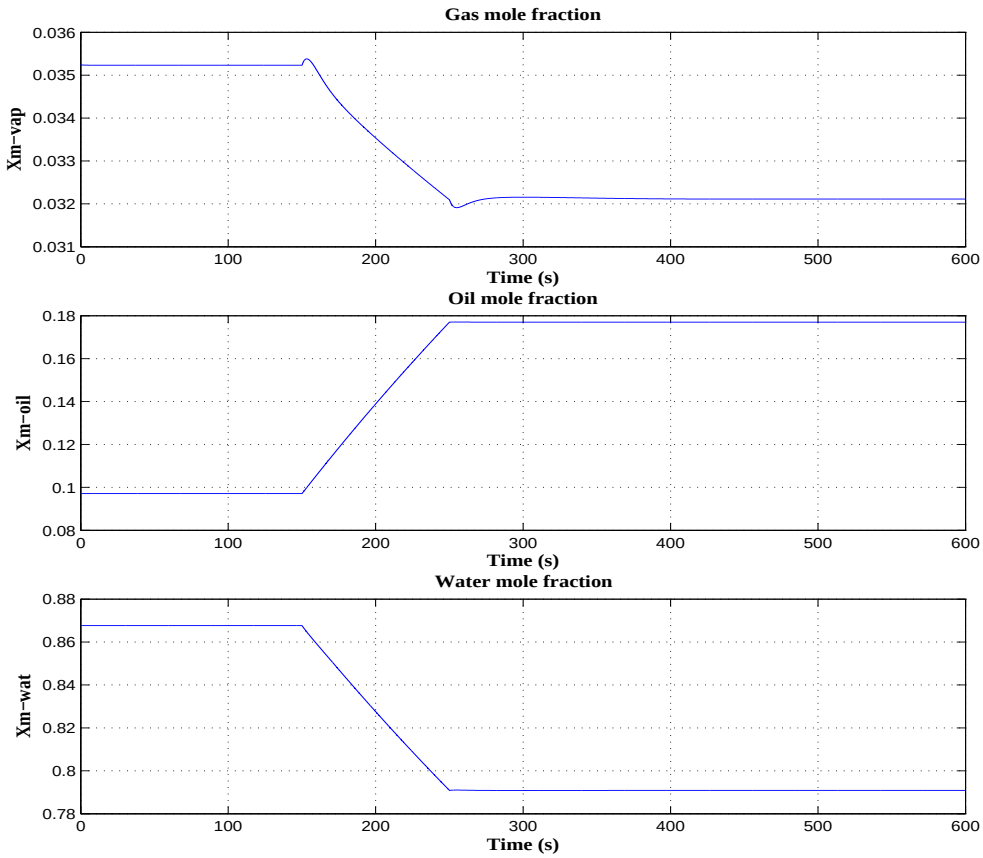

Figure A.9: Two-phase separator liquid discharge molar composition

Although the incoming stream upset was rejected and corrected in the two-phase separator, the resulting change in the quantity and quality of the produced liquid transmitted the upset to the three-phase separator and other downstream processes. As portrayed in figure A.10, the upsets in the separator process variables did not have much impact, as the three PI control loops corrected such an upset. Given the difference between the three phases' dynamics (i.e., the fast gas phase dynamics compared to the two liquid phases), the upsets are rejected in approximately 300 seconds. However, two main events should be noticed; the first event is the slight increase in the water discharge molar flow. This can be attributed to inefficiency in the gravity separation hydrodynamics, which implies that some oil could not be separated and was discharged with water. We can verify this event by plotting the volumetric composition of the dumped water ("1^{st} phase"), as shown by the top plots in figure A.11. The dumped water volumetric composition reveals that some amounts of unseparated oil has been lost.

Although the volume loss of oil is slight (around 1.6%), it represents a major economic loss of approximately $\$\,50$ million per year at current prices. On the other hand, the separator did compensate for the incoming fluid upset by increasing the produced oil outflow, as illustrated in figure A.10. The second interesting event is the decrease in the flashed gas amounts (i.e., gas outflow) due to the quality change in the incoming fluid stream. This can be verified by plotting the molar composition of the produced oil, as shown in the bottom plots of figure A.11. While the oil mole fraction in the produced oil increased, the dissolved gas mole fraction decreased. This simulation study demonstrated the sophistication of the three-phase separator model, in spite of its simplicity. Not only did the model address the quantity dynamics of separator process variables, but the quality of the produced oil and water also.

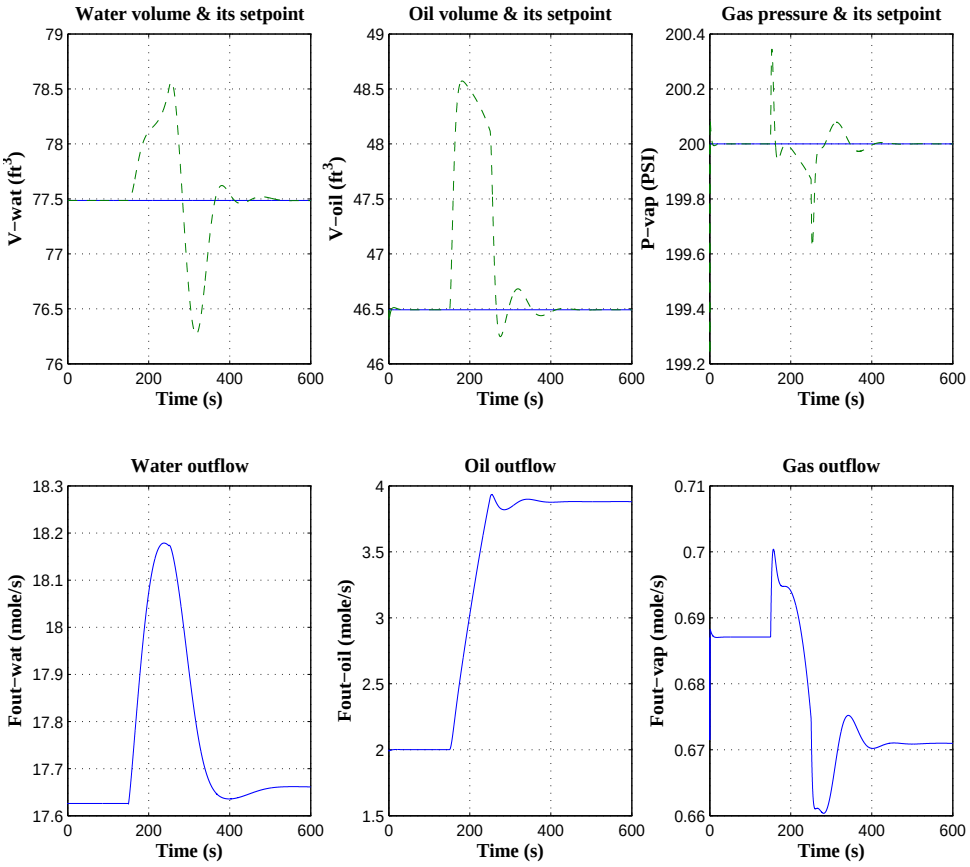

Figure A.10: Three-phase separator process variables change during the incoming stream upset

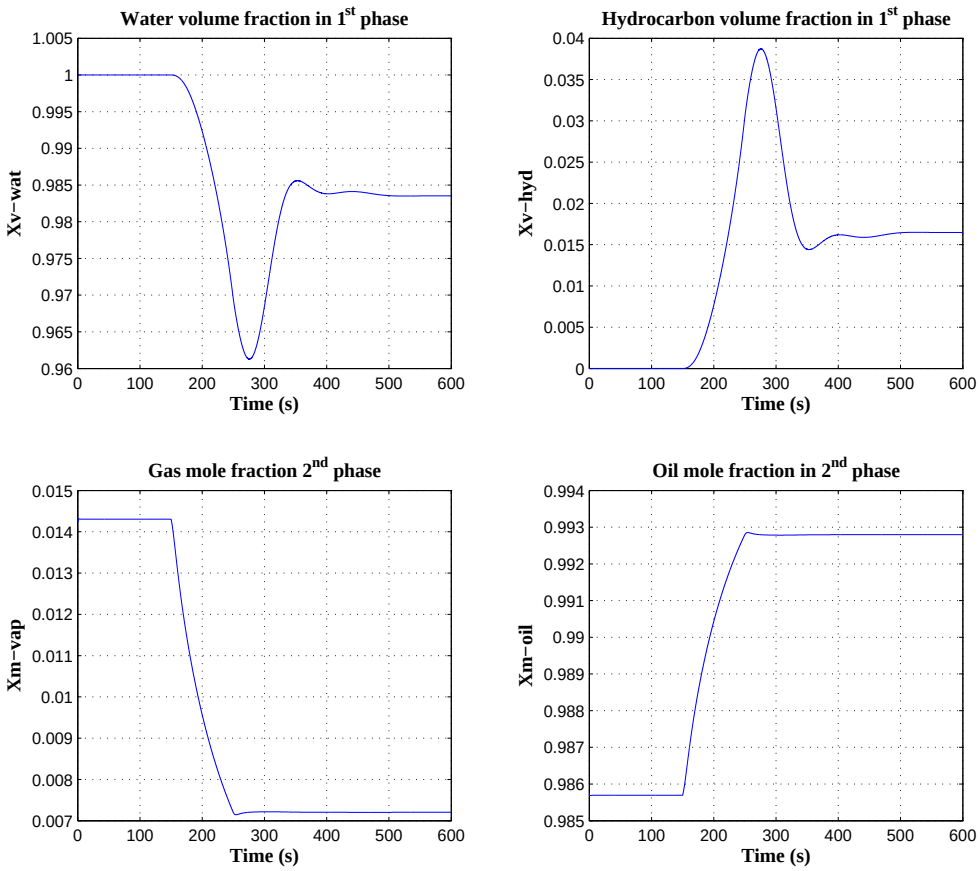

Figure A.11: Three-phase separator produced water and oil compositions

A.6 Conclusions

A dynamic mathematical model was developed for an oil production facility. The focus of this study was on the three-phase separator, where each phase's dynamics were modeled. The hydrodynamics of liquid-liquid separation were modeled based on the API design criteria, which were extended to address the process dynamics in addition to its statics. The oil and gas phases' dynamic behaviors were modeled assuming vapor-liquid phase thermodynamic equilibrium at the oil surface.

An oil production facility simulation was designed based on such a model, in order to test and validate the developed mathematical model. The simulation model consisted of a two-phase separator followed by a three-phase model. The separation processes were controlled by PI control loops to maintain the operating point at its nominal value. An upset in the oil component of the incoming oil-well stream was introduced to analyze its effect of the different process variables and produced oil quality. The simulation results proved the sophistication of the model in spite of its simplicity. Furthermore, this study demonstrated the challenging task of modeling and controlling oil and gas production facilities, and that more work has to be done to develop higher fidelity models.

Vita

Candidate's full name: Atalla F. Sayda
University attended:

University of New Brunswick M.Sc.EE. 2002
University of Damascus B.Sc.EE. 1996

Publications:

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

2. **A. F. Sayda** and J. H. Taylor, "Toward a practical multi agent system for integrated control and asset management of petroleum production facilities", accepted for *the IEEE International Symposium on Intelligent Control*, Singapore, 1-3 October 2007.

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

4. **A. F. Sayda** and J. H. Taylor, "An intelligent multi agent system for integrated control and asset management of petroleum production facilities", *Proc. of the Flexible Automation and Intelligent Manufacturing (FAIM)*, Philadelphia, 18-20 June 2007.

5. **A. F. Sayda**, "Approach to criticality on initial core startup of CANDU reactors", *Proc. 28th Annual Conference of the Canadian Nuclear Society*, Saint John, New Brunswick, Canada, 3-6 June 2007.

6. **A. F. Sayda** and J. H. Taylor, "An implementation plan for integrated control and asset management of petroleum production facilities", *Proc. IEEE International Symposium on Intelligent Control*, Munich, Germany, 4-6 October 2006.

7. **A. F. Sayda**, "Model identification and robust h-infinity controller design of a motor-synchronous generator group", *Proc. IEEE International Conference on Control Applications*, Munich, Germany, 4-6 October 2006.

8. **A. F. Sayda**, "Modeling and control of three-phase gravity separators in oil production facilities", *Rocky Mountain/Mid-Continent/Eastern Region Student Paper/Presentation Contest*, Society of Petroleum Engineers (SPE), University of Alberta, Edmonton, Alberta, 30 April - 1 May 2006.

9. J. H. Taylor and **A. F. Sayda**, "Intelligent information, monitoring, and control technology for industrial process applications", *Proc. FAIM 2005 (Flexible Automation and Intelligent Manufacturing)*, Bilbao, Spain, July 18-20, 2005.

10. J. H. Taylor and **A. F. Sayda**, "An intelligent architecture for integrated control and asset management for industrial processes", *Proc. IEEE International Symposium on Intelligent Control*, Limassol, Cyprus, 27-29 June 2005.

11. **A. F. Sayda** and J. H. Taylor, "Model predictive control of a mechanical pulp bleaching process", *4th IFAC Workshop on Time Delay Systems (TDS)*, INRIA, Rocquencourt, France, September 8-10, 2003.

Conference Presentations:

1. **A. F. Sayda** and J. H. Taylor, "Toward a practical multi agent system for integrated control and asset management of petroleum production facilities", *IEEE International Symposium on Intelligent Control*, Singapore, 1-3 October 2007.

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

3. **A. F. Sayda**, "Approach to criticality on initial core startup of CANDU reactors", *28th Annual Conference of the Canadian Nuclear Society*, Saint John, New Brunswick, Canada, 3-6 June 2007.

4. **A. F. Sayda** and J. H. Taylor, "An implementation plan for integrated control and asset management of petroleum production facilities", *IEEE International Symposium on Intelligent Control*, Munich, Germany, 4-6 October 2006.

5. **A. F. Sayda**, "Model identification and robust h-infinity controller design of a motor-synchronous generator group", *IEEE International Conference on Control Applications*, Munich, Germany, 4-6 October 2006.

6. **A. F. Sayda**, "Modeling and control of three-phase gravity separators in oil production facilities", *Rocky Mountain/Mid-Continent/Eastern Region Student Paper/Presentation Contest*, Society of Petroleum Engineers (SPE), University of Alberta, Edmonton, Alberta, 30 April - 1 May 2006.

7. J. H. Taylor and **A. F. Sayda**, "An intelligent architecture for integrated control and asset management for industrial processes", *IEEE International Symposium on Intelligent Control*, Limassol, Cyprus, 27-29 June 2005.

8. **A. F. Sayda** and J. H. Taylor, "Model predictive control of a mechanical pulp bleaching process", *4th IFAC Workshop on Time Delay Systems (TDS)*, INRIA, Rocquencourt, France, September 8-10, 2003.