European Symposium on Computer Aided Process Engineering – 15
L. Puigjaner and A. Espuña (Editors)

An Agent-oriented Approach to Integrated Process Operations in Chemical Plants

Magid Nikraz and Parisa A. Bahri*

School of Engineering Science and A. J. Parker Centre for Hydrometallurgy
Murdoch University, Dixon Road, Rockingham WA 6168, Australia

Abstract

Integration of process operations involves the coordinated management of operational tasks in a process plant. In the context of chemical process plants, these tasks can be categorised as data acquisition, regulatory control, data reconciliation, process monitoring, fault diagnosis, supervisory control, scheduling and planning. While each of these tasks is responsible for a particular function, they are also dependent on each other and thus cannot be treated in isolation – this is why integration is necessary.

The focus will be on the use of agent-oriented techniques to achieve integration. Limitations of previous integration techniques in the context of chemical plants will be outlined and this will elicit the need for a more powerful software engineering paradigm utilising software agents. The main barriers to achieving integration will then be discussed and how an agent-oriented approach represents a logical solution to these problems. Finally, a pilot plant application of the proposed technique will be presented.

Keywords: integration, agent, chemical process plant

1. Introduction

Agent-oriented software engineering techniques present a possible solution to the problem of integration of process operations. The tasks which come under process operations in the case of chemical plants can be loosely classified as data acquisition, regulatory control, process monitoring, data reconciliation, fault diagnosis, supervisory control, scheduling and planning. Integration in this context refers to the process of bringing these separate tasks together under a coordinated framework (Venkatasubramanian, 1994). Though integration is highly desirable, its achievement is made more difficult by the complexity of today's chemical process plants and the different approaches to each of the tasks.

Agent-oriented techniques are suited to the problem of integration primarily because they provide a uniform means of communication between the modules representing the individual operational tasks. Furthermore, agents can encapsulate legacy systems written at different times and bring them into the integrated framework.

The focus of this study is on the software aspects of integration and the relationship between the individual tasks. A brief review is presented of previous integration

* Author to whom correspondence should be addressed: P.Bahri@murdoch.edu.au

techniques and their limitations. Then, some barriers to achieving integration are outlined and the role of agent-oriented techniques in providing a solution is discussed. Finally, a pilot plant application of the proposed technique is presented.

2. Review of Previous Integration Techniques

The major works in the area of integration of process operations can be classified as:

- Functional hierarchy (Sardis, 1983)
- Blackboard architecture (Fjellheim et. al., 1994)
- A framework for integrated process supervision (Rengaswamy, 1995)
- Coordinated Knowledge Management Method (Power, 2004)

2.1 Functional hierarchy

The idea is based on the hierarchy shown in Figure 1. There are three levels: execution level, coordination level and organisation level. The low levels require more precision and less intelligence, while for higher levels, this requirement is reversed. Layers are added on top of each other one by one, with each layer being tested and accepted before another more complex layer is introduced on top of it. The new layer then acts on the lower layer, which modifies the status of the objects associated with them.

A very rigidly structured organisation with many levels functions best when environmental conditions are relatively stable. Unstable environmental conditions, which are prevalent in the chemical processing industry, require a flexible organisation that can adapt quickly; hence, a rigid structure is no longer considered appropriate.

2.2 Blackboard architecture

The idea involves a group of generic problem solvers or experts which look at the same blackboard recording individual states of the ongoing problem solving process. Each expert takes appropriate actions based on the information presented on the blackboard. A key feature of this structure is that the problem solving states are made available in the form of global data structures, while maintaining the isolation of each of the modules. The blackboard model consists of three major components: 1. knowledge sources, which are independent but complementary subsets of the knowledge about the process; 2. blackboard data structure, where all the knowledge sources have exclusive access for retrieval and storage modification of information; 3. control mechanism, which consists of the knowledge sources responding to the changes of the blackboard.

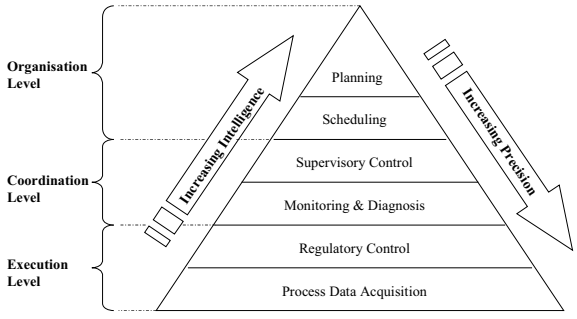

Figure 1. Functional hierarchy applied to chemical processing plants

The blackboard model only outlines the organisation principle and does not specify how the system is to be realised as a computational entity. Application of the blackboard framework often requires extensions to the framework. Also, when applying the blackboard framework, one must address the problem of maintaining data consistency in the blackboard by controlling asynchronous references to shared data.

2.3 A framework for integrated process supervision

This framework, shown in Figure 2, attempts to integrate the lower and mid-level process operational tasks for continuous operations.

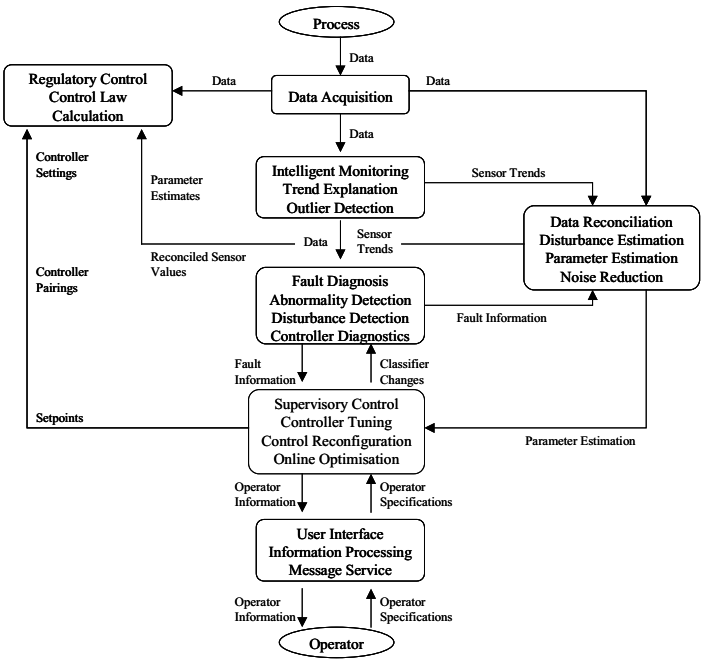

Figure 2. Framework for integrated process supervision

The framework is useful for considering the information flow, however, no mention is made as to how coordination of tasks is achieved in the references cited nor if the information has been applied.

2.4 Coordinated Knowledge Management Method

This method, shown in Figure 3, allows tasks to communicate directly with the coordination mechanism dispensing with the requirement of an external control mechanism. Hierarchy is present but it is not as rigidly structured as in the functional hierarchy. As in the case of the blackboard framework, tasks represent knowledge sources and act autonomously. However, the blackboard is no longer present; instead, a Petri-net is used to: 1. coordinate tasks, 2. monitor the system, 3. activate the knowledge sources (tasks), 4. request data to be updated in the data structures and 5. receive notice when the task is completed. Visualisation of the state of the system is achieved through the moving tokens in the Petri-net. Each individual module contains information (including rules, procedures, Petri-nets, optimisation and neural networks), which enables each module to operate autonomously.

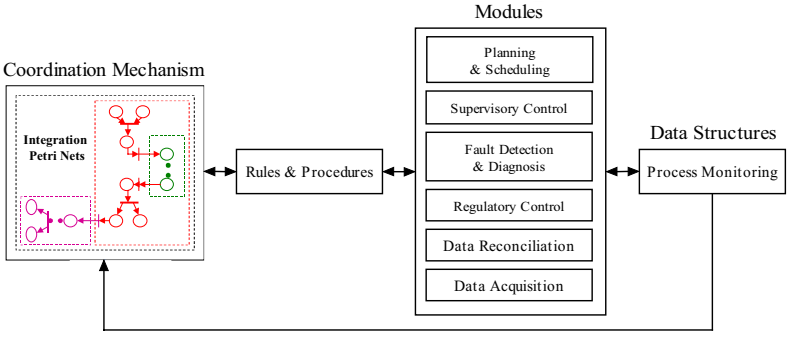

Figure 3. Coordinated Knowledge Management Method

Module interaction, organised through the structure of the hierarchical timed place Petri net (low-level net), is adequate for a small plant. However, as plant size increases, not only are there module interactions, but also plant section interactions; so, the integration Petri-nets must be extended to coordinate both module and section interaction. This extension will mean that the Petri-net model will become very complex and difficult to manage.

3. An Agent-oriented Solution

The paradigm of agent-oriented programming was introduced by Shoham in 1993 as a specialisation of object-oriented programming (Shoham, 1993). The meaning of the term agent is fuzzy and a subject of ongoing debate. However, one popular characterisation is (Jennings and Bussmann, 2003):

> *"an agent is an encapsulated computer system that is situated in some environment and is capable of flexible, autonomous action in that environment in order to meet its design objectives"*

In addition, most problems require or involve multiple agents which interact with each other in a common environment; such an environment is referred to as a multi-agent system. For an introduction to agent-oriented software engineering, the reader is referred to Wooldridge (2002).

A major problem when attempting to integrate a system is heterogeneity. This refers to the fact that some software may be pre-existing. Furthermore, the pre-existing (non-agent) software may incorporate different solution techniques for different tasks, each of which may have been developed at a different time by different people. One of the benefits of using agents is that it permits the use of pre-existing software through encapsulation using a wrapping mechanism. The wrapping software looks like another agent from the outside, while serving as an interface to the legacy components at the same time. As pointed out by Jennings and Bussmann (2003):

> *"this ability to wrap legacy systems means that agents may initially be used as an integration technology"*

Furthermore, this capability permits a more flexible system structure and ultimately results in time and cost savings since new software is not required to be developed. Flexibility is a major advantage of the agent oriented approach which is lacking in the previous techniques presented in Section 2.

Another requirement for integrating systems is that the software components must be able to interoperate with each other by exchanging information and services. An agent-oriented solution permits communication between the software components using a universal communication language. The framework shown in Figure 3 highlights the vital need for communication between the individual software components. Some may argue that objects also have means of communication through message passing. This is true, but the advantage of an agent-oriented approach is the use of a common language with agent-independent semantics (Genesereth and Ketchpel, 1994). There are also several other important differences between agents and objects, which have been indicated in the literature (Jennings and Bussmann, 2003).
The notion of permitting heterogeneity in the system and a universal communication language for information exchange make an agent-oriented approach effective from the perspective of integration. Furthermore, this approach also addresses the shortcomings of previous techniques discussed in Section 2 by permitting flexibility and an effective means of information exchange for all components.

4. Pilot Plant Application

The pilot plant representing the Bayer Process at Murdoch University is being used as a base for testing the agent-oriented approach to integration. The necessary modules representing the tasks have been developed: data reconciliation, process monitoring, fault detection and diagnosis and supervisory control. Some modules are pre-existing (via Honeywell SCAN 3000): data acquisition and regulatory control. Furthermore, the scheduling and planning have not been implemented because they are application specific and implemented over a timescale of months or years.
The proposed model for integration is shown in Figure 4. The system consists of a group of pre-existing and custom-built heterogeneous agents. The agents communicate via an agent communication language (ACL) standardised by FIPA (FIPA, 2004). The use of the wrapper agent can be observed, which serves as an interface between the pre-existing modules and the other agents. The agents are situated over several pilot plant computers. The user agent acts as a means of integrated information exchange between the system and the user.

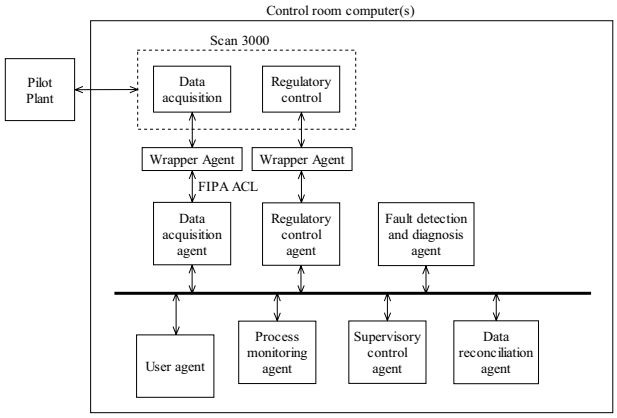

Figure 4. Proposed multi-agent system architecture for integration

Each of the modules in Figure 4 has a known function. Based on this, the dependencies between the modules can be defined through a brainstorming session. The dependencies are transformed into cooperative actions or interactions between the agents. Once the interactions between the agents are known, a suitable interaction protocol for each interaction, possibly one defined by FIFA is selected depending on the nature of interaction between the agents. For example, if the fault detection and diagnosis agent needs to know whether the pilot plant is at steady-state, it will need to interact with the monitoring agent which is responsible for determining whether the plant is at steady-state. The fault detection and diagnosis can communicate with the monitoring agent using a FIPA-Query interaction protocol, which returns a True of False value depending on whether the plant is at steady state. It must be noted that the only means of agents communicating is via ACL and an agents methods can not be accessed directly by other agents contrary to objects where this requirement does not exist. Once the interactions and the interaction protocols have been defined and ontology formed, the system can be implemented using a FIPA compliant agent framework such as the Java Agent Development Framework (Bellifemine et. al., 2001).

5. Conclusions

The need for an integrated framework for coordination of tasks in a processing plant stems from the growing complexity of current systems, as well as from the traditional expense, time constraints and limited availability of human expertise. In light of the many benefits, very little attention has been devoted to finding a solution to the problem of integration with respect to chemical processing plants. As a result, many plants today operate without realising the benefits of integration via software inter-communication. It has been suggested that an agent-oriented solution presents an effective approach to solving this problem by providing a powerful mindset to the engineer and allowing the system to expand relatively easily and in a modular fashion.

References

Bellifemine, F., A. Poggi and G. Rimassa, 2001, JADE: a FIPA2000 compliant agent development environment, Agents 2001, 216-217.

FIPA, 2004, Foundation for Intelligent Physical Agents, http://www.fipa.org.

Fjellheim, R.A., T.B. Pettersen, B. Christoffersen and A. Nicholls, 1994, Application Methodology for REAKT Systems, proceedings of the IFAC Artificial Intelligence in Real Time Control, Valencia, Spain, 325 – 332.

Genesereth, M.R. and S.P. Ketchpel, 1994, Software Agents, Comm. of the ACM, 37(7), 48 – 53.

Jennings, N.R. and S. Bussmann, 2003, Agent Based Control Systems, IEEE Control Systems Magazine, 23(3), 61 – 64.

Power, Y., 2004, The Development of an Integrated Process Operation Management System, Doctor of Philosophy Thesis, Murdoch University.

Rengaswamy, R., 1995, A Framework for Integrated Process Monitoring, Diagnosis and Supervisory Control, Doctor of Philosophy Thesis, Purdue University.

Sardis, G.N., 1983, Intelligent Robotic Control, IEEE Trans. on Auto. Control, 28, 547 – 557.

Shoham, Y., 1993, Agent-oriented programming, Artificial Intelligence, 60(1), 51-92.

Venkatasubramanian, V., 1994, Towards Integrated Process Supervision: Current Status and Future Directions, proceedings of the IFAC Computer Software Structures Integrating AI/KBS Systems in Process Control, Lund, Sweden, 1 – 13.

Wooldridge, M., 2002, An Introduction to Multiagent systems, John Wiley and Sons Ltd.