Manufacturing Information Coordination and System Integration By A Multi-Agent Framework

Riyaz Sikora
Michael J. Shaw

Artificial Intelligence group
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
Urbana, IL 61801.

Abstract

Modeling typical enterprise information systems requires the modeling of multiple agents with different functionalities. This paper describes a methodology that provides the representational formalism, coordination mechanisms, and control schemes necessary for integrating heterogeneous units of distributed information systems while meeting such performance criteria as overall effectiveness, efficiency, responsiveness, and robustness.

An important component of the methodology is the coordination among agents. The framework is applied to the development of a manufacturing information system for managing the production processes for making printed circuit boards. Performance results confirm that the multi-agent framework for system integration is important to support complex business processes that involve multiple steps of activities processed by a group of agents across a variety of functionalities. The framework also provides an unified model for integrating software and decision modules in complex information systems.

1. Introduction

There are several objectives of systems integration, such as the establishment of good communication and coordination procedures; promoting collaboration among a group of networked participants, sharing of information among different entities across an information system; maintaining a variety of functional modules to make an information system flexible and adaptive; efficiently manage the coordination between different information systems in an enterprise; and consolidating various business functions in information systems to make processes more efficient. One of the crucial issues in system integration is the identification of coordination mechanisms, the representation of different roles and functionalities, and the performance measures. The are several dimensions to the systems integration problem, each with its own set of objectives (Sikora&Shaw, 1994):

• Integration of heterogeneous information systems, data bases, or application software to facilitate the data flow between them and to make the overall system robust and efficient.

• Integration of different physical stages in business processes to improve the internal performance metrics. For example, information systems (IS) for the integration of the manufacturing and the assembly stages on an manufacturing shop floor to improve the coordination between them and to better react to any emerging demand from customer orders or from other product groups.

• Integration of different functional units in an organization to improve the external performance metrics. For example, information system for the integration of the marketing, product design, and production functions of an organization to improve the product development life cycle.

• Integration of subsystems into a well-coordinated, networked system. For example, information system for the integration of a network of suppliers, producers, and distributors to reduce the inventory buildup in the network of suppliers.

In order to effectively deal with the problem of system integration in all of the different dimensions outlined above we need a general framework that can be used as a formalism to represent and analyze the structure of the enterprise activities involved and their interactions. The common feature of system integration is that it requires coordination among physically or logically distinct and sometimes complex processes, component units, and subsystems.

In this paper we will present a multi-agent framework of coordination. We will show how our framework can be recursively applied to analyze a multi-agent system at finer levels of granularity. In other words, each agent in a multi-agent system can be further

decomposed into sub-agents to analyze its internal working and this process can be continued recursively until the primitive elements of the system are reached.

Finally, we discuss the application of the above framework of coordination to the problem of system integration in general and present a specific application of the above framework to a real-world manufacturing problem in a printed circuit boards facility. In conclusions we discuss some of the other applications in which the above framework can be applied and present directions for future research.

2. A Framework of Information Systems Coordination

The essential requirement for the need of coordination mechanisms in information systems is the presence of a collection of agents[1] working together to solve the same problem simultaneously or working asynchronously on different parts of the same problem (Crowston[1991]; Shaw and Fox [1992]). One of the main reasons the study of coordination among different problem solvers has become more important in recent times is because the advances in hardware technology for processor construction and interprocessor communication have made it possible to connect large numbers of sophisticated, yet inexpensive, processing units that execute asynchronously. This makes it possible to build systems for tackling complex problems which have not been possible before. Malone and Crowston (1991) present an interdisciplinary theory of coordination. Below we give the theoretical foundations that will help in analyzing the working of any multi-agent system.

In analyzing a multi-agent information system we define the working of the system based on a *performance measure*, π (formally defined later). The *goal* of the system is to improve the performance measure. The agents have their individual performance measures, π_i. Note that this characterization is general enough to include what is referred to as Open Systems in Hewitt(1991), where there are no specific goals; rather the system is composed of independently developed parts (or agents) in continuous evolution. Each agent has its internal behavior model which defines the role of the agent and how it would react to the changing environment or to messages received. The goal of this interaction among the agents is to improve the performance of the system as a whole. Note, however, that any multi-agent system where the agents have specific *sub-goals* can still be captured in our framework. The performance measure of the system as a whole can be defined in terms of these sub-goals.

[1]By agents we mean software modules, decision units, or, in general, information subsystems.

We next present an analytical model of multi-agent systems that is motivated by the work done on modelling of distributed systems (Chandy and Misra [1986], Halpern and Fagin [1989], Milner [1980]) and Actor Systems (Hewitt[1977]).

Any multi-agent information system can be analyzed as consisting of individual *agents* $AG_1, AG_2, \ldots, AG_n$ and an *environment*, E. We define the interdependencies among the agents based on the *interaction variables*. The interaction variable z_{ij} gives the information for which agent AG_j is dependent on agent AG_i. The interactions among the agents then consists of sending and receiving the interaction variables. We define z_i, the *dependency vector* for agent AG_i, as a vector given by

$$z_i = (z_{1i}, z_{2i}, \ldots, z_{ni})$$

Let Z_i be the set of all possible dependency vectors for agent AG_i. We define the *interdependency structure*, $\mathcal{Z}$, as the space given by

$$\mathcal{Z} \subseteq Z_e \times Z_1 \times \ldots \times Z_n$$

We characterize the working of such a multi-agent system in terms of the states of the agents and the activities that they can perform. We denote s_e as the state of the environment and s_i as the *local state* of the agent AG_i. The local states of the agents also include the information about the interaction variables. The *global state* of the system is then defined as the (n+1)-tuple $(s_e, s_1, \ldots, s_n)$. Let S_e be the set of all possible states of the environment and S_i be the set of all possible local states of the agent AG_i. We define the *global structure* $\mathcal{G}$, given by

$$\mathcal{G} \subseteq S_e \times S_1 \times \ldots \times S_n,$$

as the set of all possible global states. The above relationship is that of a subset to allow for the fact that some of the global states might not be possible. Note that the interdependency structure is part of the global structure.

We define the system performance measure, π, as a function from the global states to a measurement in real numbers, i.e.,

$$\pi : \mathcal{G} \longrightarrow \mathcal{R}$$

We say that the performance measure of the system improves as it moves from global state g_1 to global state g_2 if $\pi(g_2) > \pi(g_1)$.

We define a_i to be the *activity* performed by the agent AG_i, and a_e as the activity performed by the environment E. We allow even the environment to perform activities. For example, messages received by the agents from the outside world can be modelled as activities performed by the environment. Since our main motivation in developing this framework is to model the working of a multi-agent information system, interaction and coordination through communication activities such as message passing will be the most important control structure in our framework. We define the basic activity to consist of

either sending messages, receiving messages, or internal computation, where messages may be control signals, data, or coordination commands. In this model we assume that the internal computations of the agents are not visible to the other agents and hence do not affect them; they only change the local state of the agent performing the computation. This is similar to the concept of "arm's length relationship" defined in Open Systems (Hewitt[1991]). A variety of coordination mechanisms are discussed in Ching et al. (1992) and Shaw & Fox (1993).

Since the agents can work asynchronously and can perform their activities simultaneously, we define the notion of a joint activity as the tuple $(a_e, a_1, \ldots, a_n)$. We define a function τ that associates a *global state transformer* with every joint activity. In other words, $\tau(a_e, a_1, \ldots, a_n)(g)$ is the global state that results when the joint activity $(a_e, a_1, \ldots, a_n)$ is performed on the global state g. More succinctly, $\tau(a_e, a_1, \ldots, a_n)$ is a function from the global states to global states

$$\tau(a_e, a_1, \ldots, a_n) : \mathcal{G} \longrightarrow \mathcal{G}$$

Let A_e be the set of all possible activities that can be performed by the environment, and let A_i be the set of possible activities that can be performed by the agent AG_i. Let $\mathcal{A}$, defined by

$$\mathcal{A} \subseteq A_e \times A_1 \times \ldots \times A_n,$$

be the set of joint activities possible. As before, $\mathcal{A}$ is defined as the subset of all possible joint activities to account for the fact that certain joint activities might not be possible.

Figure 1 shows the conceptual model of an agent. Each agent is modelled as consisting of three important components. A *knowledge base* $\mathcal{KB}_i$,, a *control unit* $\mathcal{CU}_i$, and a *functional component* $\mathcal{F}_i$. The knowledge base of agent AG_i in turn consists of a *list of acquaintances*, $\mathcal{Q}_i$, which is a finite collection of agents that it directly knows about together with their skills, and a case history of the past experiences, $\mathcal{H}_i$. The control unit of the agent AG consists of protocols $\mathcal{P}_i$ and a learning module $\mathcal{ML}_i$. The functional component consists of computational procedures and a behavior model.

An agent can communicate only with those who are in its acquaintance list. The list of acquaintances models the *control structure* at the global level. We define the control structure, γ, as an nxn matrix such that $\gamma[i,j] = 1$ if agent AG_i can communicate directly with the agent AG_j, i.e., $AG_j \in \mathcal{Q}_i$. Note that the control structure does not have to be symmetric. In the above case it is possible that agent AG_j does not know about agent AG_i. Agent AG_j can still receive messages from agent AG_i and reply to it without knowing about the sender of the messages. This is meant to capture the common type of communication available in most computer languages, i.e, procedure or sub-routine calling. In procedure

calling, the procedure being called does not know which program is making the call; it can still return the result of its computation to the caller.

The protocols of the agents define how they respond to the changes in their local state resulting from an internal computation or from a receipt of a message. Note, the protocols could be either *strategies* in a decision making context or *algorithms* in a computational context. We define the individual protocols run by the agents as functions from their local states to their activities. If $\mathcal{P}_i$ is the protocol run by the agent AG_i then

$$\mathcal{P}_i : S_i \longrightarrow A_i$$

In other words, each agent decides what to do based on its local state. This is in keeping with the fact that the agents do not have a truly global view of the system and their actions depend only on what they perceive as their immediate environment and on the messages received, i.e., their local state.

We define a *coordination mechanism*, $\mathcal{C}$, as a joint protocol $(\mathcal{P}_e, \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n)$, where $\mathcal{P}_i$ is the protocol run by the agent AG_i, and $\mathcal{P}_e$ is the protocol run by the environment. Note that just as it was useful to view the environment as performing an activity it is also useful to view the environment as running a protocol. We can use the environment's protocol to capture the possibilities that messages might get lost. Input from the outside world can also be modelled as messages from the environment. Thus, a coordination mechanism can be viewed as a function from the global states to the joint activities, i.e.,

$$\mathcal{C} : \mathcal{G} \longrightarrow \mathcal{A}$$

Figure 2 shows different database management systems (DBMS) of a typical company connected on a network and illustrates an example of the application of the above framework for integration. It consists of a main corporate database and departmental databases (for example, engineering and manufacturing). The corporate database might have information about the personnel, marketing data, financial data, etc. The departmental databases might contain information pertaining to their respective areas. Within each department there might be local databases containing information that is localized to a specific operation or component. For example, within manufacturing there might be local databases that contain work assignments, movements of materials, etc., for the respective machines.

Based on our formalism each DBMS can be modelled as an agent that has to interact with the other company and corporate databases for information sharing. The list of acquaintances, Q, of each DBMS would be the list of other DBMSs on the network with which it can interact. As mentioned above, the list of acquaintances models the control

structure of the network of agents. For example, to model restricted access of the corporate DBMS, it might be on the list of acquaintances of the engineering and manufacturing DBMSs but not on the local DBMS within the manufacturing domain.

The knowledge base of each DBMS would consist of the actual data stored and a list of acquaintances. The control unit of each DBMS would consist of protocols for interacting with the other DBMSs. For example, it could be based on a query language like SQL to send queries to or handle queries from other DBMSs. The control unit would also redirect queries it is unable to handle to the appropriate DBMS. The functional component would consist of various computational procedures needed to extract meaningful information form the data base. A truth maitanance mechanism (Huhns &Bridgeland, 1991) or the equivalent would be needed to ensure the consistency of such multi-agent systems.

One of the important advantages of using the above framework is the flexibility it provides in modelling any multi-agent system at different levels of granularity. Each individual agent in a multi-agent system can itself be considered as a multi-agent system and can be analyzed using the same framework. In other words, a system can be broken down and analyzed in successively finer levels of granularity until we reach the *atomic* elements constituting the system.

For instance, an agent AG_i can be modelled as having its own internal environment E_i and sub-agents AG_{ij}, i.e., $AG_i = \{E_i, AG_{i1}, AG_{i2}, \ldots, AG_{im}\}$. The local state, s_i, of the agent can be modelled as a tuple consisting of the local states of its sub-agents and its internal environment, i.e., $s = (s_{ie}, s_{i1}, s_{i2}, \ldots, s_{im})$. An activity, a_i, of the agent can be modelled as the tuple corresponding to the joint activities of its sub-agents, i.e., $a_i = (a_{ie}, a_{i1}, a_{i2}, \ldots, a_{im})$. The set of all activities, A_i, of the agent would then be same as the set of joint activities of its sub-agents, $\mathcal{A}_i$, where

$$\mathcal{A}_i \subseteq A_{ie} \times A_{i1} \times \ldots \times A_{im}$$

Similarly, the protocol, $\mathcal{P}_i$, of the agent can be modelled as the joint protocol of its sub-agents, i.e, $\mathcal{P}_i = (\mathcal{P}_{ie}, \mathcal{P}_{i1}, \mathcal{P}_{i2}, \ldots, \mathcal{P}_{im})$.

Thus, we see that the above framework lends itself easily to analyzing a multi-agent system at multiple levels of granularity *recursively*. For example, at the top level an organization information system can be analyzed as a multi-agent system where the agents correspond to the information systems of different functional departments like finance, marketing, manufacturing, etc. Each of these departmental information systems can in turn be analyzed in terms of its different divisions of tasks and missions depending on the structure and strategies of the organization. For example, the working of the manufacturing department information system can be analyzed in terms of the interactions among the

information systems of its different activities such as design, facility layout, planning, processing, material handling, quality control, etc.

Agent coordination and task allocation have been studied by research in distributed artificial intelligence (Gasser and Huhns [1989]; Durfee and Lesser [1989]; Derfee et al. [1989]; Adler et al. [1992]). A related framework for understanding large-scale systems has been the development of the Actor and Open Systems (Hewitt [1977,1991]). The basic concept underlying these systems is that, rather than viewing a control structure as a sequence of choices made by a single decision maker in a space of choice points, it could instead be seen as a pattern of messages passing among a collection of computational agents called "actors." The main features of the Actor model are (1) the ability for a society of experts or actors to achieve intelligent behavior; (2) a theory of computation that supports fine-grained and naturally concurrent computation; (3) control structures as patters of messages rather than a sequential choice among alternatives; and (4) nonserialized actors, as opposed to serialized actors, having a local state that persists on reinvocation. The Open System model, on the other hand, formalizes the characteristics of information systems composed of independently developed units in continuous evolution. Taking a more system-level viewpoint, the model consider information systems with concurrent and asynchronous processes. Coordination is this type of information systems can be based on such mechanisms as debate and negotiation (Holsapple et al., 1991; Sycara[1989]). Werner (1989) developes a theory of communication for cooperating agents.

3. Application of the Agent-Based Framework to System Integration

The common feature of system integration is that it requires coordination among physically or logically distinct and sometimes complex processes or systems. According to Mize(1991), there have been two traditional approaches of doing system integration in the past. First is the "islands of automation" approach in which realistically sized modules are first build independently and are then linked together into an integrated system. The drawback of this approach is that it results in only interfacing, not integration. In other words, the effects of other modules are not taken into account in the design and working of a module. The second approach is to start at the top and design a comprehensive system from the very beginning. The drawback of this approach is that it is not adaptive to frequent system or environmental changes. What is needed is a combination of the two approaches that can exploit the advantages of the both without incorporating their drawbacks.

The agent-based framework presented in this paper can be thought of as such a hybrid approach. On one hand a general framework for the total system can be specified,

on the other hand the modules can be individually implemented within the overall framework. Agents are the basic entities in an agent-based system. For example, they can be either stand-alone processors interconnected on a network or heterogeneous components of a system. Associated with every agent is a set of procedures and functions that define the meaningful operations of that agent and how it interacts with the other agents. From the design perspective, agents model the entities in the application domain. For example, if the system under consideration is an organization, the agents can model the different departments within the organization. Thus, the agent-based approach provides an "architectural framework" for system integration.

The traditional approach of managing a system utilizes functional decomposition to specify the tasks to be completed by each component of the system. In contrast, the agent-based approach emphasizes the importance of information and control by utilizing the relationships and interdependencies between the agents as a fundamental part of the system. This point of view is shared by recent research in agent-based systems (Lee et al.[1993]) and network organizations (Ching et al.[1993]). Thus, the design of the information and control system resulting from using an agent-based approach represents a tighter coupling of data and functionality (or behavior). It is both natural and flexible. It is natural in the sense that the agents are closely identified with the real-world elements or concepts which they model. It is flexible in the sense of quickly adapting to changes in the problem specifications. The main advantages offered by the agent-based approach are uniformity, modularity, and reconfigurability.

In a fundamental way, a parallel can be drawn between the proposed agent-based framework for system integration and the object-oriented model for software development (Kamel&Syned[1989]; Korson&McGregor[1990]; Pan&Tenenbaum[1991]). Both frameworks emphasize the modularity of the approach in which the basic element of the system is the same (objects in object-oriented approach and agents in agent-based approach). The uniformity not only provides the agent-based approach the portability and the ease of reuse (as in the object-oriented approach), but it also provides the key linkage for integration of a system.

In order to apply our multi-agent framework to system integration we first need to define what the agents represent. The important concept in the application of our multi-agent framework to system integration would be the *encapsulation* of the different entities or processes involved into their ***behavior models***. The behavior models are computational processes that can mimic the activities of the corresponding entities(Ferber&Carle[1991]). For example, a simulation model can be thought of as a behavior model.

We define an *agent* corresponding to each entity as a decision node consisting of

• a functional component that in turn consists of the behavior model of the entity and computational procedures;

• a knowledge base which consists of a list of the agent's acquaintances along with their skills and cases related to its past experience;

• a control unit which consists of communication protocols that specify how the agent would react, in response to the messages received, as a function of the state of its local state and a learning module which modifies the knowledge base based on its interaction with other agents and its current performance.

The resulting integrated information system can thus be thought of as a collection of agents - software modules, information subsystems, data bases, and decision units - connected together via a communication network, sharing and exchanging information. This information processing perspective also underscores the fact that the advantages of distributed processing, namely reliability and easy access of information, can be brought to bear upon the integrated system. It also supports the view that the key in achieving system integration is coordination through the use of appropriate levels of computers and information/communication technologies (Mize[1991]; Roboam&Fox[1990]). More importantly, it provides a means for coordination of the activities of the agents as well as integration of the heterogeneous components of a system.

Figure 3 shows a general representation of the application of the multi-agent framework to system integration. For example, the entities can be the different computer-based machines in a shop floor that have to be integrated. The physical links would correspond to the material handling system that interconnects the islands. The application of the multi-agent framework for integrating the working of the islands of automation would then involve creating the agents corresponding to each island of automation and enabling them to interact and control the entities. Note that the agents do not necessarily have to correspond to physical entities (for e.g., Agent A in figure 3). They could be logical decision modules which also have to be integrated with the working of the physical entities. In other words, system integration not only pertains to integrating the activities of physically distinct stages or processes but also includes the coordination of logically separate decision modules. Each agent can in turn be analyzed within the multi-agent framework as consisting of sub-agents (for e.g., Agent 4 in figure 3).

4. Information Systems Integration in a Printed Circuit Boards Manufacturing Environment

In this section we present an example of applying the above framework of coordination in the manufacturing domain. Specifically, we look at the problem of system integration in a real world Surface Mount Technology (SMT) plant for manufacturing Printed Circuit Boards (PCB). We analyze the coordination problems among the various components of the system at four different levels of granularity. The purpose is to ensure that the individual IS and decision modules act to meet global system objectives in a distributed fashion.

Figure 4 shows the schematic of the PCB manufacturing environment under consideration. It consist of surface mount technology (SMT) based assembly lines where the placement of the components on the PCB boards takes place. The facility consists of four such assembly lines and thus has considerable flexibility. The four lines together constitute the so-called *Feeder* stage of the facility. The facility is designed to produce 18 different types of PCB boards. Each SMT line consists of screen printers for masking and attaching solder on the bare PCBs, three types of automated component placement machines (Fuji, Zevatech, and automated robots), PROM programmers, manual assembly stations (for placement of components too large for the machines to handle), infrared ovens, and automated and manual test stations.

After leaving the SMT lines the boards go through a cleaning station. At this point the boards are ready to be processed by the Final Inspection and Testing (FIT) stage of the facility where the boards are assembled into the final products. It consists of three FIT lines that are operated manually. One of the main issues concerning this facility is the coordination between the two stages so that the production of boards that go into the same product is coordinated. The current practice at the facility is to de-couple both the stages by having a very large buffer capacity between them.

The machines in the feeder stage are connected by conveyors which can hold at most two boards at a time. Thus, one of the important characteristics of the SMT lines to be considered when making scheduling decisions is the limited amount of buffer capacity available between the machines. Bar code readers are used to scan each board to determine board type. Bar code information is used to change programs for the chip placement machines, robots, ovens, and PROM programmer for each batch of PCB's without human intervention. Also, the set of components in the chip placement machines have to be changed whenever a different kind of PCB boards have to be processed by the machines. The time required to change the set of components in a placement machine, before the PCB boards can be processed by that machine, is called the set-up time. Since these changes in the machines are done manually and require considerable skill, the set-up times are significant as compared to the processing times of the PCB boards on the machines.

Normally, the different kinds of PCB boards require components some of which are also needed by the other PCB boards. Therefore, the time required to set a machine for processing a kind of PCB board usually depends on the kind of the PCB board that was processed previously on the machine. In other words, the set-up times on the machines depend on the sequence in which the different kinds of PCB boards are processed. Thus, the other important characteristic of the SMT line to be considered while making the scheduling decisions is the sequence dependence of the set-up times, which usually dictate the performance of the overall system.

There are many factors in this environment that motivate the use of our multi-agent framework for integration. There are different stages in the manufacturing process that have their own objectives and are the responsibility of different managers. However, there are many interdependencies among the stages and ignoring them by trying to optimize each individual stage separately leads to a sub-optimal system performance. For example, the assembly stage and the feeder stage have their own objectives of minimizing the through put time and reducing the inventories level, but are interdependent because of the flow of products from one stage to another. Even within one stage there are different information systems or decision modules that have their own well-defined functions but are interdependent on each other for information. For example, in the feeder stage the information system responsible for scheduling each flow line is dependent on the information about the schedule of other flow lines primarily because of the complex product structures requiring parts of the same product being manufactured on different lines. Similarly, the information system for each of the flow lines itself consists of different decision modules that need to be integrated. For example, the lot-sizing, sequencing, and capacity planning modules have to interact in order to construct a feasible schedule for the flow line. All these factors motivate the use of our framework for integration as it allows the underlying information system to retain its distributed nature at the same providing it with the means for achieving the desired coordination and integration.

The main goal of this research is to model the complex information flows and decision processes involved in the manufacturing example described earlier and to develop coordination mechanisms that can make the system more efficient and effective. The important issue in developing the coordination mechanisms is how to maintain the distributed control of the entities involved, at the same time providing them with means of information exchange to achieve efficient integration of the processes.

Figure 5 presents the information infrastructure for the manufacturing environment discussed above and shows various software modules and databases that have to be integrated. For example, based on the customer orders for the final products, the product

structure information, and the parts already in stock, the material resource planning (MRP) module has to update the daily part demand information and interact with the short interval scheduler (SIS) to set realistic due dates for the final products. The SIS, in turn, has to interact with modules for lotsizing, sequencing, capacity planning, and simulation to find the daily schedules. Based on the information about the daily schedules, the dispatching rules, and the current machine and inventory level status, the dynamic scheduler informs the flowline controllers of the actual dispatching schedule on a continuous basis. The controllers also update the machine and inventory status database from time to time.

One of the main difficulties in coordinating the above activities is that there is no central control or authority overlooking the entire operation. That is, the underlying information system is distributed. As a result, the various objects, modules, and processes need to coordinate with each other. In practice, for example, the MRP manager would constantly negotiate with the scheduling manager to decide the due-date committed to a customer's order. This type of coordination can be supported by a computerized negotiation support system (Hosapple et al., 1991) or conflict resolution (Klein[1991]). In our approach, the multi-agent framework provides an excellent alternative as it allows for the different modules, objects, or processes --i.e., agents -- to maintain their local control by treating them as autonomous agents, at the same time providing them with a means for achieving the desired coordination.

On the logical level, we study the integration of the different modules for the working of the short interval scheduler. The problem of finding a schedule for a flow line with sequence dependant setup times has been dealt with extensively in the production and operations research literature and has been shown to belong to a class of problems called NP-hard (Bahl and Ritzman[1984]). Several approaches that treat the problem as an optimization problem have been suggested (e.g., Manne[1958]), but they all suffer from the drawback that they do not scale up to the real-world problems. As a result several heuristic approaches that decompose the scheduling problem into two sub-problems of lotsizing and sequencing and treat them as independent sub-problems have been proposed (Bahl et al [1987]; Elmaghraby [1989]). In reality, however, these two sub-problems, i.e., lotsizing and sequencing, are not independent of each other. For a more realistic solution we need an approach or a methodology that, while treating the two sub-problems separately, has the additional capability for integrating the solution process of the two subproblems. In other words, we need an approach that takes an information perspective of the problem by highlighting the need for information sharing required between the two processes. The multi-agent framework provides such an approach. It allows us to model the decision modules for solving the two subproblems as different agents (or modules),

while at the same time providing them with a means of interacting with each other and coordinating their actions while building their solutions incrementally.

We apply the multi-agent framework of system integration to this example by first analyzing the system in terms of its components (or agents) and their interdependencies and developing individual protocols for the agents that would result in a coherent behavior of the system. To reiterate, the single flow line scheduling problem is to find the weekly schedule of production that meets the daily demand, satisfies the capacity constraints, minimizes the costs and improves the productivity. The weekly schedule consists of daily lot-sizes and their sequence. Thus, we have two main decision modules of lot-sizing and sequencing in our controller, and a capacity planning module which provides the evaluation of a schedule. Below, we give the detailed analysis of the scheduling system based on our multi-agent framework.

An important issue related to system integration is the integration of different software modules working on either different parts of the same problem or different but interdependent problems. The system developed has one super class, called Scheduler, and two sub-classes, called Flowline and Factory, as shown in figure 9. The Flowline class, as the name implies, contains the detailed information about a flowline (like processing times, setup times, etc.) together with functions needed to do lotsizing and sequencing. The Factory class has information about the factory (for example, objects of the Flowline class corresponding to the number of flowlines in the factory, information about the product mix, the demand at the product level, the line of which each board is to be made, etc.) together with the functions for coordinating the production schedules of the individual lines. The Scheduler Class is essentially an abstract class created to contain the functions that are common to both the Flowline and the Factory Classes (for example., finding the best product to schedule based on their utilities, scheduling the production for a day, etc.)

The coordination among the modules is done at two levels. At the first level the Factory Class does the lot-sizing at the product level by finding the utilities of producing each product and scheduling the product with the best utility for production. In order to find the utility for a product it sends a message requesting the utility of producing each of the constituent boards to the Flowline Object corresponding to the flowline on which the board is made. In other words, the utility of producing a product is calculated based on the utilities of producing the individual boards going into that product.At the second level, the control is passed to the individual Flowline Objects for coordinating their production schedules. Each Flowline Object does the scheduling concurrently. The coordination and message-passing among the Flowline objects is achieved by using "shared data structures" (Lee et al.[1993]).

5. Implementation and Performance Evaluation

In this section we present a summary of the results and performance evaluation of the implementation discussed above. To evaluate the impact of integration and the effectiveness of coordination on the first level, experiments were conducted to evaluate the integration of the three decision modules of lot-sizing, sequencing, and capacity planning as explained in section 4. The level of work-in-process, need of overtime, and the cycle-time of the process are used as measures of the process efficiency. The integration of the three modules is essential as each module is designed for a specific function and uses only a specific criteria. The generation of a schedule for the flowline, on the other hand, requires some form of integration of the multiple objectives and functions. For example, the lot-sizing module determines the lot sizes to be produced each day and only tries to minimize the total costs associated with it. The sequencing module tries to find the best sequence for each day that minimizes the total cycle time (or makespan). However, what is needed is a schedule that not only meets the above two objectives of minimizing the total costs and the makespan but also minimizes the amount of overtime used. In other words, the integration of the modules is expected to be more than just the sum of the individual modules. An information system methodology emphasizing system-wide integration and coordination provides the global perspective seeking overall performance goals rather than the performances of individual components.

To demonstrate this feature, a number of experiments were carried out to study the behavior of the system performance as a function of the interactions among the modules and the results are presented in Figures 11 and 12. The results at no interactions in the plot correspond to the results given by the individual modules functioning independently without any integration. As hypothesized earlier, the integration of the modules leads to a significant improvement in all the three performance measures. All the results are averaged over five trials. At the second level, experiments were carried out to evaluate the integration of the software modules corresponding to each of the four flowlines in the factory as explained in section 5. Again, the integration and coordination of the software modules of the different flow lines is important because of the need to coordinate the production of the board units made on different lines but going into the same final product. A lack of coordination among the modules leads to the work in process (WIP) inventory buildup.

The object-oriented system described in section 5 was tested on five different problems consisting of four flowlines, ten products, and twelve types of circuit boards. Each flowline made three different boards. The criteria used for evaluation were the total

cycle time of the four lines combined and the WIP inventory build up in terms of the number of board-days. Table 1 shows the results of integrating the modules together with that of using the different modules independently without any coordination and integration. The results are averaged over the five problems. The results show that both the total cycle-time and inventory build-up due to lack of coordination are reduced by a significant amount. These results clearly demonstrate the benefits of integrating and coordinating the working of various decision and software modules in an information system. They illustrate the efficacy of systems integration. These results also confirm that the concept of system integration is especially important for supporting business processes such as the one we have in the printed circuit board manufacturing system.

In general, the performance results from the experiments confirm that information systems can be the enabler of improved business processes involving multiple steps of activities, processed by a number of agents (Davenport, 1990; Hammer, 1990). Process parameters such as work-in-process, need of additional overtime, and the cycle-time of the process are all measures of the process efficiency. As shown by the experimental results, the system integration approach enforces the coordination among the agents as well as among the subprocesses. As a result, both the cycle-time and work-in-process inventory are reduced.

6. Conclusions

In this paper we have presented a general multi-agent framework for coordinating and integrating a collection of stand alone units, each of which can be viewed as an "agent." The main advantages of using the above framework are: (1) it provides uniformity of a general framework for the total system by treating the heterogeneous components in a homogeneous fashion, (2) it provides modularity by allowing different modules to be individually developed and implemented, (3) it provides the capability to reconfigure the system easily, and (4) it provides for distributed control by allowing the "agents" to retain their local control at the same time providing them with the capability for interacting and coordinating their activities through message-passing.

We presented a real world example of a manufacturing environment and showed how the above framework can be applied for integrating not only different decision and software modules but also integrating physically distinct stages of a system. One of the most important applications of the framework presented here can be the integration of various heterogeneous components, data bases or software systems, at the organizational level.

References

ADLER, M., DURFEE, E., HUHNS, M., PUNCH, W., AND SIMOUDIS, "AAAI Workshop on Cooperation Among Heterogeneous Intelligents," *AI Magazine*, Summer, 1992.

BAHL, H.C. AND RITZMAN, L.P., "A Cyclical Scheduling Heuristic for Lot Sizing with Capacity Constraints," *Intl. Journal of Prod. Res.*, vol. 22, no. 5, pp. 791-800, 1984.

BAHL, H. C., RITZMAN, L. P., AND GUPTA, J. N. D., "Determining Lot Sizes and Resource Requirements: A Review," *Operations Research*, Vol. 35, No. 3, May-June 1987, pp. 329-345.

CHANDY, K. M. AND MISRA, J., "How Processes Learn," *Distributed Computing*, 1: 40-52, 1986.

CHING, C. , HOLSAPPLE, C.W., AND WHINSTONS, A. B., "Reputation, Learning, and Coordination in Distributed Decision Making Contexts," *Organization Science* Vol. 3, No. 2, pp. 275-297, 1992.

CONRY, S., KUWABARA, K., LESSER, V., AND MEYER, R., "Multistage Negotiation for Distributed Constraints Satisfaction," *IEEE Transactions on Systems, Man, and Cy.*, V.21, N. 6, 1991, pp.1462-1477.

CROWSTON, K., "A Topology og Organizational Coordination Methods," Working Paper, Sloan School of Management, MIT, 1991.

DAVENPORT, T.H. AND SHORT, J.E., "The New Industrial Engineering: Information Technology and Business Process Redesign," *Sloan Management Review*, Summer 1990.

DURFEE, E. & LESSER, V., "Negotiating Task Decomposition and Allocation Using Partial Global PLanning," in *Distributed Artificial Intelligence, Vol. II*, L. Gasser, and M. Huhns (Eds), Pitman, London, 1989, pp.229 -244.

DURFEE, E., LESSER, V., and CORKILL, D. "Cooperative, Distributed Problem Solving," in *The Handbook of Artificial Intelligence, Vol. IV*, A. Barr, P. Cohen, and E. Feigenbaum (Eds),Reading, PA: Addison Wesley, 1989, pp. 83-147.

ELMAGHRABY, SALAH E., "On Heuristics and their Performance Evaluation for Dynamic Lot Sizing," *Management Science*, vol 26, No. 1, 1989, pp. 1-10.

FERBER, J. AND CARLE, P., "Actors and Agents as Reflective Concurrent Objects: A Mering IV Perspective," *IEEE Transactions on Systems, Man, and Cy.*, V.21, N. 6, 1991, pp.1420-1408.

GASSER, L. AND HUHNS, M., "Themes in Distributed Artificial Inteeligence Research," in *Distributed Artificial Intelligence, Vol. II*, L. Gasser, and M. Huhns (Eds), Pitman, London, 1989, pp.259-290.

HAMMER, M., "Reengineering Work: Don't Automate, Obliterate," *Harvard Business Review*, July/August, 1990.

HALPERN, J. Y. AND FAGIN, R., "Modelling Knowledge and Action in Distributed Systems," *Distributed Computing*, 3: 159-177, 1989.

HEWITT, C., "Viewing Control Structures as Patterns of Message Passing," *Journal of Artificial Intelligence*, 8(3), pp.323-364, 1977.

HEWITT, C., "Open Information Systems Semantics for Distributed Artificial Intelligence," *Artificial Intelligence*, 47, pp.79-106, 1991.

HOLSAPPLE, C.W., LAI, H., AND WHINSTON, A.B., "Negotiation Support Systems: Roots, Progress, and Needs," *Journal of Information Systems*, Vol. 1, No. 4, 1991.

HUHNS, M. AND BRIDGELAND, "Multiagent Truth maintenance," *IEEE Transactions on Systems, Man, and Cy.*, V.21, N. 6, 1991, pp.1437-1445.

KAMEL, M. AND SYNED, A., "An Object-Oriented Multiple Agent Planning Systems," in *Distributed Artificial Intelligence, Vol. II*, L. Gasser, and M. Huhns (Eds), Pitman, London, 1989, pp.259-290.

KLEIN, M., "Supporting Conflict Resolution in Cooperative Design Systems," *IEEE Transactions on Systems, Man, and Cy.*, V.21, N. 6, 1991, pp.1379-1390.

KORSON, T. AND McGREGOR, J.D., "Understanding Object-Oriented: A Unifying Paradigm," *Communications of the ACM*, Vol. 33, No. 9, pp 40-60, 1990.

LEE, K-C, W. MANSFIELD, AND A. SHETH, "A Framework for Controlling Cooperative Agents," *IEEE Computer*, pp. 8-16, July 1993.

MALONE, T. W. AND CROWSTON, K., "Toward an Interdisciplinary Theory of Coordination," Technical report, CCS TR# 120, SS wp# 3294-91-MSA, MIT, Cambridge, MA., 1991.

MANNE, A.S., "Programming of Economic Lot Sizes," *Management Science*, Vol. 4, No. 2, pp.125-135, 1958.

MILNER, R., *A Calculus of Communicating Systems*, (Lect Not Comput Sci, vol 92) Springer, Berlin Heidelberg New York, 1980.

MIZE, J.H., H.C. BHUSKUTE, D.B. PRATT, AND M. KAMATH, "Modeling of Integrated Manufacturing Systems Using an Object-Oriented Approach," *IIE Transactions on Design and Manufacturing*, Vol. 24, No. 3, pp.14-26, July 1992.

MIZE, J. H., "Fundamentals of Integrated Manufacturing," in J. Mize (Ed.), *Guide to Systems Integration,* Industrial Engineering and Management Press, Institute of Industrial Engineers, Norcross, Georgia, pp. 27-43, 1991.

PAN, J. AND TENENBAUM, J., "An Intelligent Framework for Enterprise Integration," *IEEE Transactions on Systems, Man, and Cy.*, V.21, N. 6, 1991, pp.1391-1408.

ROBOAM, M. AND FOX, M., "The Enterprise management Network Architecture: A PLat for Multi-Agent Manufacturing Systems," Working Paper, Robotics Institute, Carnegie Mellon University, 1990.

SILVER, E.A. AND MEAL, H.C., "A Heuristic Selecting Lot Size Requirement for the case of a Deterministic Time Varying Demand Rate and Discrete Opportunities for Replenishment," *Production and Inventory Management,* 14, 1973, pp.64-74.

SHAW, M.J. AND M. FOX, "Distributed Artificial Intelligence for Group Decision Support," *Decision Support Systems*, Vol. 9, pp. 349-367, 1993.

SHAW, M.J., SOLBERG, J., AND WOO, T., "System Integration for Intelligent Manufacturing: An Introduction," *IIE Transactions on Design and Manufacturing*, Vol. 24, No. 3, pp. 2-6, July 1992.

SHAW, M.J. AND WHINSTON, A., "Learning and Adaptation in Distributed Artificial Intelligence Systems," in *Distributed Artificial Intelligence, Vol. II,* L. Gasser, and M. Huhns (Eds), Pitman, London, 1989, pp.259-290.

SIKORA, R. AND SHAW, M., "A Multiagent Framework for The coordination and Integration of Information Systems," Beckman Institute Technical Report AI-DSS-94-01, University of Illinois, Urbana, ILL, 1994.

WERNER, E., "Cooperating Agents: A Unified Theory of Communication and Social Structure," in *Distributed Artificial Intelligence, Vol. II,* L. Gasser, and M. Huhns (Eds), Pitman, London, 1989, pp.3-36.

SYCARA, K., "Multiagent Compromise via Negotiation," in *Distributed Artificial Intelligence, Vol. II,* L. Gasser, and M. Huhns (Eds), Pitman, London, 1989, pp.119 -138.

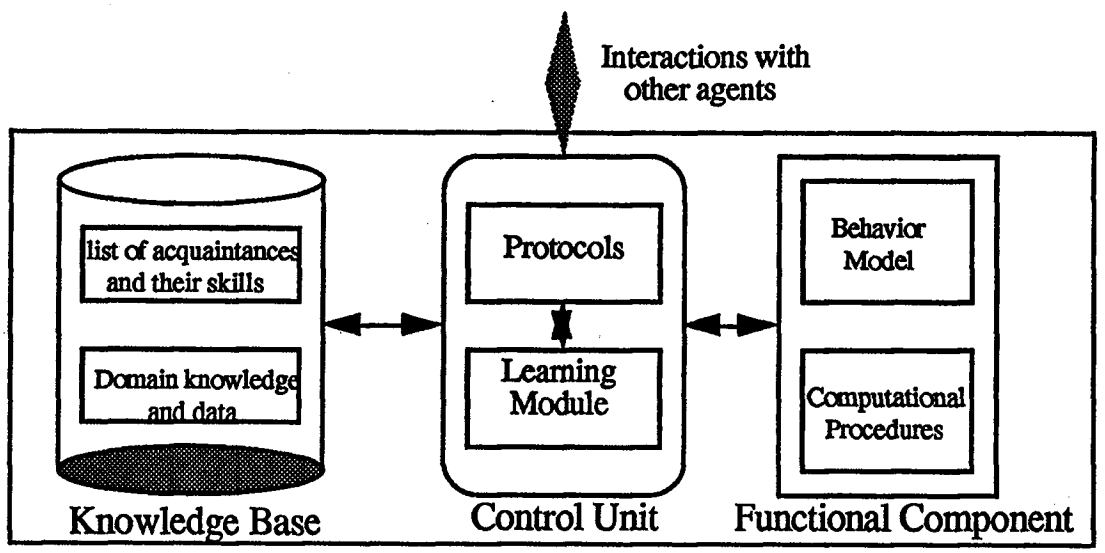

Figure 1. Conceptual Model of an Agent

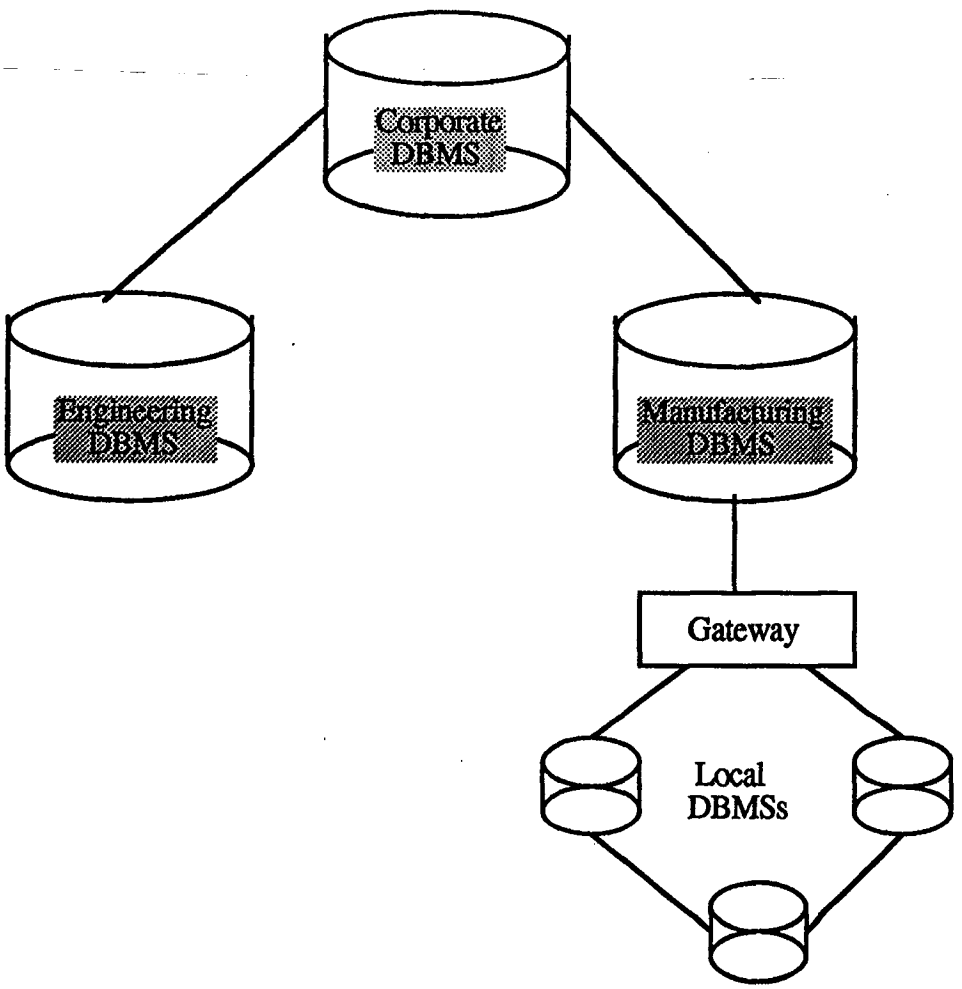

Figure 2. An Example Application in Database Management

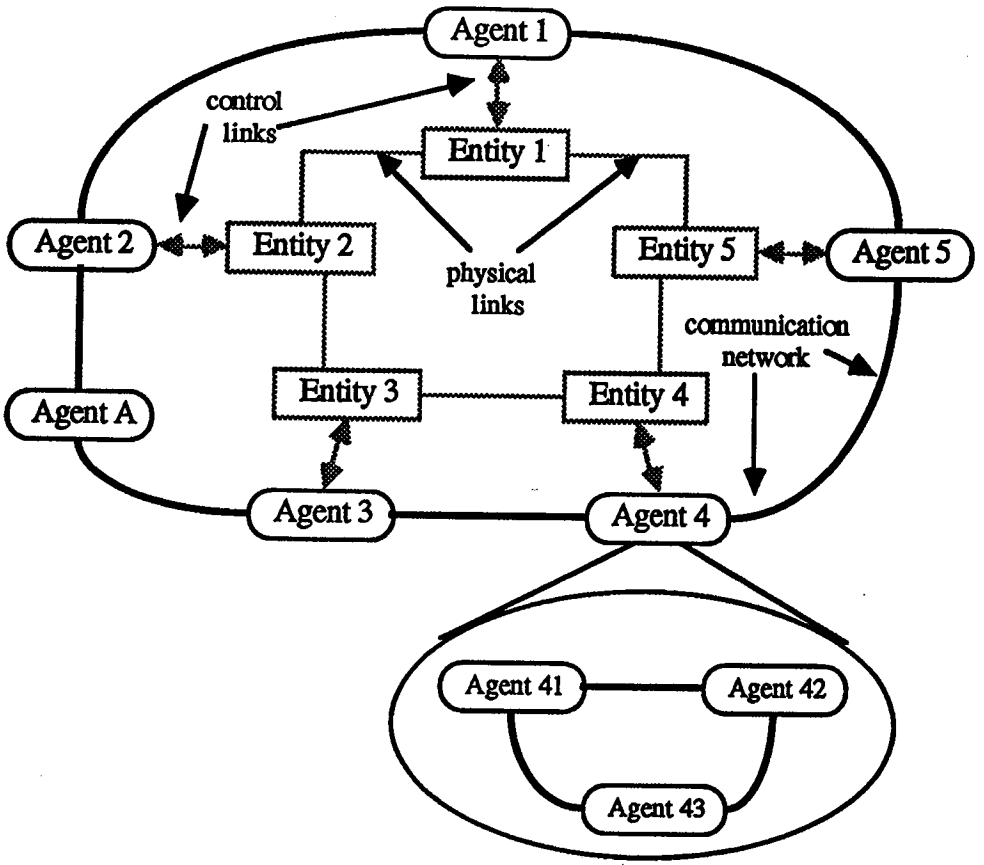

Figure 3. Multi-Agent Framework for System Integration

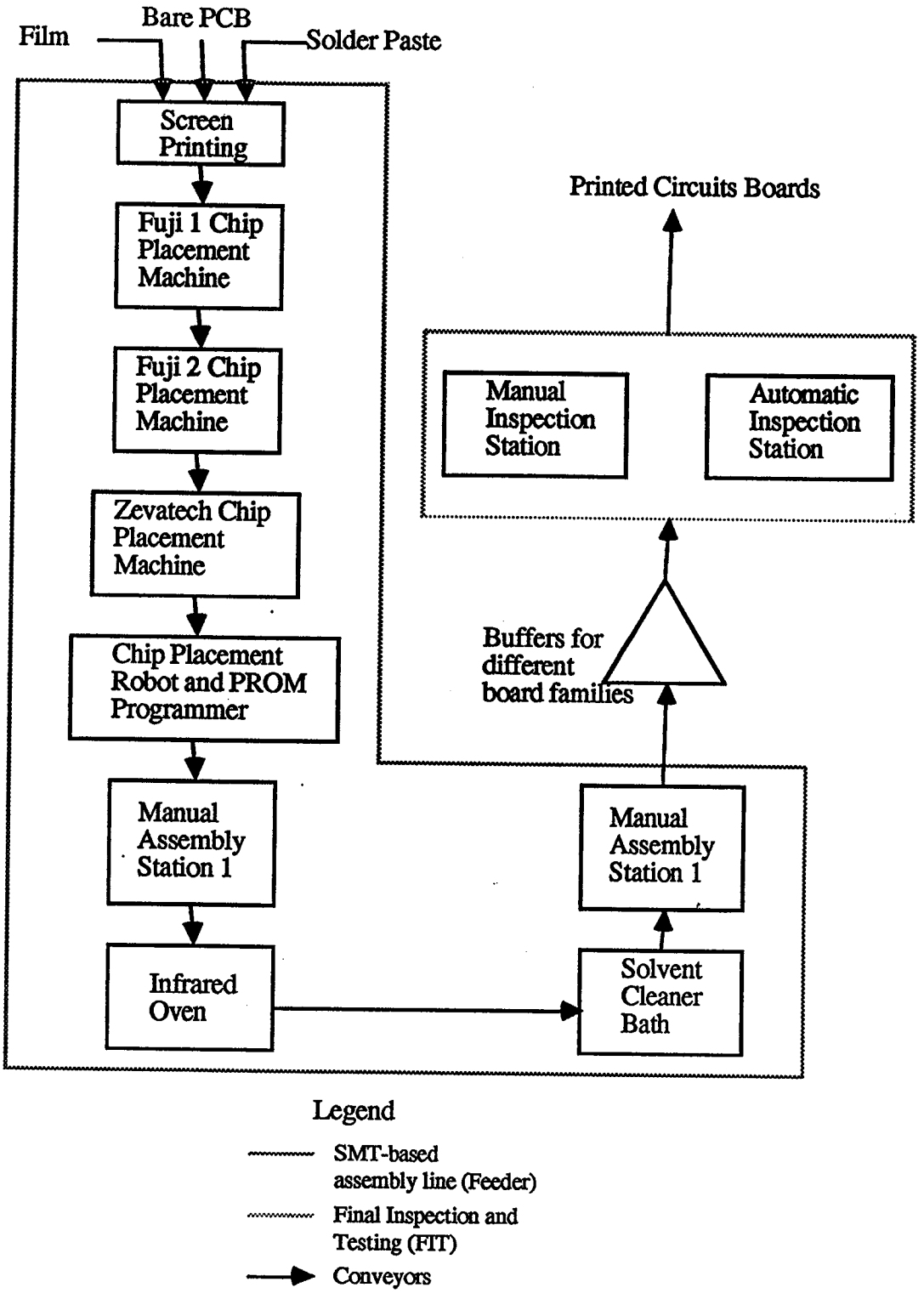

Figure 4. The Schematic of the SMT Environment

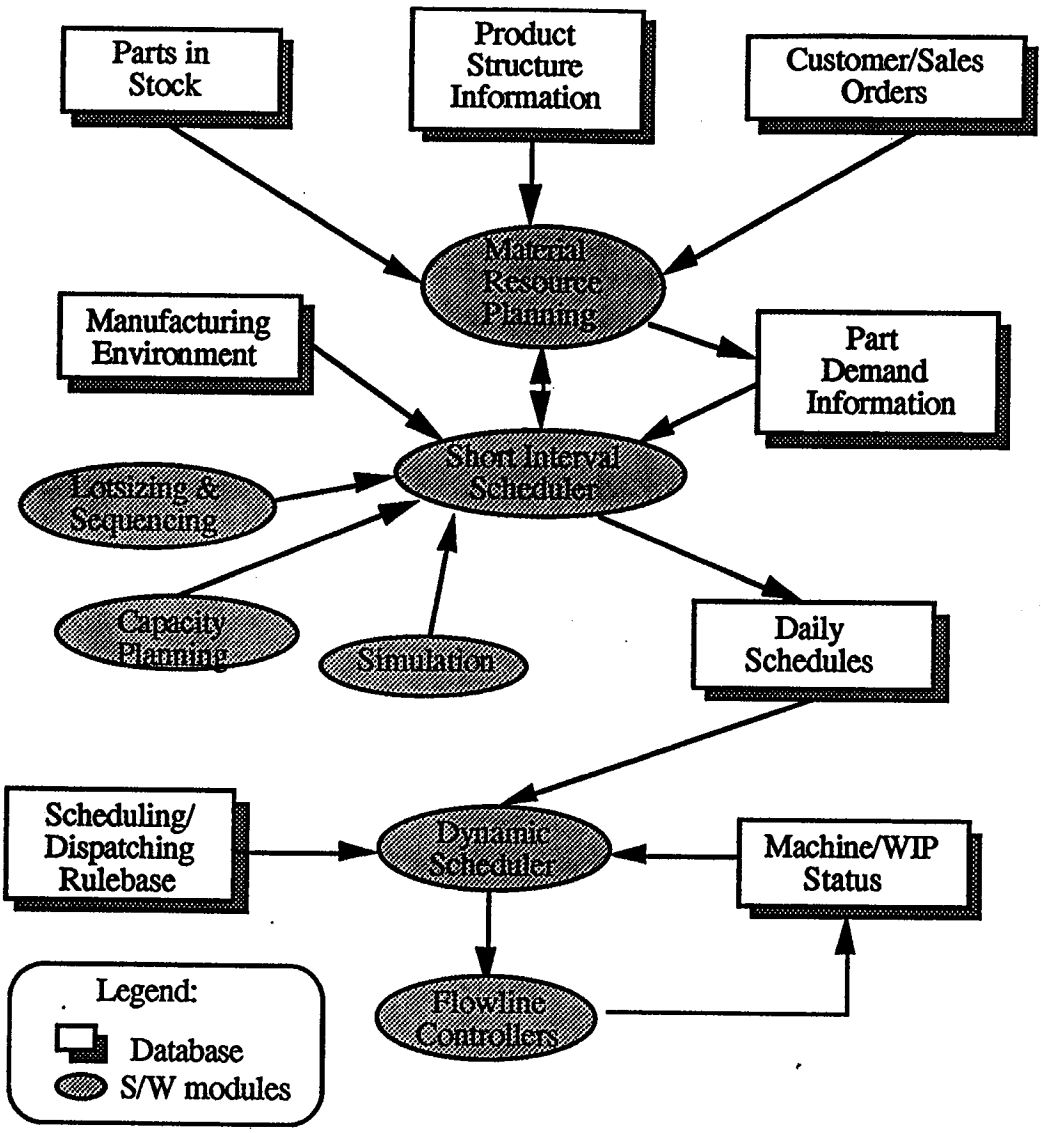

Figure 5. Information Infrastructure for Factory Integration

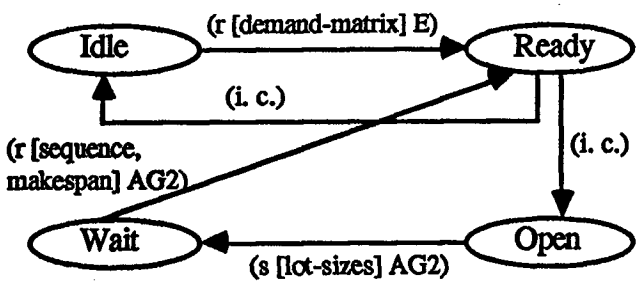

Figure 6. State Diagram for the Working of Agent AG_1

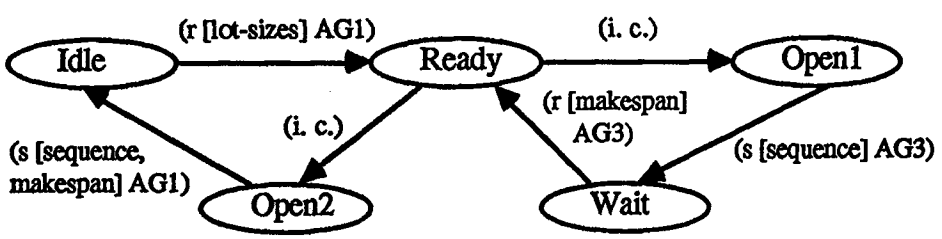

Figure 7. State Diagram for the Working of Agent AG_2

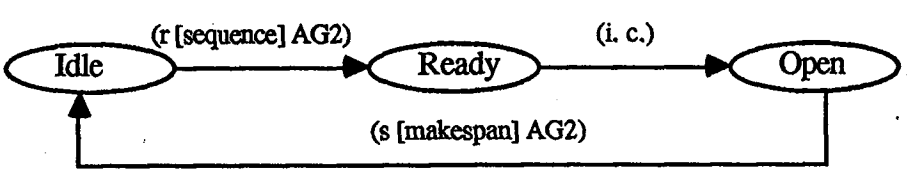

Figure 8. State Diagram for the Working of Agent AG_3

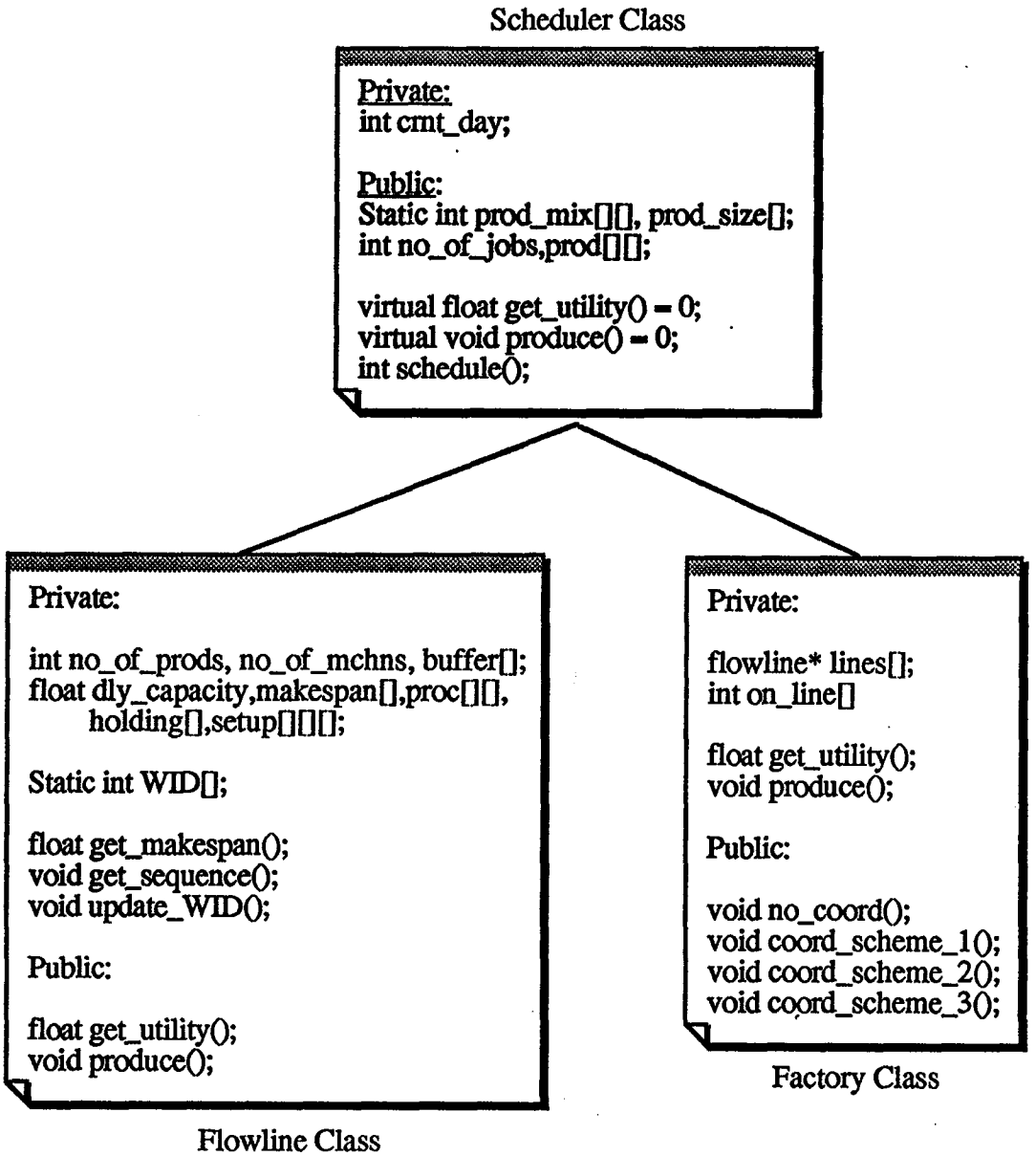

Figure 9. Class Structure of the Object Oriented System

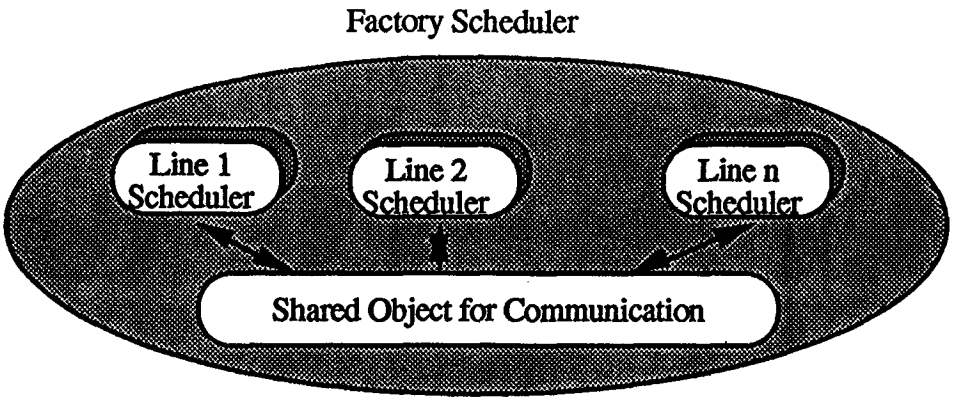

Figure 10. Application Instance of the Object-Oriented System

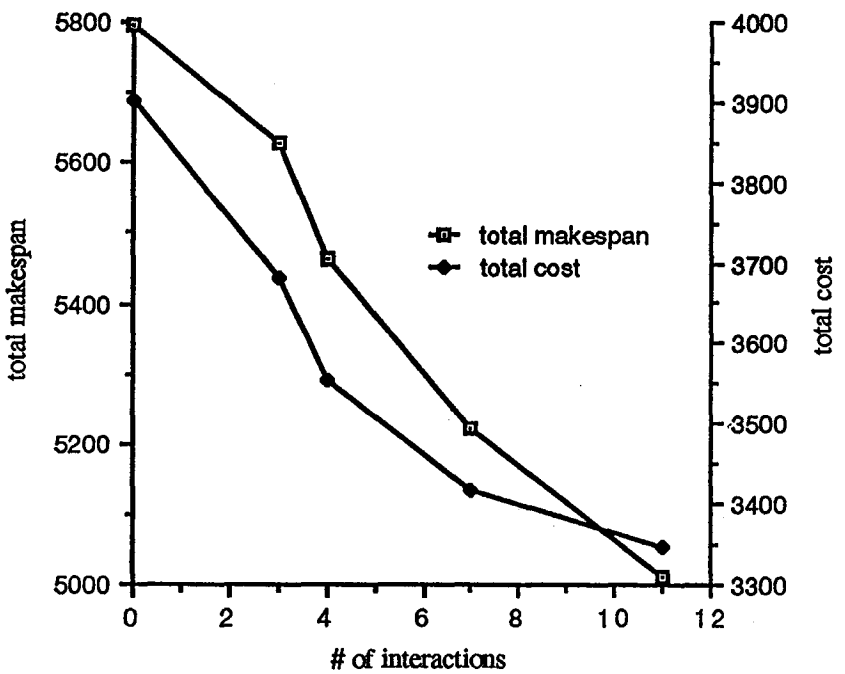

Figure 11. Improvement in the Makespan and the Total Cost as a Function of the Number of Interactions

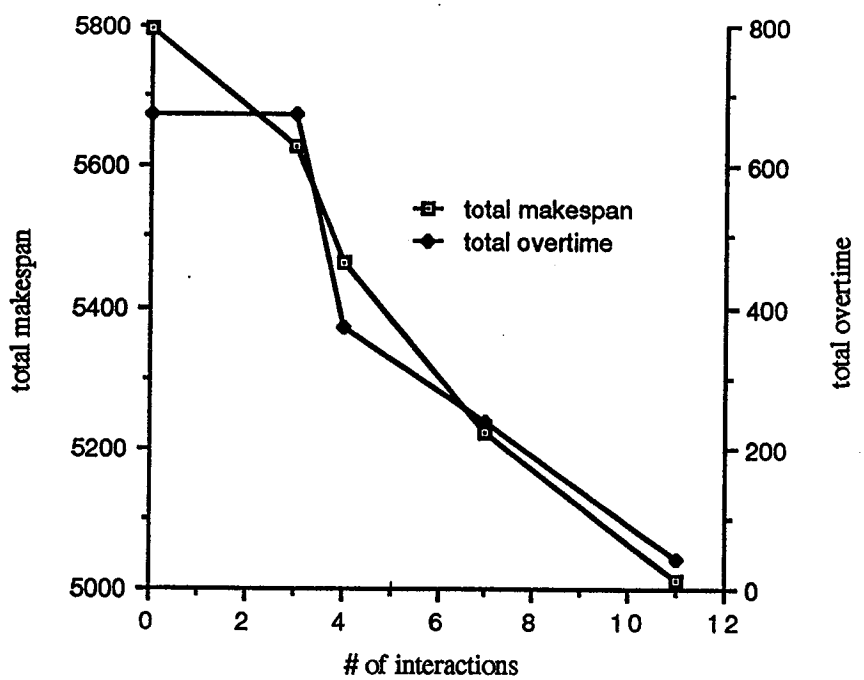

Figure 12. Improvement in the Makespan and the Total Overtime as a Function of the Number of Interactions

Method	Makespan	Board-Days
w/o Coordination	5950.13 min.	535
with Coordination	5996.37 min.	84

Table 1. Improvement in Performance Due to Coordination