An Autonomous Cooperative System for Material Handling Applications

Francisco Maturana[1] **Sivaram Balasubramanian**[1] **Dave Vasko**[1]

Abstract. The key to the creation of flexible automation is to use inherent redundancy in the capabilities of the system being controlled. This requires the control system to be continuously self-configurable under varying conditions. The ability of the control system to react to and predict changes will ultimately determine the economic viability of that system. In this paper, an Autonomous Cooperative System to control material handling systems is presented. Important components of the material handling system are combined with intelligence and autonomy rules to flexibly control the operations of the physical equipment. Control is carried out while the overall operation of the system is optimized through cooperation among the controlled sections. The operation of the material handling system is observed during conditions of equipment and product changes. The results from simulation show how an Autonomous Cooperative System can be used to reduce the impact of changes in material handling applications.

1 INTRODUCTION

Programmable controllers provided an alternative to hardwired relays in industrial applications. The relay ladder logic provided a flexible mechanism to make changes to the operation of machines in the factory without rewiring relays. Typically, a single programmable controller controlled a machine or a group of related machines. As technology progressed, the execution speed and I/O capacity of the programmable controllers increased, but not without a cost. As the applications became larger, the cost of development and maintenance of programs over their lifecycle rose dramatically [1].

The advent of computer networks at the factory floor allowed control to be divided into cells thus allowing geographic distribution and reduction in the size of individual programs. In multi cell systems, control engineers confronted a major problem of coordinating the operations of many small controllers. The coordination of these distributed controllers relied on interlocks of data or I/O points in hierarchical structures of master controllers. The proper operation of the linked controllers depended on the accuracy of the preplanned operations. The expansion, maintenance and development of such systems were tightly coupled and their associated lifecycle cost remained high.

Trends predict a future increase in the frequency of change in factories. This involves rigorous quality and delivery performance measures [2][3][4]. To cope with these changes, the underlying control system needs to tackle the numerous changeovers in the mainline configuration, tool allocation, material distribution, process steps, and product quality. The control system needs to configure its nodes into effective networks within a short period.

Flexible change in the factory relates to the effective partitioning of information [5][6]. The solution to flexible change in industrial environments depends on effective information partitioning and coordination protocols for automated units. In the context of this paper, flexibility is the capability of the system to react and/or pro-act in accordance with the system needs, thereby organizing the system resources and capabilities in the most efficient manner.

An Autonomous Cooperative System (ACS) architecture to augment the intelligence of the controllers is proposed. This architecture operates in enterprise and plant levels. This consists of autonomous object components that represent machines and processes. These components organize the machines to enhance the performance of the operations and changes in configuration.

Material handling systems were selected as the factory layout due to their large number of components, parameters, and rigid operation times. These systems present discrete complexity and stochastic combination of behaviors that interrupt the normal functioning of the system. There are no equations or methodology that can describe the overall system to simplify control.

In this paper, we describe a methodology for creating an ACS application for automated material handling systems and present results from simulation.

2 BACKGROUND

Important developments in cooperation techniques simplify the distribution of knowledge among dissimilar and specialized systems [7][8]. These expert systems drive solutions toward pre-established goals. Other models facilitate interaction and decision-making tasks among distributed software [9][10]. These contributions provide tools toward the realization of smarter control systems.

Distributed Artificial Intelligence (DAI) research proposes Intelligent Agents [11] to facilitate flexible and highly distributed information. Intelligent agents are self-contained software entities capable of communicating and making individual decisions. They work autonomously, handle goals, maintain beliefs, and cooperate to create solutions. The beliefs of intelligent agents in factory control correspond to the knowledge about the state of the machine and the state of the adjacent machines associated with the intelligent agents. Intelligent agents control every node in the system [12]. The Agile Infrastructure for Manufacturing Systems (AIMS) is an example in which an open information infrastructure

[1] Architecture and System Development
Rockwell Automation
1 Allen Bradley Drive
Mayfield Heights, Ohio, 44124, USA

FPMaturana@ra.rockwell.com
SBalasubramanian@ra.rockwell.com
DAVasko@ra.rockwell.com

permits access into agile production services [13]. The Intelligent Agent (IA) framework [14] demonstrates the integration of humans with computers in large distributed systems. Presently, the Holonic Manufacturing Systems (HMS) consortium carries out important standardization work for the use of intelligent agents in industry [15]. This latter research provides important results to build inexpensive, expandable cooperative systems.

Other research efforts establish analogies between biological systems and the distributed behaviors found in factory operations [16][17]. Object-oriented technology permits the construction of distributed software, which facilitates the software scalability and behavior specifications at a lower cost [18].

This project envisions the ACS architecture as a solution to cope with flexibility requirements. This consists of the creation of autonomous software wrappers around the physical equipment. The software wrappers are intelligent agents that act during the evaluation of plans and the selection of control programs. ACS promotes the creation of high-value plans (e.g., increased throughput, increased machine utilization, optimized load distribution, etc.) by providing the autonomous software with application heuristics, agent language, and coordination protocols.

The ACS domain extends from the control level into the information level. This permits the combination of different requirements and priorities to process the enterprise information. Intelligent agents achieve coordinated decisions in and across networked controllers.

ACS is an emerging technology that will provide an efficient solution to flexibility requirements and will bring together the different pieces of distributed control systems.

3 ACS ARCHITECTURE

ACS is an adaptive architecture, which is founded on the following specifications: Autonomy, Cooperation, Communication, Reliability, Fault tolerance, Learning, and Forecasting. These specifications are defined as:

- *Autonomy*: Agents make local decisions and are responsible for carrying out the decisions toward successful completion;
- *Cooperation*: Agents merge their capabilities into collaboration groups to adapt and respond to diverse events;
- *Communication*: Agents share a common language to encode states and plans;
- *Reliability*: Agents perform their activity autonomously and through cooperative interaction to accomplish global plans;
- *Fault tolerance*: Unforeseen failures are circumvent by using alternative plans whenever possible;
- *Learning*: Agents use past actions to direct future responses and to minimize computing overheads;
- *Forecasting*: Agents proactively propose plans of action to enhance the system performance or to prevent the system from entering into a harmful state.

3.1 Autonomous control architecture

Each component of ACS has an activity, a connection interface, and a visualization system. The autonomous software wrappers for the machines and processes are called Autonomous Cooperative Units (ACUs). The architecture consists of 5 main components: Human Machine Interface, Broker ACU, Service ACU, Supervisor ACU, and Equipment, as shown in Figure 1. The activities of the industrial environment can be divided among Supervisor and Service ACUs. This division of activities is similar to that of client and server partitioning. The ACS components have a common communication language and interface that permits communication from and to any level within the control infrastructure. Each ACU uses a job description language to encode the application information, parameters, and states.

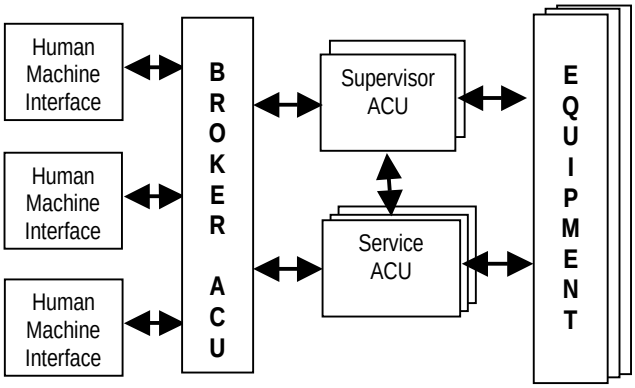

Figure 1. Autonomous control architecture

The Human Machine Interface (HMI) provides the graphical interfaces to access the overall system information, controller settings, machine views, and configuration screens. Each ACU has knowledge about its associated HMI interface. Bi-directional event communication is allowed between the ACUs and their respective interfaces.

The Broker ACU is a facilitator directory that establishes communication links among the ACUs. Each ACU advertises its capabilities and physical address with the broker. The Broker ACU maintains the system registry and uses the application knowledge to discover dynamic relationships among the ACUs.

The Service ACU represents the physical equipment and process steps. It acts as an automation server to respond to client requests. Each Service ACU is programmed with application rules to facilitate cooperation and form coordination clusters.

The Supervisor ACU filters information between the users of the system and the Service ACUs. Users generate messages in the graphical interfaces in the form of plan requests and execution orders. The rules of a Supervisor ACU specify the application needs and the order in which actions should be committed and executed. For example, a Supervisor ACU representing an inventory resource knows that it needs to download pallets from a truck, move the pallets on conveyors, and place them into storage locations.

3.2 ACU cooperation

The ACU cooperation emerges from a cyclic exchange of information among the ACUs in which the process steps and cost are optimized. Messages conveying the process steps trigger an internal evaluation within the ACUs (Supervisor or Service). Each ACU uses its process model to verify the feasibility of performing the operation. The ACU can accept all or part of the operation. This decision depends on the equipment capability to handle the process steps of the operation. Also, if the operation is large and complex, the ACU can recruit services from peer ACUs. This evaluation requires a qualitative analysis stage in which only the capabilities of the equipment are taken into consideration.

The ACUs calculate the operation cost and execution overhead using a quantitative analysis. After this internal reasoning, the ACUs expose their decisions (e.g., process steps, schedules, and operation costs) to other ACUs in the form of bids and counter

bids. The cycle is completed when the Supervisor ACU chooses the best bid, as shown in Figure 2.

The ACU interaction is based on the exchange of synchronous and asynchronous messages, corresponding to 3 types of interactions: 1) interaction of *n*-number of Supervisor ACUs, 2) interaction of a Supervisor ACUs with *n*-number of Service ACUs, and 3) interaction among *n*-number of Service ACUs. The first type of interaction occurs when a Supervisor ACU discovers that its solution is insufficient to complete all the requests. In such a case, the Supervisor ACU recruits additional capabilities from the Supervisor ACUs. Each Supervisor ACU assumes responsibilities for a subset of requests and synchronizes it with associated supervisors. In the second type of interaction, the solution of the Supervisor ACU is sufficient for all requests. The third type of interaction corresponds to a cooperation cluster of Service ACUs.

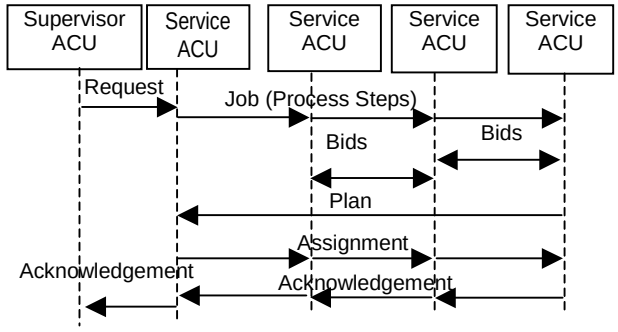

Figure 2. ACU interaction during cooperation

Figure 2 also shows the transactions for the second type of interaction. In this case, the Supervisor ACU multicasts a request to a selected subset of Service ACUs. Service ACUs subcontract additional resources using type 3 interaction. The subcontracting process increases the parallelism of the message sequencing. When the Service ACUs agree upon a global plan, a global consensus and near-optimal results is obtained. The Service ACUs inform the Supervisor ACU of all good plans and other relevant information at the end of a predefined deadline. Subsequently, the Supervisor ACU selects the best plan and assigns the plan to the Service ACUs.

4 APPLICATION OF ACS TO MATERIAL HANDLING SYSTEMS

This experiment observed different material-handling facilities. Most facilities corresponded to distribution centers or express-delivery centers. These types of facilities were very similar in their composition, but varied the way in which they handled the products. Because the distribution centers had storage resources, products arrive and stayed in the facility as inventory for a period of time until the consumers request for their distribution (e.g., warehouse). In the express-delivery centers, products stayed in the system only for the time it took to route them to a distribution vehicle (e.g., post office).

4.1 Distribution center

Distribution centers serve as consolidation and shipping resources for manufacturing industry. Various product types manufactured at different factories (providers) or procured through other channels (distributors) are consolidated into the distribution centers as inventory. The product consignments are shipped out according to sales centers or customer demand. They typically involve a wide variety of products to meet demand.

Distribution centers are dynamic environments, in which various seasonal changes and unplanned perturbations can affect the shipment of the products. The goal is to create a highly flexible control system to change product shipments, scalability, reconfiguration, and fault tolerance on the fly.

4.2 Express delivery center

The express delivery and sorting center is a subset of a distribution center. In these systems, delivery time and balanced load distribution are critical factors for success.

Products are downloaded in presort conveyors and transported to accumulation areas, where human operators move the products onto sorting conveyors. Products are measured and scanned to obtain their destination.

The sorting conveyors (conveyors plus diverters) move the products to the transport vehicles for final delivery. Products arriving in the systems are not known a priori, since they generally arrive in bulk amounts. The goal is to coordinate very fast measurements and barcode readings to balance the load in the accumulation resources. This is intended to prevent unnecessary shutdowns of the system by reducing the occurrence of jams. The express delivery and sorting facilities operate under very rigid schedules; therefore, operation times must be kept to a minimum.

4.3 Material Handling Negotiation

In this project, negotiation models represented the operations of the material handling systems, including the transactions to move parts through the system. We designed three main models: 1) products moving from providers to storage areas, 2) products moving from storage areas to distributors, and 3) products moving from providers to distributors. Models 1 and 2 corresponded to a distribution center operations. Model 3 corresponded to an express delivery operation.

In model 1, the provider alerts the inventory supervisor (i.e., Supervisor ACU) of the plant that a transportation resource will soon arrive in the facility (1). The inventory supervisor contacts the docking and storage area simultaneously to reserve their resources for handling the specified quantities, storage space, and storing time (2). The inventory supervisor selects the best bids from both resources (docking and storage area) and immediately contacts the conveyors to determine the route to reach the storage from the docking area (3). The final plan conveys information about the docking location, storage and routes. The inventory supervisor informs the provider the plan to complete the request (4). The provider allocates the offered resources and proceeds to execute the plan according to the schedule given, as shown in Figure 3.

In model 2, the consumers request the inventory supervisor to supply a product quantity (1). The inventory supervisor contacts the storage area to retrieve the product quantity (2). Upon success, the inventory supervisor contacts the docking area to park the transportation resource (container) (3). Subsequently, the inventory supervisor contacts the conveyors to determine the best route to connect the storage and the docking location (4). The final plan is then given to the consumer (5) telling the docking location and schedule, as shown in Figure 4. The consumer allocates the resources to execute the loading operation (6).

In model 3, the providers ask the plant supervisor (1) to obtain the outbound docking locations and storage (2) to receive the products (or packages for a mail delivery system) in the outbound

area after sorting. The plant supervisor contacts the conveyors to find accessible paths (3). Upon success, the plant supervisor informs the provider (4) the inbound docking where to park the transportation resources, as shown in Figure 5. The provider allocates the resources (5). In these situations, the three models above use a combination of interaction type 1,2, and 3.

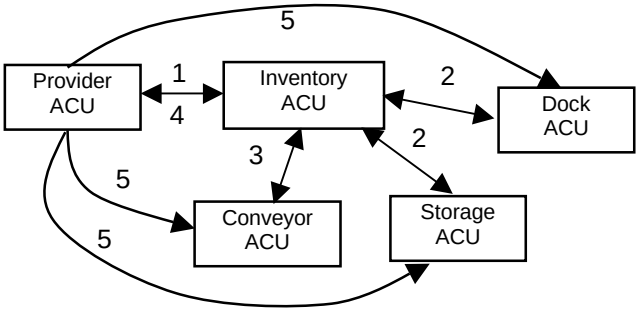

Figure 3. Inbound product negotiation

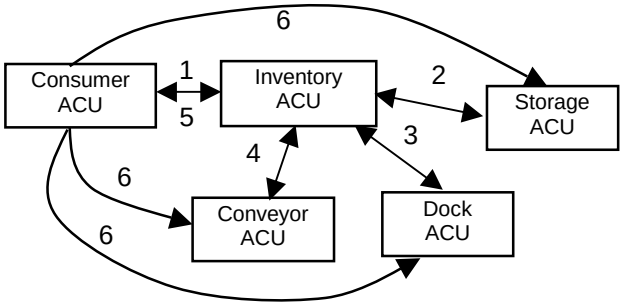

Figure 4. Outbound product negotiation

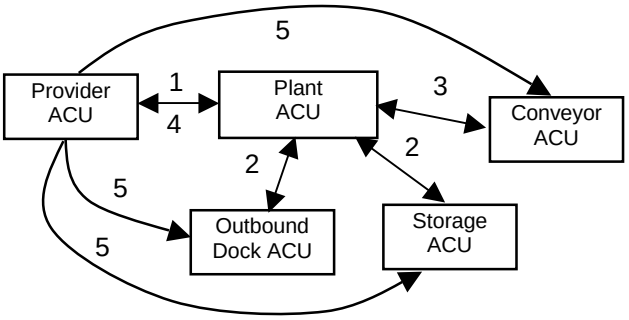

Figure 5. Express delivery negotiation

5 SIMULATION IMPLEMENTATION

This project built a simulation environment to simulate the material handling systems. Various Supervisor and Service ACUs represented the physical layer of the material handling systems. The simulation testing required several trials to ensure that the ACS architecture was capable of handling the case scenarios observed in normal operations.

Figure 6 shows the four salient components of ACS simulation environment: graphical interfaces, ACUs, execution interface agents, and discrete-event simulator. The graphical user interfaces include system view, device view and ACU view accessed in a hierarchical manner. The system view shows all controller devices in the system and the device view can be accessed for any controller device. The device view lists all ACU views for ACU present on a controller device; it can be accessed for any ACU of interest. The ACU view provides detailed configuration and operation information of the ACU.

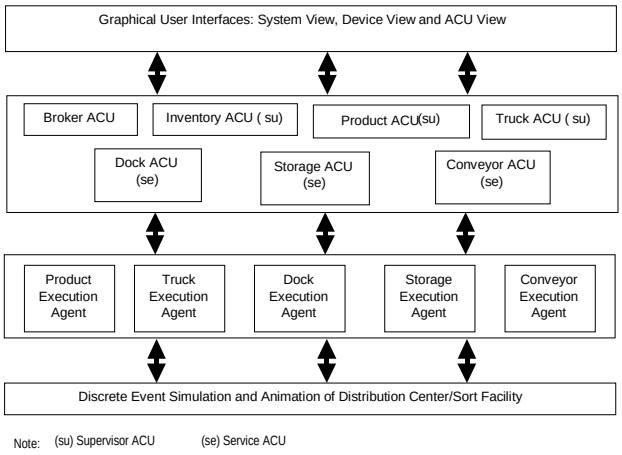

Figure 6. ACS Simulation environment

The execution agents encapsulate the logic of the physical devices. The simulator contained a simulation model of the physical entities. The system executed synchronous to real time and contained 100 ACUs. The simulation ran in a network of industrial computers with Pentium-Pro II CPU under NT 4.0. The network was based on TCP/IP through Ethernet cables.

6 RESULTS

Two important claims of the ACS architecture are flexibility and fault tolerance. These were selected to test the system. A qualitative analysis was carried out in a simulated distribution center to observe the flexibility behaviors of the system in response to changing requirements and capabilities.

Speed was not a critical issue in the distribution center simulation. However, the ability to react to changes is important in order to utilize the resource efficiently. On the other hand the speed of the material handling operations is a primary factor in fast-delivery and sorting centers. Therefore, a fast-delivery and sorting center was simulated to observe the fault-tolerance capability.

Two job orders tested the distribution center: *Inbound* and *Outbound*. An inbound request corresponded to the job order submitted by arriving trucks in the facility. The parts were downloaded from the container in the dock area and moved through the conveyors toward a storage location. The product generated an outbound request to move parts from the storage area to the distribution trucks.

An order for 100 parts was placed for shipment. Figure 7 shows the results while processing this order as the time taken for shipping every part. The shipping time includes both planning and execution. Since the system had previous knowledge about plan patterns under normal operation, the first ten parts had constant shipping time.

A conveyor section was added while planning for part number 11. The system added the new condition into the plans. This behavior was reflected in the higher planning time and lower execution time: lower shipping time for this part. Since the plans were regenerated using learned patterns, the plan creation became trivial for parts 12 through 50. When shipment for part number 51 was planned, the redundant conveyor section was failed. The new

condition resulted in higher planning and execution times: a longer path had to be taken. After learning, the planning became trivial.

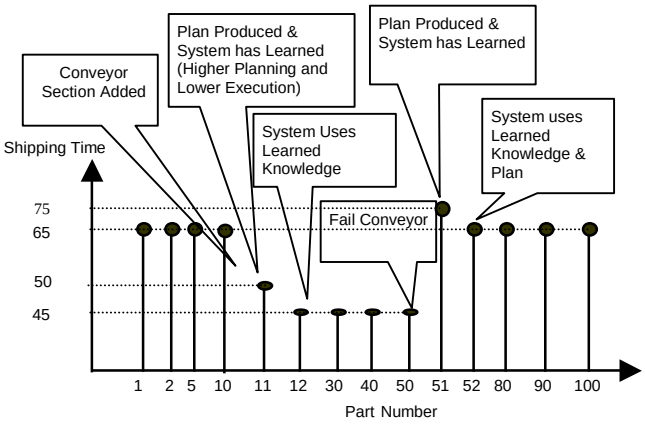

Figure 7. Distribution center behavior during configuration changes

A load of 1000 packages was used to test routing and load distribution. Packages in the inbound dock area were moved to accumulation areas, from the accumulation areas to primary conveyors for zip code and size scanning. The zip code was used to direct the routing. Balanced loads and quick responses were quantified by observation of the packages' accumulation in different locations, as shown in Figure 8.

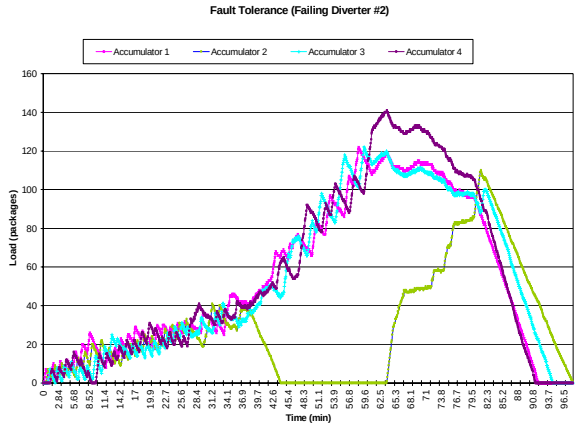

Figure 8. Fault Tolerance

The goal was to allow the system approach a high-peak load before introducing a capability failure, which consisted of closing the access to chute 2 at 36 time units. The excess packages were redirected to other chutes. A balanced load was maintained among the conveyors, between 36 and 63 times units. At 63 time units, the access to chute 2 was reestablished. This action added extra sorting capability, which was added to the plans almost instantaneously, as shown after 65 time units.

7 CONCLUSION

The results from simulation are that the machine operations and states can be encapsulated in Autonomous Cooperative Units. This is a very relevant result since the agents can be programmed with a very diverse set of rules and inference engines to support intelligent decision making. This also facilitates the incorporation of global consultation among distributed intelligent agents that continuously observe and control the state of the system for high performance and stability. The next phase of research considers the real time aspects of the architecture, which will be evaluated using real physical equipment and automation controllers.

8 REFERENCES

[1] F. Brooks, Mythical Man Month, Addison-Wesley Publishing Company, 1975.

[2] Agile Precision Sheet Metal Stamping Proposal, Advanced Technology Program, National Institute of Standard and Technology, April 11th, 1995.

[3] Proceedings of the 4th Technical Advisory Committee Meeting, NSF Engineering Research Center for Reconfigurable Machine Systems, May 6-7th, 1998.

[4] Proceedings of Auto Body Consortium, Inc.: Near Zero Stamping, Inc. Kickoff Meeting, Dec.1995.

[5] Microprocessor Report, http://www.chipanalyst.com/

[6] Gigabit Ethernet, http://www.gigabit-ethernet.org/

[7] A. Bond and L. Gasser, Readings in Distributed Artificial Intelligence, Morgan Kaufmann Publishers, Inc. 1988

[8] G. O'Hare and N. Jennings, Foundations of Distributed Artificial Intelligence, John Wiley & Sons, Inc. 1996

[9] S. Clearwater, Market-based control. A paradigm for distributed resource allocation, World Scientific, 1996.

[10] J. Rosenschein & G. Zlotkin, Rules of Encounter: Designing for Automated Negotiation among Computers, MIT Press, 1994.

[11] M. Wooldridge and N., Jennings, Intelligent agents: theory and practice. *Knowledge Engineering Review*, 10(2),115-152, 1995.

[12] R. Davis and R. Smith, Negotiation as a metaphor for distributed problem solving. Artificial Intelligence,20:63-109, 1983

[13] K.H., Park, M., Cutkosky, A.B., Conru, and S.H., Lee, An agent based approach to concurrent cable harness design, Artificial Intelligence for Engineering Design, Analysis and Manufacturing Journal, 8(1), 45-61.

[14] J.M., Tenenbaum, T.R., Gruber, and J.C., Weber, Lessons from SHADE and PACT. In Petrie, C. (Eds), Enterprise Modeling and Integration. McGraw-Hill, New York, 1992.

[15] J.H., Christensen, Holonic Manufacturing Systems: Initial architecture and standards direction, First European Conference on Holonic Manufacturing Systems, Hanover, Germany, 20pp, 1994.

[16] H. Van Parunak, "Go to the Ant": Engineering Principles from Natural Multi-Agent Systems, Annals of Operations Research, special issue on Artificial Intelligence and Management Science.

[17] L. Steels, "Toward a Theory of Emergent Functionality" From Animals to Animats: Proceedings of the 1st International Conference on Simulation of Adaptive Behavior. MIT Press, 1991.

[18] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, Englewood Cliffs, NJ, 1991.