Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

A SIMULATION TEST-BED TO EVALUATE MULTI-AGENT CONTROL OF MANUFACTURING SYSTEMS

Robert W. Brennan
William O

Department of Mechanical and Manufacturing Engineering
University of Calgary
2500 University Drive N.W.
Calgary, AB T2N 1N4, CANADA

ABSTRACT

Current research in the area of manufacturing planning and control has moved away from traditional centralized solutions towards distributed architectures that range from hierarchical to heterarchical. Between these two extremes of the control architecture spectrum lies the holonic manufacturing systems paradigm, where partial dynamic hierarchies of agents cooperate to meet global system objectives in the face of disturbances. This paper describes a simulation test bed for the evaluation of a distributed multi-agent control architecture for holonic manufacturing systems that integrates discrete-event simulation software into its design to allow the control architecture to be evaluated with a variety of emulated manufacturing systems.

1 INTRODUCTION

To meet the requirements of agile manufacturing, various distributed control architectures have been proposed that span a spectrum from hierarchical to non-hierarchical (or heterarchical) control architectures (Dilts et al. 1991). These various architectures are intended to enhance a control system's adaptability and flexibility against disturbances such as machine failure or uncertain processing times.

At the heterarchical end of the control architecture spectrum, the most commonly used distributed scheduling and control approach is the contract-net (Smith 1982) "auction-based bidding" protocol to allocate manufacturing resources to jobs.

With this approach, when a job arrives, it will request machines in the system to submit bids for its first operation. Upon receiving the job's request, machines that can perform the operation will evaluate their task agenda, then reply the job with a message containing information like the earliest time they can start/finish the operation, and/or the number of jobs already reserved the usage of the machines. The job will then evaluate all the responses based on some criteria and choose a machine to reward the operation to it. The job will confirm with the selected machine about the reservation, so that the machine can allocate a time slot in its task agenda for the job. The job will repeat the aforementioned procedures to find a machine for its remaining operations.

Due to the fact that in a heterarchical control system, agents use purely localized information and all forms of hierarchy are eliminated, heterarchical control results in problems with global optimization and predictability of system behaviors. In an attempt to combine the best features of hierarchical ("top down") and heterarchical ("bottom up", "cooperative") control structures, some researchers (Bongaerts et al. 1998, Brennan and Norrie 1999, Brennan 2000, Van Brussel et al. 1998, Zhang and Norrie 1999) have proposed the Holonic Manufacturing concept to preserve the stability of hierarchy while providing the dynamic flexibility of heterarchies.

Valckenaers et al. (1997) have defined the Holonic Manufacturing System (HMS) as "system components of autonomous modules and their distributed control. A holonic manufacturing architecture shall enable easy self-configuration, easy extension and modification of the system, and allow more flexibility and a larger decision space for higher control level." (Van Brussel et al. 1998)

The following list of definitions are developed by the HMS consortium to help understand and guide the translation of holonic concepts into a manufacturing setting (Van Brussel et al. 1998):

- Holon: An autonomous and co-operative building block of a manufacturing system for transformation, transporting, storing and/or validating information and physical objects. The holon consists of an information processing part and often a physical processing part. A holon can be of another holon.

- Autonomy: The capability of an entity to create and control the execution of its own plans and/or strategies.
- Co-operation: A process whereby a set of entities develops mutually acceptable plans and executes these plans.
- Holarchy: A system of holons that can co-operate to achieve a goal or objective. The holarchy defines the basic rules for co-operation of the holons and thereby limits their autonomy.

Although numerous researchers have investigated distributed (multi-agent) or holonic control systems, most studies are based on architectural issues, and few have investigated how these modular control entities (agents or holons) can be built, distributed (across a network) and integrated into a production control system.

In this paper, we use a distributed control approach based on holonic concepts to build a shop floor control system, and simulate the (distributed) control and production processes.

In the next section, the design of the distributed control system is discussed. In Section 3, a description of the implementation of the experimental model is given which includes a description of the test production system, the distributed control system interface with the discrete-event simulation model and an example of the distributed control software. Finally, results and conclusions drawn from the authors' work with the test bed are presented in Section 4 and future plans with this model are discussed in Section 5.

2 DISTRIBUTED CONTROL SYSTEM DESIGN

The experimental manufacturing system used for this research is developed using the Arena discrete-event simulation package (Kelton et al. 1998, Pegden et al. 1995). This system is integrated with a distributed multi-agent system for shop-floor scheduling and control developed in C++ (Ellis and Stroustrup 1990) that utilizes Component Object Model (COM) and Distributed-COM (DCOM) technology (Li and Economopoulos 1998).

The discrete-event simulation model contains a number of manufacturing resources, which include workstations and machines as well as a number of jobs to be processed by the system. In this section we describe the multi-agent system that is used to control this emulated manufacturing system. First, we describe the various agents that make up the control system, then we describe the methodology used to distribute these agents across multiple computers.

2.1 The Control Agents

One of the goals of this research is to develop a multi-agent control system that shares many of the characteristics of a holonic manufacturing system described previously. As a result, the four basic agents used in the control system reported here each correspond closely to the basic holons that are used in the PROSA architecture developed by Van Brussel et al. (1998):

- job agent,
- station agent,
- machine agent, and
- mediator agent.

Each job in the discrete-event simulation model is represented by a job agent, which is responsible for initiating the auction-based bidding process to find the resources for the job's operations, and monitoring the job's production progress.

Since a workstation can contain a number of homogeneous machines, a station agent's responsibilities are to assign tasks to the machines it manages, to monitor the production progress of the machines and to response to the job agent's bidding request.

It was pointed out in (Dilts et al. 1991) that the functional limitations of some commercially available low-level controllers can prevent the application of intelligent subordinate. Therefore in our experimental model, we define two types of machine agents, namely machSimp (the simple machine) agent and machIntel (the intelligent machine) agent.

As was discussed in the previous section, in order to carry out the resource bidding process, each resource must have the capability to respond to a job agent's bidding request. The machIntel agent represents the machine with the controller that has the information processing and communication capability to participate in a bidding process, and bears similar responsibilities as a station agent. The machSimp agent represents the machine with a controller that can only perform simple operation recording duties. As we will see in the later, the machSimp agents are usually aggregated with the station agent to form a workstation.

The mediator agent is similar to the Yellow Page agent defined by Shen et al. (1999) or the staff holon defined by Van Brussel et al. (1998). It is responsible for registering the manufacturing resources in the system, and responding to the job agent's query regarding which resource in the system can perform a particular type of operation.

2.2 Representing the Agents as COM Objects

Referring to the holon definition stated previously, a holon consists of an information processing part and often a physical processing part. In our experimental model, the information processing part of a holon is represented by a COM object, and the physical part is represented by the corresponding entity in the simulated production system in Arena. The COM diagram for the 5 agents mentioned above are shown in Figures 1-5.

Component Object Model (COM) is a platform-independent, distributed object-oriented standard for creating binary objects that can interact.

The essence of COM is an agreed binary interface that based on the Remote Procedure Call (RPC) technology with some wrappers that form the concept of objects and interfaces between the objects.

Conventionally, the interface on the top of Figures 1-5 represents the IUnknown interface, which is the base interface inherited by all other COM interfaces. The IUnknown interface provides three functions (methods), namely AddRef(), Release() and QueryInterface(). AddRef() and Release() are reference counting mechanisms for COM objects to manage their lifetimes.

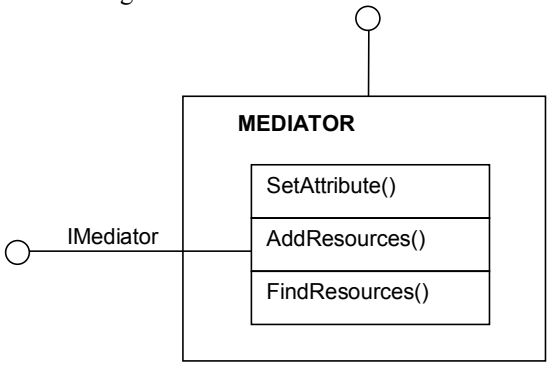

Figure 1: The Mediator COM Diagram

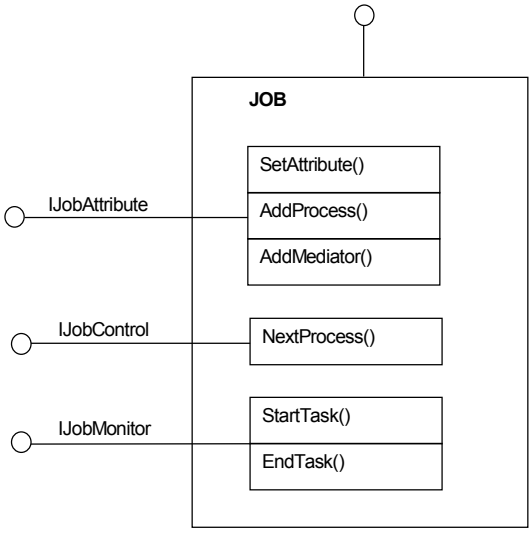

Figure 2: The Job COM Diagram

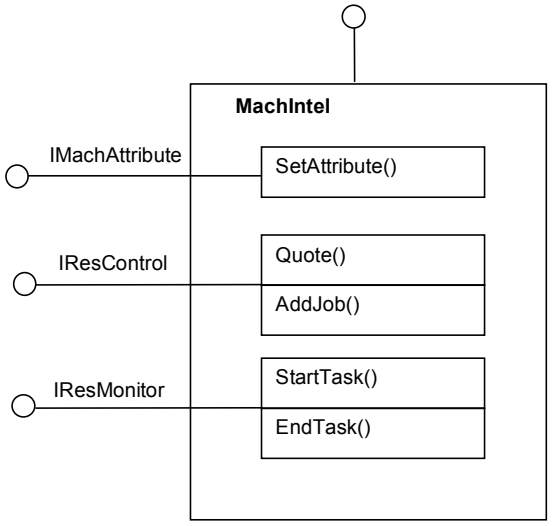

Figure 3: The MachIntel COM Diagram

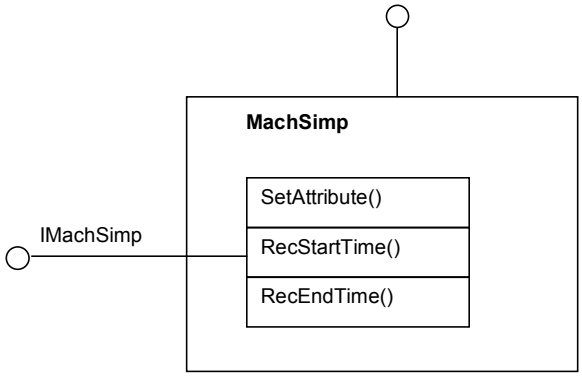

Figure 4: The MachSimp COM Diagram

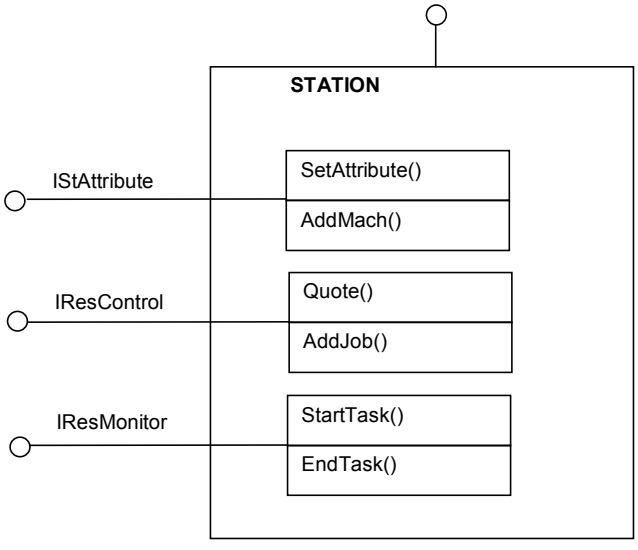

Figure 5: The Station COM Diagram

Each COM object has an internal counter that holds the number of users referencing the component. As suggested by its name, QueryInterface() is used by a client to query if a COM server supports a particular interface. If it does, a pointer to the required interface will be returned to the client.

Since all COM interfaces are based on IUnknown, they must also implement the AddRef(), Release() and QueryInterface() methods. Therefore, given any interface pointer to an COM object, a client should also be able to obtain any other interface supported by the object by calling QueryInterface() on the existing interface pointer.

DCOM (Distributed COM) extends COM so that COM clients and servers can all run on a single machine or distributed across a wide area network.

3 EXPERIMENTAL MODEL IMPLEMENTATION

Each agent in the distributed control system (except the mediator agent) represents the controller of a corresponding manufacturing entity in the Arena model. In this section, we will present an example to demonstrate the interaction model of the agents and the production processes.

In our example, the production system, shown in Figure 6, contains the following resources and job types:

1) a workstation (Station 100) contains 2 machines (Mach 10 and Mach 20 of MachSimp type) and can provide the drilling operation,
2) a single machine (Mach 200 of MachIntel type) that can perform the milling operation,
3) a single machine (Mach 300 of MachIntel type) that can perform the cutting operation, and
4) three job types (each job has 2 with no precedence constraints).

3.1 The Simulation Interface

At the beginning of the simulation, a mediator COM object is created. Next, a station and two machIntel COM objects are created. The attributes of the station and the machIntel objects (such as resource number, function type) are set via the SetAttribute method. As one can see for the station object, there is an AddMach method in its IStAttribute interface, this is for creating and initializing the (MachSimp) machines that it contains.

The instantiated station and machIntel objects register with the mediator via the AddResources method of its IMediator interface, so that the mediator will know what resources are available in the system, and what function each resource can offer.

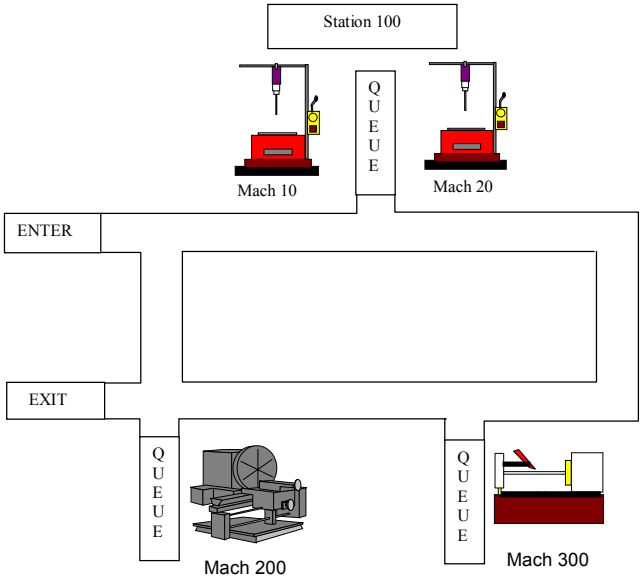

Figure 6: The Production Plant Layout

A job COM object is created for each of the jobs introduced into the system. Since a job has to contact the mediator to query about the resource that can perform its operations, a job is informed about the existence of the mediator via the AddMediator method of its IJobAttribute interface.

After the jobs and the manufacturing resources are instantiated, each of the jobs will start finding the resources for their operations. From hereon, we will regard the above-mentioned COM objects as agents. The resource reservation bidding processes are as follows:

1) To find a resource for its next operation, the job agent will ask the mediator agent (via the FindResource method of its IMediator interface) which resource can do the selected operation type.
2) The mediator agent answers the job agent with the corresponding resource address. The job agent then contacts the resource (station or machIntel agent) for a quote (when can it start the operation, how many queuing jobs are there now).
3) Since the processing sequence is not important for a job, a job agent will try to do an operation that can start on a resource earliest. Therefore, the job agent repeats steps 1 and 2 for all remaining operations, then select an operation with the resource that has the best quote (can process the job earliest).
4) The job agent contacts the selected resource to add itself to the resource's reservation list.
5) The job moves to the selected resource's location.

Referring to Figures 3 and 5, we can see that each station and machIntel COM object has to support an

IResControl interface which provides the Quote and AddJob methods for a job COM object to request for a quote and confirm the resource reservation, respectively.

When the mediator agent answers the job agent with the address of the resource, the job agent doesn't need to know what the type of the resource is. It will contact the resource through the same method (with the same parameters) of the same interface (IResControl). This provides the robustness for using different types of resource controllers. As long as the controllers support the IResControl interface, how they implement the Quote and AddJob methods is irrelevant.

When it's time for a machine to start processing a job in the simulated production system, the corresponding station/machIntel agent will be notified. The station/machIntel agent will then notifies the job agent via its IJobMonitor interface about the start of the operation (so that a job agent can keep track of its production progress).

For a machIntel agent, it will then record the start time of the operation (for statistics collection). After contacting the job agent, the station agent delegates the operation recording duty to its selected, contained machine (machSimp) agent.

Since the (machSimp) machine agent (or controller) has the capability to record the operation time, it is reasonable for the station agent to delegate this task to its contained machSimp agent (each machine contained in a workstation is represented by a machSimp agent) via the RecStartTime method of its IMachSimp interface.

The same procedures are carried out when a machine finishes an operation in the production system. Once again, one can see that both the station and machIntel objects have to support the IResMonitor interface, so that when the Arena application notifies the station/machIntel agent about the start/end operation event, it doesn't need to know what type of resource it is communicating with, even though the station and machIntel agents implement the StartTask/EndTask methods in different ways.

3.2 Distributing the Control Agents

In the previous section, we have shown how the different agents can interact with each other to carry out the control of the production processes. After developing the COM objects, we distribute them over the network, and have them interact with each other as described above to simulate the communication and co-operation of the actual controllers distributed in a production plant.

Figure 7 shows the layout of our networking model. In our model, the Arena application was run on the same computer as the mediator, job and mach 200 agents. The mach 300 and station 100 agents were distributed to another computer that was connected to the Arena computer.

The production simulation worked in the same way as described previously. To monitor the status of the agents, we have each agent log all its activities in a local database.

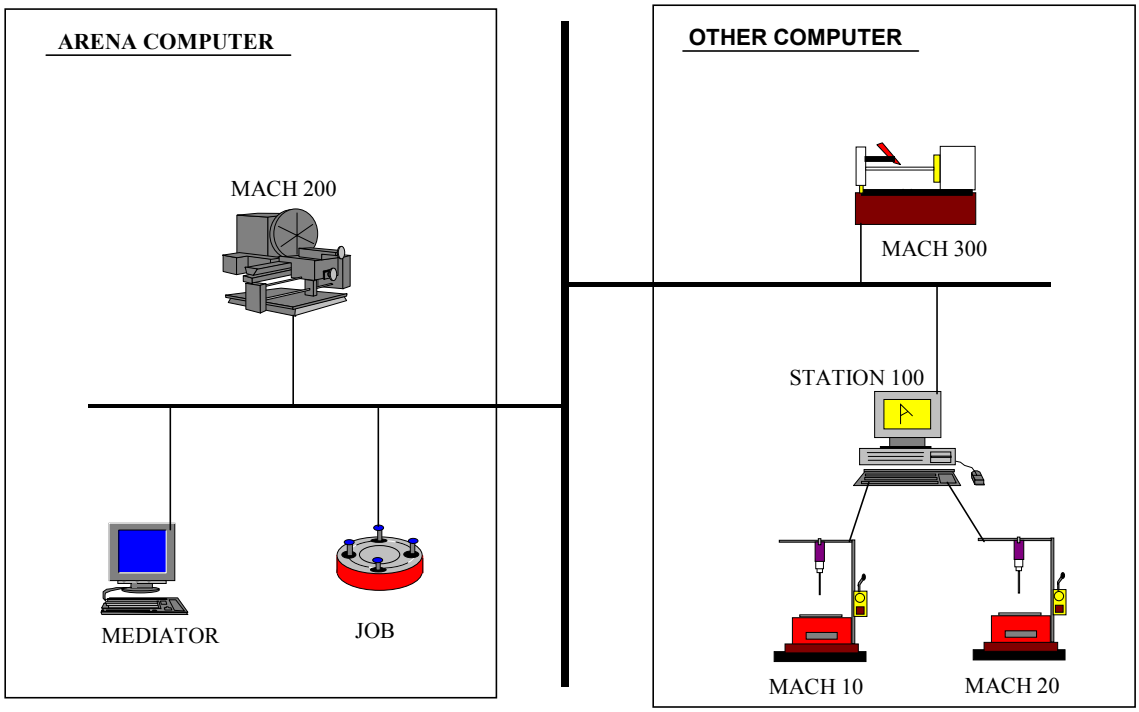

Figure 7: The Layout for the Networking Model

Since the job, machIntel, and machSimp agents all keep records of the operation start/end times, we have each of the agents record the times in a local file (local database). This local data provides a channel for someone (such as centralized staff controller) to check on the status of these agents at any instant of time and at any location by viewing the data through a browser.

For example, to view the status of the mach 300 and station 100 agents from the Arena computer, we launch an internet browser to view what resources are running on the "other computer" as shown in Figure 8. Then to view the status of the station 100, we just choose the WStation item.

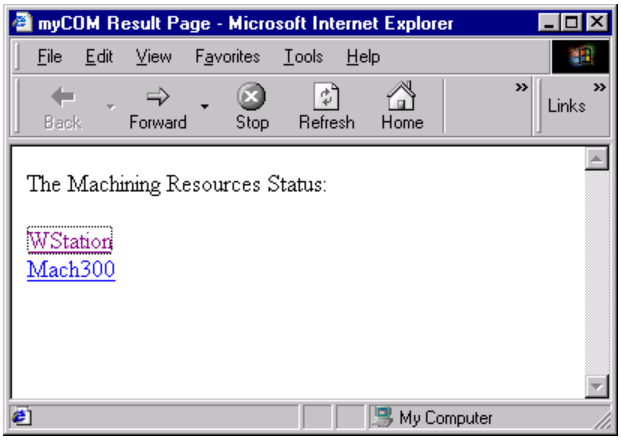

Figure 8: The Manufacturing Resources on a Network Computer

Figure 9 shows the status of the station 100 at time 0. As can in this figure, at time 0, job 4 first joined the station, and the machines of the station were idle at that time (JX indicates no job is loaded on the machine). Then job 4 was loaded to mach 10 and job 3 arrived. Then job 3 was loaded to mach 20 and job 1 joined the station. Since no machine was available at that time, job 1 stayed in the queue. In our example, two jobs of types A, B and C were created.

4 PERFORMANCE RESULTS

Current research in multi-agent heterarchical control systems usually implement "part driven" real-time scheduling algorithms, where part agents use an auction-bidding resource reservation protocol to explore the routing or process sequencing flexibility in real-time (Parunak 1987, Duffie and Prabhu 1994). Alternatively, traditional dispatching control systems usually implement "resource driven" scheduling (dispatching) algorithms, where the resource controllers (agents) use dispatching rules to sequence the processing of the arriving jobs, and the routing decisions are usually determined in advance. Although quantitative results are available for bidding-

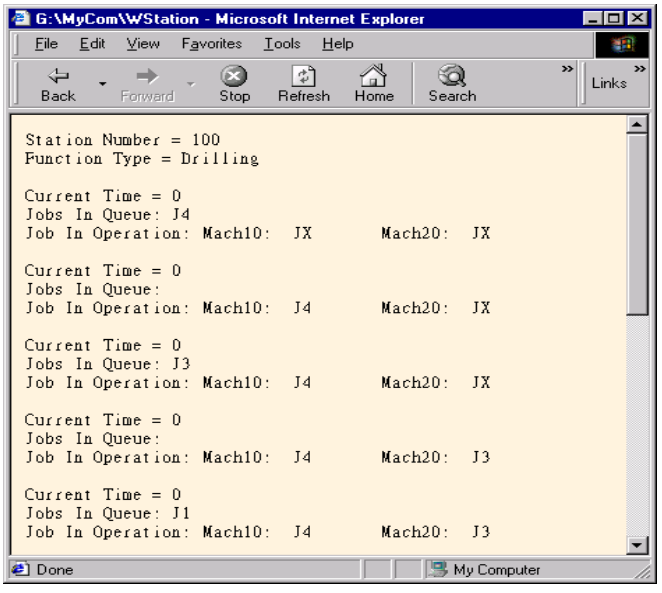

Figure 9: The Status of Station 100 at Time 0

based scheduling and there is a wealth of literature on scheduling heuristics and dispatching rules, few researchers have compared the performance of these alternative approaches on a common platform.

In this section we present an example of some of the performance analysis that has resulted from the model described in this paper to investigate the impact of the dynamic job routing and job sequencing decisions on the control system's performance and adaptability against disturbances. The tested control systems will have varying production volumes (to model the production system with looser/tighter schedules) and disturbance frequencies, so that the impact of the job routing and sequencing decisions in various manufacturing environments can be evaluated. In our experimental models, routing flexibility is introduced into the production system by providing jobs with a flexible processing order for their operations. That is, there is no technological constraint on the processing sequence of the operations of the jobs.

4.1 Experimental Models

The shop floor layout chosen for the experiments is the generic machining system proposed by Cavalieri et al. (1999) to serve as a common benchmark platform for comparing multi-agent control systems. The production system consists of four types of machines, and two machines per type are present. Although it is proposed that the transport system is modeled as a set of serial transporters (AGVs), these AGVs are assumed always available and transport times are set equal to zero. Therefore for these experiments, the transportation entities and transport times are not modeled in order to simplify the system.

Two types of job shop problem are proposed by Cavalieri et al. (1999). For the first problem, products have fixed process plan constituted by four non-preemptive operations (one for each machine type), and the third machine to be visited is the bottleneck resource (last-longing operation). The second problem is similar to the first one, except that in this problem, routing flexibility is introduced into the system. That is, products have a flexible processing order of their operations (a bottleneck resource is still present, but not necessary the third one to be visited). This second problem is used for the experiments reported in this paper. More detailed descriptions of the production system and performance measures are covered in (Cavalieri et al. 1999).

To evaluate the impact of dynamic job routing and job sequencing decisions in various manufacturing environments, the following control strategies are implemented in our experimental models:

(a) AUC_BID (AUCtion-BIDding) - In this control approach, the job control agents use the contract net auction-bidding protocol to collect bids from the workstations to explore the process sequencing/routing flexibility. Job sequencing will not be implemented in this control approach. That is, to decide which operation to process next, for each of the job's remaining unprocessed operations, the job agent will contact the system mediator to find out which workstation is responsible for that type of operation. Then the job agent will contact the corresponding workstation agent to see when the workstation can start the operation. Since job sequencing is not implemented, the workstation agent will rank the incoming jobs on the First-Come-First-Serve (FCFS) basis, and respond to the job agent with the answer that states the earliest possible start time for that operation. After receiving responses from all the workstations that can process its remaining unprocessed operations, the job agent will evaluate all the responses and pick the operation whose corresponding workstation can start the job soonest to be processed next.

(b) JSEQ (Job SEQuencing) – In this control approach, the workstation control agents use the adopted priority dispatching rule to sequence the incoming jobs, and the jobs do not explore the routing flexibility. That is, even though there is no technological constraint for the operations of the jobs, the job agents will not explore the routing flexibility, and will have their operations processed in some predetermined order (the order that is originally stated in their process plan). When a job enters a workstation, the workstation agent will rank the incoming jobs based on some adopted priority rules. In our experiments, the Least Work Remaining (LWKR) heuristic priority dispatching rule will be used. This is because the performance measure of our experiments is the minimization of the mean flow time, and the empirical experimental results conducted by other researchers (Conway et al. 1967) have suggested that the LWKR rule can help minimize the mean flow time.

(c) AUC + JSEQ – In this control approach, while the job agents will use the auction bidding mechanism as stated in (a) to explore the routing flexibility, the workstation agents will sequence the incoming jobs based on the dispatching rules as stated in (b). That is, to decide which operation to process next, the job agents will collect bids from the workstations that correspond to its remaining unprocessed operations. Unlike in (a), when a workstation agent receives a bid request from a job agent, instead of quoting the job's earliest possible start time based on the First-Come-First-Serve rule, the workstation agent will try to insert the job into its queue and quote the job with the earliest possible start time that is based on the adopted priority dispatching rule as stated in (b). After receiving the response from all the workstations that correspond to its remaining unprocessed operations, the job agent will evaluate all the responses and pick the operation whose corresponding workstation can start the job soonest to be processed next.

(d) COMT+AUC+JSEQ (COMmitmenT + AUC + JSEQ) – One of the problems regarding the control approach stated in (c) is the role of commitment in the auction-bidding processes. In deciding which operation to be processed next, the job agent will make the decision based on the returned "earliest start time" quote of the workstations that correspond to its remaining unprocessed operations. The returned quoted start time represents that the workstation is willing to commit some of its resource capacities to process the job at certain times. But when the workstation agents use the LWKR rule to sequence the incoming jobs, the workstation agents might violate some of the previous commitments that it has made to some jobs. In this control approach, when the workstation agents insert a new job into its queue, the affected jobs will be notified so that they can explore other routing opportunities. For the affected job agents, if no other workstations can start their other remaining operations sooner, then they will decide to stay in the original workstation. Otherwise, they will change the workstation (and the process sequence).

The process plan for the job types that will be used in the experimental models is shown in Table 1 below. The notation used for operations in this table is: process time (in minutes) / operation type.

Table 1: Process Plan of the Various Job Types

	Operation (proc. time (min)/operation type)			
Job ID	1	2	3	4
J1	6/1	8/2	13/3	5/4
J2	4/1	3/2	8/3	3/4
J3	3/4	6/2	15/1	4/3
J4	5/2	6/1	13/3	4/4
J5	5/1	3/2	8/4	4/3

4.2 Experimental Results

4.2.1 No Machine Failures

In this experiment, we evaluate the performance of the four control strategies described in §4.1 in control systems with no disturbances. Each of the four control strategies is implemented in control systems with varying production volumes, so that the impact of the alternative control approaches in control systems with various degree of tightness of schedules can be evaluated. In each test, equal amounts of each of the job types described in Table 1 above will be produced. The results of the tests are shown in Figure 10.

4.2.2 Machine Failures

In this experiment, we increase the machine failure disturbances by having both the machines in workstation WS1 down for 30 minutes after they have done their first operation, and both machines in workstation WS2 down for 30 minutes after the production has run for half an hour. It should be noted that since the operations of the jobs are non-preemptable, when the scheduled downtime is reached and a machine is operating on a job, the failure of the machine would be initiated after the job is finished. The results of the tests are shown in Figure 11.

4.3 Discussion

As can be seen in Figure 10, the control strategies that incorporate the MWKR dispatching rule (i.e., incorporate JSEQ) outperform the auction-bidding approach using FCFS (i.e., AUC_BID). This result is most evident when the dispatching rule is used in combination with auction-bidding and when congestion increases.

Figure 11 illustrates how the opportunistic behaviors of the job agents can help them avoid the bottleneck workstation (workstation with down machines) while making the routing decisions. As can be seen, the AUC_BID policy can help improve the system performance, but only in manufacturing environments with production volumes under certain limits. This is because as the manufacturing system's production volume increases, each workstation will be occupied by more jobs at any instant of time, and thus a job agent will have lesser chance to find an alternative workstation that can start processing its other operation sooner. As resulted, in control systems with high system congestion, even when there are disturbances, job sequencing can better improve the control systems performance than the routing flexibility control mechanism.

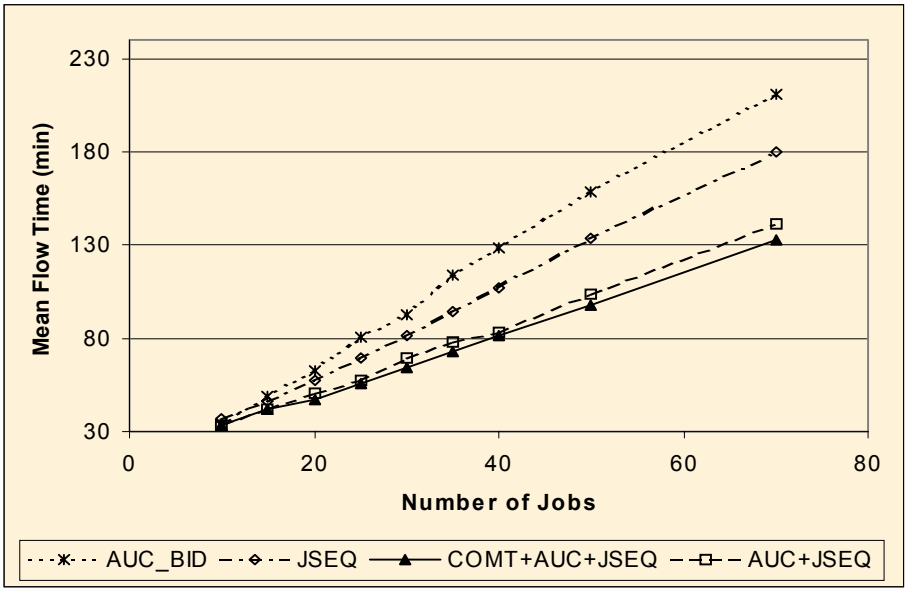

Figure 10: Mean Flow Time Performance for the No Machine Failures Scenario

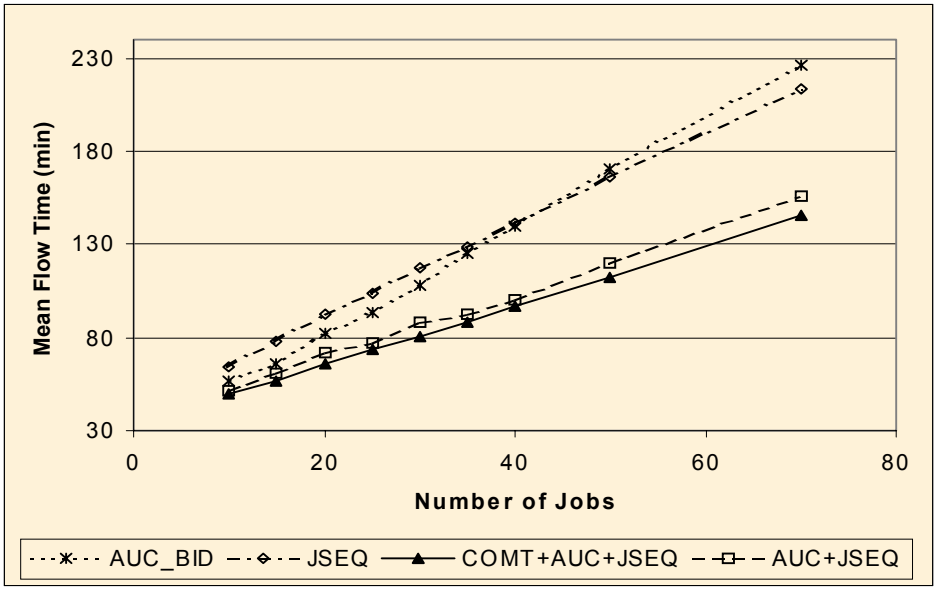

Figure 11: Mean Flow Time Performance for the Machine Failures Scenario

5 CONCLUSION

In this paper, we have discussed how to develop the different control roles (or agents) into COM modules (objects) that can be easily distributed over a network of computers. As one can see in the previous sections, it doesn't matter who takes what role. But for a controller (agent) to take a particular role, the controller must have the capability to fulfill the responsibilities of the role.

This provides for flexibility and robustness. For example, by having various controllers (servers) that support the same interface, other controllers (clients) can communicate with these controllers via the same interface without having to differentiate their types. As well, different controllers can implement the responsibilities in different ways.

It is also relatively easy to modify an entity's (controller's) responsibilities by having it support or not support certain interfaces. The station and machSimp objects demonstrate this software reusable advantage wherein, we can create an object that uses some of the functionality of an existing object without duplicating that functionality in the new object.

With the help of object-oriented analysis and design technique (Larman 1997), we can identify the roles and responsibilities in a manufacturing control system, and then assign the roles to the entities that have the capabilities to fulfill the corresponding responsibilities. These (control) entities can be easily developed into COM objects, which can then be distributed to work with whatever applications that need them (or distributed to other researchers that might need to use or test the objects). "Rather than write large monolithic object-oriented applications, you can write applications as small independent components that can slot together to make a complete application. With a little extra work, your C++ objects can become COM objects. As COM objects, they are not as tightly tied to one running process or computer as a conventional C++ object would be." (Bates 1999)

The other advantage of developing the manufacturing (control) entities as COM objects is that some large industrial vendors such as GE Industrial System and Sisco, Inc. already have the automation and control products that support the COM/DCOM technology. Therefore, by using the COM/DCOM approach, we can close the gap between the academic field and the manufacturing industry, and can also minimize the times and facilitate the process for shifting from the design phase to the implementation.

Our current research is concerned with evaluating the performance of the multi-agent control approach described in this paper. In future studies, the modular nature of the test bed design will allow us to evaluate this control approach with various manufacturing systems, emulated in Arena, as well as investigate the interactions between the agents that comprise the control model.

ACKNOWLEDGMENTS

The authors wish to thank the Natural Sciences and Engineering Research Council of Canada for their generous support of this research under grant OGP-019-7339.

REFERENCES

Bates, J., 1999. *Creating lightweight components with ATL*. Indianapolis: SAMS Publishing.

Bongaerts, L., L. Monostori, D. McFarlane, and B. Kadar. 1998. Hierarchy in distributed shop floor control. In *the Proceedings of the 1st International Workshop on Intelligent Manufacturing*.

Brennan, R. W. 2000. Performance comparison and analysis of reactive and planning-based control architectures for manufacturing. *Robotics and Computer Integrated Manufacturing* 16(2-3): 191-200.

Brennan, R. W. and D.H. Norrie. 1999. The performance of partial dynamic hierarchies for manufacturing. In *the Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems*, ed. H. Van Brussel and P. Valckenaers, 5-13.

Cavalieri, S., L. Bongaerts, M. Macchi, M. Taisch, J. Wyns. 1999. A benchmark framework for manufacturing control. In *the Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems*, ed. H. Van Brussel and P. Valckenaers, 225-236.

Conway, R. W., W. L. Maxwell, and L. W. Miller. 1967. *Theory of Scheduling*, Addison-Wesley.

Dilts, D. M., N. P. Boyd, and H. H. Whorms. 1991. The evolution of control architectures for automated manufacturing systems. *Journal of Manufacturing Systems* 10(1): 79-93.

Duffie, N. A. and V. V. Prabhu, 1994. Real-time distributed scheduling of heterarchical manufacturing systems. *Journal of Manufacturing Systems* 13(2): 94-107.

Ellis, M. and B. Stroustrup. 1990. *The annotated C++ reference manual*. Reading: Addison-Wesley.

Kelton, W. D., R. P. Sadowski, and D. A. Sadowski. 1998. *Simulation with arena*. New York: McGraw-Hill.

Larman, G. 1997. *Applying UML and patterns, an introduction to object-oriented analysis and design.* Upper Saddle River: Prentice Hall.

Li, S. and P. Economopoulos. 1998. *Professional COM applications with ATL*. Wrox Press.

Parunak, H. V. D. 1987. Manufacturing experience with the contract net. In *Distributed Artificially Intelligence*, ed. M. N. Huhns, 285-310.

Pegden, C. D., R. E. Shannon, and R. P. Sadowski. 1995. *Introduction to simulation using SIMAN.* New York: McGraw-Hill.

Shen, W., D. H. Norrie, and R. Kremer. 1999. Developing intelligent manufacturing systems using collaborative agents. In *the Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems*, ed. H. Van Brussel and P. Valckenaers, 157-166.

Smith, R. G. 1982. The contract net protocol: high-level communication and control in a distributed problem solver. Defence Research Establishment Atlantic D.R.E.A. Report 80/1. Dartmouth, Nova Scotia.

Valckenaers, P., H. Van Brussel, L. Bongaerts, J. Wyns. 1997. Holonic manufacturing systems. *Integrated Computer Aided Engineering* 4: 191-201.

Van Brussel, H., J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. 1998. Reference architecture for holonic manufacturing systems: PROSA. *Computers in Industry* 37: 255-274.

Zhang, X. and D. H. Norrie. 1999. Holonic control at the production and controller levels. In *the Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems*, ed. H. Van Brussel and P. Valckenaers, 215-224.

AUTHOR BIOGRAPHIES

ROBERT W. BRENNAN is an Assistant Professor in the Department of Mechanical and Manufacturing Engineering at the University of Calgary. He holds B.Sc. and Ph.D. degrees from the University of Calgary. He is a member of INFORMS, IIE, and SME. His interests manufacturing systems control, holonic manufacturing, and project management. His email and web addresses are <brennan@enme.ucalgary.ca> and <isg.enme.ucalgary.ca>.

WILLIAM O is an M.Sc. candidate in the Department of Mechanical and Manufacturing Engineering at the University of Calgary. His interests object-oriented analysis and design and holonic manufacturing. His email and web addresses are <wio@ucalgary.ca> and <isg.enme.ucalgary.ca>.