Sādhanā, Vol. 23, Part 4, August 1998, pp. 359–376. © Printed in India.

An agent-based information architecture for shop-floor control

N VISWANADHAM and A LAKSHMINARAYANAN

Mechanical & Production Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260
e-mail: mpenv@leonis.nus.edu.sg

MS received 5 January 1998; revised 6 March 1998

Abstract. The principal aim of this paper is to develop a highly flexible shop-floor control system architecture using software agents which communicate with one another using the peer-to-peer paradigm. A shop-floor control system is responsible for the coordination of material and information flow in the manufacturing system as well as for communicating with the suppliers and the distributors or customers. It has to make several dynamic decisions like release of parts into the system, scheduling and routing of parts and transportation, machine selection, rerouting in case of failures etc. In our proposed control system such decisions are made by *software agents* which are equipped with local databases and independent computing power and a reliable communication interface. The different agencies and agents that make up the control system are discussed in this paper. In this proposed system, a part agent (as part of the part agency) enters the factory floor with process requirements, contacts the task assignment agent for the process plan, the monitoring agent for the status of the system, the scheduling agent for the detailed schedule, and the router agent for transportation before finally proceeding to get served. Mechanisms for handling issues like scheduling, routing, and failures have been discussed. Java and its underlying programming concepts form the backbone of our agent architecture. Agents use KQML (knowledge query and manipulative language) constructs for sending queries and KIF (knowledge interchange format) constructs for representing knowledge and exchanging information. Our system has several advantages over other control architectures and it is highly likely that future factory floor control systems will follow similar paradigms.

Keywords. Agent-based information architecture; shop-floor control; peer-to-peer paradigm.

1. Introduction

Currently the manufacturing industry is facing a tremendous amount of change. The dramatic increase in international competition, emergence of demanding customers, and rapid changes in technologies are prompting organizations to redesign their manufacturing facilities. Also, companies want to deliver value to the customers by packaging the product with value-added services and delivering them faster than competition. The goal of a manufacturing system is to attain faster lead times, reliable deliveries, low cost and high quality for a wide variety of products. These are important for both current products as well as new products. With shrinking product life cycles, it is important that factory floors deliver new products, in fact generations of them, is a short span of time. More specifically, it is important that the manufacturing system responds to design and demand changes of current products quickly and supplies all the combinations and options of current products desired by the customer. Also, new products should be designed, prototyped and manufactured in a short time. In this paper, we are concerned about the control system architecture that enables the factory floor to produce rapidly, a variety of current products and new products.

Factory floors get expanded by adding new machines or contracted by removing old machines. Also, factory floor technologies change and new techniques of factory floor management come into practice. It is important that the control system be adaptive and incorporate these changes. In other words, the control system should be flexible to cope with several environmental and technological changes.

The emergence of the Internet and the possibility of secure communications over it, has created new opportunities and also put new demands on the factory floor control system. It has created opportunities for telemanufacturing, for downloading part programs and route maps from a remote host as and when required, and for monitoring the progress of their order on the internet by the customers themselves. The use of EDI and barcoding have made it possible to monitor the WIP of components and sub-assemblies by suppliers and to use this information to schedule their factory floors. The Internet also provides for sharing information on CAD drawings, test data, lists of qualified vendors, consultants etc. by the suppliers. An Internet compatible factory floor control system should have several capabilities: the ability to interact with members whose IT sophistication varies enormously from a lone PC to an array of state-of-the-art CAD/CAM workstations, securely linking and delinking partners, ability to handle transfer of large files such as CAD drawings to an EDI message from a supplier or transmission of WIP information to a supplier, and provide access to partners to its database. As described by Upton & McAfee (1996), such a control system would make the factory floor the real virtual factory.

In general, the manufacturing system should cope with uncertainties such as:

(i) *Resource changes*: Variations in the number of human and machine resources on the factory floor due to machine failures, absenteeism, transport breakdown, supplier unreliability etc. are issues that arise routinely.

(ii) *Design and demand changes in the product*: Customer demand is random and this combined with inaccurate forecasting will cause uncertainty in the design and mix of products. Also, proactive introduction of new products will reflect as planned changes in demand.

(iii) *Technology changes*: These could be continuous or discontinuous. The discontinuous technology changes such as the ones that occur in the PC industry and the hard disk drives are difficult to cope with. The Internet is a new technology with tremendous potential. The company should have the ability to predict and develop competences in new technologies for production of future product generations.

(iv) *Regulations*: The green manufacturing revolution has an impact on the choice of materials and technologies used on the factory floor. Also, socio-political changes such as deregulation of telecommunications, airlines, information networks, liberalization of certain closed economies, legislations on health care etc. had tremendous impact on the corresponding industries. This also results in changes that the factory floor has to cope with.

1.1 *Coping with uncertainty*

We see that as we traverse from resource changes to design and demand changes in regulations, both the magnitude and the effect of change will increase. They can be classified as operational, tactical and strategic changes and the flexibility strategies could be correspondingly named. Resource changes occur daily and appropriate procedures have to be designed and built into the system. Events such as failure of the machine or a truck or a rush order occur at random times and places. Procedures, like in hospitals and civil defence, should be evolved for all frequently occurring events. Design and demand changes occur, say, monthly and have to be met through proper scheduling of orders. They are tactical in nature. Technology and regulatory changes are sporadic but occur in a predictable way for companies with learning capabilities. Some companies see the opportunity and lead the change in a dynamic way. They are strategic in nature and involve proactive strategies for product development, technology adaptation, etc. Flexibilities are to be built into the system to cope with these changes. These abilities are built into the system via information technology tools such as databases, internet, expert systems etc., and procedures, and control mechanisms such as scheduling, information processing, order tracking etc.

Any type of ability to cope with changes comes from two sources: Assets such as flexible machines, material handling equipment, assembly stations, IT infrastructure and organization and co-ordination of these assets to gain competitive advantage and attain the goals of delivering high quality products at low cost faster than competition. Here, we assume that we have assets that can be programmed to manufacture a wide variety of parts. We could like to design a control system that has the properties of modifiability and expandability, which provide a communication ability for entities with varying capabilities, and requirements to interact, which can accommodate new designs and jobs and tolerate failures in equipment and in the control system.

1.2 *The factory floor*

We are mainly focusing on integrated computer-controlled, flexible shop-floors which are characterized by

- versatile NC machine centres and assembly stations equipped with automatic tool changing and part load/unload capabilities;
- automated material handling equipment to move parts and tools;
- load/unload stations through which the jobs enter and exit the factory floor;
- automated inspection stations;
- storage facilities for raw and semi-finished workpieces under computer control;
- a communication network linking facilities to one another, to the control system and to the Internet;
- a control system which monitors, schedules and manages all the activities of the manufacturing system and communicates with the external world as well as with the equipment.

The most important features of a shop-floor control system are the decision making entities. Decision making in a manufacturing environment can be very complex because of the alternatives available and the uncertainties involved (Viswanadham & Narahari 1992). There are three levels of decisions in a manufacturing system: strategic, tactical, and operational. The strategic decisions are long term and determine the competitiveness and survivability of the firm. The second level decisions have a horizon of weeks/days. These include dividing the overall production target into batches, taking into account the availability of raw materials, tools and due dates, while at the same time maximizing the utilization of assets, and balancing the workload on the system. The operational level decisions are concerned with real time control of manufacturing operations. These include issues such as machine loading decisions, AGV scheduling decisions, which part type to be introduced into the system, how many different part types can coexist, monitoring of available machines, reconfiguration and rescheduling in case of failures. We also consider the maintenance of an information database which provides secure and selective information access to suppliers, customers etc. regarding the status of orders, status of equipment, status of WIP etc. Also it provides downloading of information such as part programs from network sites as and when needed. We address the operational level issues here and design a control system that can effectively make these decisions.

1.3 *Factory floor control system architectures*

Rapid technological advances in computing and communications have made it possible to consider a wide range of possibilities in the design of control architectures. There are four basic forms of architectures (Dilts *et al* 1991). The first one is the *Centralized architecture.* This is characterized by a "mainframe computer" that performs all planning and information processing functions and maintains global databases to record the activities of the whole system. The machine controllers execute commands received from the centralized control facility. This architecture has become obsolete because of poor fault-tolerance capability. Moreover modifications to the software as well as to the system are very difficult and expensive.

Drawbacks of centralized control led to the consideration of other control architectures like the *hierarchical* control system, which is the most popular one. There are two types – the proper hierarchical and the modified hierarchical. The modified hierarchical system is characterized by higher autonomy for system entities. Hierarchically controlled systems are constructed using a philosophy of "levels" of control and contain a number of control modules arranged in a pyramidal structure. Modules at each level make decisions based on commands received from the level above, and feedback received from the level below. However, studies have shown (Dilts *et al* 1991; Upton 1992) that the organization and structure of these systems get fixed in the early stages of decision-making and unforeseen modifications are very expensive (a clear case of long-term inflexibility). A module at one level requires substantial information about the levels below and above it, particularly when fault tolerance must be incorporated. This tends to make large hierarchical systems difficult and expensive to design, maintain and modify. Experience has shown that fault-tolerance is obtained in hierarchical systems with considerable expense and complexity (Dilts *et al* 1991).

To summarize, the main disadvantages (Upton 1992) of traditional manufacturing systems based on centralized or hierarchical control are as below.

Control software: The software for controlling medium/large automated manufacturing systems has to handle all the tasks involved like scheduling, routing and monitoring the state of the system. The complexity of the software necessary to provide short-term flexibility affects the introduction of new products.

Product change: If new products need to be introduced, it might involve introduction of new machines or reconfiguration of the system which might be difficult using the old architectures.

Failure and recovery as sources of complexity: Failures have to be anticipated and capabilities for fault tolerance have to be built in.

Technological advances in the area of distributed computing and the availability of cheap computers has popularised decentralized control architectures. The *heterarchical* control architecture derives the advantages through distributed intelligence, creating a flat architecture that divides control responsibilities among many cooperating entities. These models offer prospects for reduced complexity by localizing information and control, reduced software development costs by eliminating supervisory levels, higher maintainability and modifiability due to improved modularity and self-configurability. The logical structure and philosophy behind these systems does not resemble the traditional hierarchical management and production control structures. Instead, these entities cooperate through communication to pursue system goals. Information is stored in distributed databases and hence aggregation of information at one point is minimized in these systems as is the complexity of relationships between entities. It is generally accepted that increased autonomy of the participating components of a system reduces or eliminates the need for an intelligent centralized governing body. Such a system will be fault-tolerant and will not be adversely affected by breakdowns and repairs (which will have local impact only). Moreover introduction of new machines can be accommodated easily. Such a system will be modular with decentralized hardware and software and the software needed becomes easier to code resulting in fewer bugs and fewer unforeseen interactions between modules.

Various heterarchical models have been suggested (Duffie 1988; Shaw 1988; Lin & Solberg 1992; Upton 1992) in recent years. Studies have shown that such systems possess many advantages. But on closer examination, it was realized that implementation of such systems involve intelligent tasks like negotiation, bidding etc. which are still research areas. We realized that a hierarchical structure modified using software agent concepts gives the flexibility of heterarchical architecture while at the same time retaining the relative simplicity of the hierarchical structure.

1.4 *Outline of the paper*

In this paper, §2 deals with a brief explanation of the factory floor control problem. In §3, we outline the agencies and agents that make up our architecture. In §4, we descibe our control system agencies and agents and in §5, we discuss the broad advantages of this scheme and conclude with suggestions for further work.

2. The factory floor control problem

We now define the control problem. We consider a typical flexible factory floor equipped with computer-controlled equipment and connected to the outside world via the internet. This would enable any machine to exchange and request information and service from any other machine. Our description here makes transparent the distributed agent architecture for control. Customers from either inside the organization or outside (in case of an independent enterprise) request the *order processing system* for service to manufacture a batch or batches of parts with a delivery schedule. The order processing system returns to the customer with a price and delivery schedule. To do this, it has to determine if the part can be produced with the capabilities available, the waiting time etc. This has to be accomplished using decision support and human expertise.

We briefly describe the five steps involved in the order processing on the factory floor.

(1) *Accept orders*: Customers choose many ordering methods and most companies accept orders through traditional means like fax or mail, through sales representatives or through EDI. Orders can also be placed on the internet provided the necessary infrastructure is available. Customers can then track the order on the internet. The order information should be visible to all stake holders. One important issue in this step is order selection and prioritization. All orders need not be accepted and not all customers are equal. Some orders may require design of products (new to the factory floor) or the production of accessories such as fixtures and tools or, in some cases, process planning including part programming. In some cases the process plans and part programs available may have to be tuned to the factory floor control system. Here, we assume that part programs, tools, and fixtures are all available and consider the dynamical control of the factory floor.

In this step, we have four tasks.

(a) Order receipt;
(b) Order selection;
(c) Order confirmation involves the following,
- Check if the process plans, part programs, fixtures and tools are available.
- Check the current production schedules and estimate the probable date of marketing;

- Make a decision on how the items from the order are to be shipped to the customer;
- Estimate cost and time;
- Dialogue with the customer on cost and delivery date.

(d) Confirm the order, price and delivery date to the customer.

(2) *Configure order*: This consists of

(i) Identifying the batch of products and services that are contained in the order.
(ii) Synchronizing delivery of each of the components/subassemblies with the transport to and from the factory floor.

(3) *Schedule the order*: The order is scheduled on the factory floor using standard scheduling methods so as to take into account the due dates, availability of components, subassemblies, fixtures, tools etc. The static schedule is implemented but the work flow is dynamically managed to reduce effect of variabilities.

(4) *Work-flow management*: This step involves making plans for co-ordinated flow of the batch products through the factory floor. Once the static schedule is available, the flow of parts through the factory floor involves three activities: processing, moving and waiting for the resources. The flexibility of the factory floor provides for alternative paths of work flow and decision support is needed to exercise the choice in an intelligent way.

(5) *Order monitoring*: Like in project management, the progress of the order is monitored and actual progress is compared with the anticipated progress. The signals of completion or failure are sent to appropriate agencies which automatically monitor the progress. The customer may monitor the progress of the order through the order number (identification).

The above process can be automated for achieving rapid cycle times and quality delivery to the customer. We now describe below the factory floor controller to monitor, control, and track orders. This controller has adequate capabilities and is equipped with decision support tools.

2.1 *The factory floor controller*

We briefly describe the factory floor controller model in this section.

(1) The order arrives at the controller and is checked for configurability resource (fixtures, tools, machines, raw material) availability and delivery dead lines. The idea is to accept only high yielding, dependable deliveries.

(2) The controller decision support system has status information on the availability of the raw materials, part programs, tools and fixtures, and status of the resources and the schedules of assemblies and subassemblies. The order route plan is made and cost and time are estimated. The cost, delivery times and specifications are configured to the customers.

(3) If the order consists of items manufactured earlier, i.e. part programs, tools, fixtures are available and raw material suppliers are all identified, then the order can be immediately scheduled. On the other hand, if the customer supplies the part program, while the tools, fixtures and the raw materials are to be made ready, then the schedule starts after they are available and tested. If the order involves new product, then design, part program preparation and tooling are to be done before scheduling the orders on the shop floor.

In this paper, we assume the first case and consider the shop floor scheduling and control problem. The sequence of events when a part enters the factory floor are as follows: A decision is made as to which part enters the system and which AGV transports that part to the machine on which the first operation is done. There could be several parts waiting in queue at that machine and the part selected for processing could be based on any one of the familiar disciplines including FCFS, LPT, SPT and so on. Then the part is transported to another machine for next operation. The AGV scheduling is another decision-making issue. Also routing flexibility provides an opportunity for processing by several machines. Selection of the machine for the next operation is another decision to be made on the factory floor. The control system has to track the movement of parts and route them through the factory floor, using the equipment status information and following some heuristic decision rules.

In this paper, we describe an agent based factory floor controller for decision making and material flow control and tracking.

3. Architecture of the agent-based control system

The factory floor can be divided into different logical entities like parts, machine servers, AGVS, control systems etc. All the entities are autonomous and independent of each other, and are usually physically separated. In object-oriented parlance these entities can be thought of as objects with resources, which request services from other objects. This has been followed while designing this agent control system.

Agents are software programs designed to perform specific goal-driven functions and have the capability to communicate with each other using an agent communication language. According to a weak notion of an agent, agents should possess (Woolridge & Jennings 1994) the attributes listed below.

- *Autonomy*: Ability to operate on their own without external interventions.
- *Social ability*: Ability to interact with other agents.
- *Reactivity*: Ability to perceive their environment and react to it.
- *Proactiveness*: Ability to exhibit goal-directed behaviour and not just act in response to their environment.

The above properties are implemented through computational algorithms, expert systems or heuristic learning methods. Furthering our software agent concept, we have introduced the concept of agency in our system. An *agency* can be loosely defined as a set of agents performing a complex task. An *agency* implements the services using *agents*. *Agents* (in our context) are software entities which run in an environment provided by the *agency* and have intelligence to execute the task assigned to them. *Agents* request various facilities from the *agency* to carry out supplementary tasks. These facilities are provided by various managers in the *agency*. It should be noted that the *agency* just coordinates various *agents* and provides facilities to the *agents*. *Agents* can be considered the active entities whereas *agencies* and managers are passive. An *agent architecture* is mainly determined by the various *agents* (and their goals and behaviour) in the system, the communication protocol for inter-agent communication and the means by which knowledge is represented (for interactions between *agents*).

To illustrate clearly the difference between *agent, agency* and *manager* we consider an example of a *cell controller agency* (which will be described in detail later) and describe its functioning. A *cell controller agency* gets requests for services from many *agencies* and at the same time it too requests services from other *agencies*. Some typical requests include the following.

- Requests for the current status of the available resources.
- Requests for determining the process plans for new parts entering the system.
- Requests for preparing routing plans and scheduling available routers (AGVs) in the system.

There will be an *agent* for each type of service offered. Whenever the *resource allocation agency* gets a request for a particular service, it spawns the respective *agent*. The *agent* will work till the task is completed. The *agency* will provide various facilities such as authentication, resource loading, resource synchronization and concurrency through managers. *Agency* is not involved directly in implementing the service.

The essential features of an agent (figure 1) are the message handler which handles the communication between the agent and the other agents, the interpreter, which maps the KIF messages to the Java implementation of the agent and finally the concept of sessions, a session assigned to each dialogue.

An agency (figure 2) contains various managers. We will briefly describe five managers which make up any agency. They are the resource manager handling dynamic loading of resources, resource information caching, and resource concurrency among other things. The agency registry manager is responsible for registering any outside agent which requests for service, the configuration manager for configuring the agency, the security manager for authenticating in-coming requests and the interface manager which provides the user interface to the agency.

Agents communicate using an agent communication language (ACL). ACL is based on the idea that communication can be best modelled as the exchange of declarative statements (definitions, assumptions, and the like). Towards this end, researchers in the ARPA Knowledge Sharing Effort (Neches *et al* 1991) have defined the components of ACL that satisfy these needs.

ACL can be thought of as consisting of three parts – its vocabulary, an inner language called KIF (Knowledge Interchange Format) and an outer language called KQML (Knowledge Query and Manipulation Language). An ACL message is a KQML expression in which the arguments are terms or sentences in KIF formed from words in the ACL vocabulary (Genesereth *et al*).

KQML is a language and protocol for exchanging information and knowledge. KQML is both a message format and message-handling protocol to support run-time knowledge

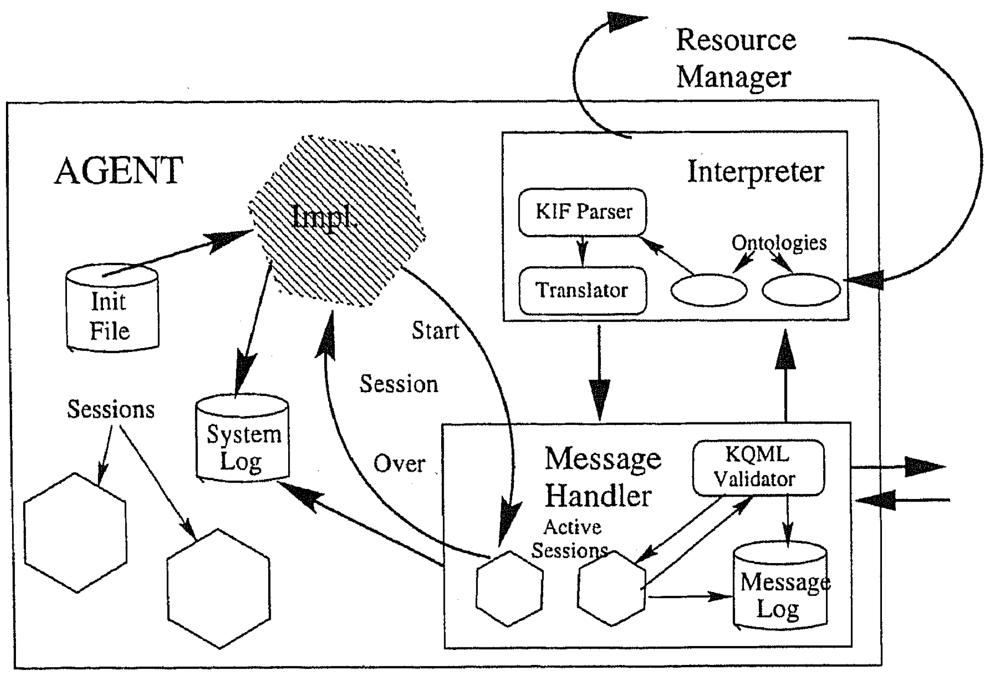

Figure 1. *Agent* structure.

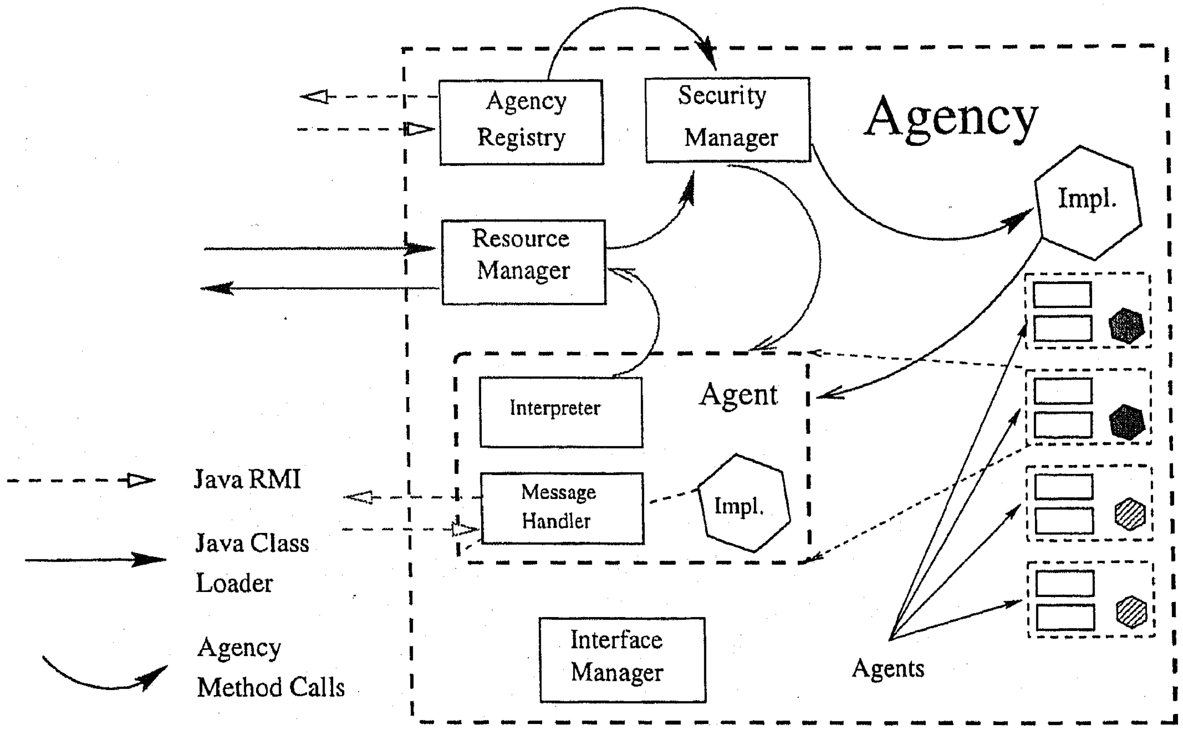

Figure 2. *Agency* structure.

sharing among agents. KQML can be used as a language for an application program to interact with an intelligent system or for two or more intelligent systems to share knowledge in support of cooperative problem solving (Finn *et al* 1993). Knowledge Interchange Format (KIF) (Genesereth *et al* 1992) is a formal language for the interchange of knowledge among disparate computer programs (written by different programmers, at different times, in different languages, and so forth).

To fix ideas, we briefly describe that factory floor work flow movement using the above mentioned architecture. Each batch of parts, once it enters the system is assigned an agency called part agency which would be responsible for all communication of information and part movement on the factory floor. Each part agency in turn will have part agents for each member (or set of identical members) of the batch. The agent has information on the part type, its schedule, part programs, tools, due dates, customer identification etc. The agent assignment occurs once the batch enters the load station. The agent would contact the machine agents, who can perform the *first* operation, for machine time to process the batch. The machine agent returns to the part agent with information on the status of waiting parts, tools, part programs etc. Using this information, the part agent decides the machine it wants to go for processing. This decision could be rule based: Go to the idle machine, in case of availability of several idle machines, go to the faster machine, in case of all machine busy condition, go to the one with the shortest queue etc. This scheduling information is provided by the scheduling agent. After selection of a machine server, the part agent contacts the AGV transporter agency for an AGV. The scheduled AGV communicates with the part agent regarding its present location and destination. It arrives to pick up the batch and delivers it to the machine. The protocol requires that the machine acknowledge arrival of the batch to the load station and the part agent and to the AGV transporter agency. This process continues until the part goes out of the system.

4. Overview of control agencies

We will now consider the various agencies (figure 3) that are used to abstract the factory floor control system. We can model the system as a colony of decentralized *agencies* that interact to service an order. The supervisor *agency* is the super user of the entire system and has overriding powers over all the entities. Moreover the *supervisor agency* sets the due-dates for the batches entering the system. The various agencies and the agents making up the system are described in this section. In table 1, we summarize various agents and their functions.

4.1 *The client or part agency*

In most cases, similar orders or orders for multiple parts are batched together. A *part agency* is responsible for each batch. For each order in the batch, the *part agency* spawns a *part agent.* When a new batch arrives, the supervisor *agency* spawns a *part agency* and provides it with the process requirements of each member of the batch. The *part agency* on its part spawns a part agent for each member and provides it with its process requirements.

The *part agency* also spawns a *co-ordinator part agent* for each batch which acts as the communication link of the entire batch with the other agencies. Once the individual

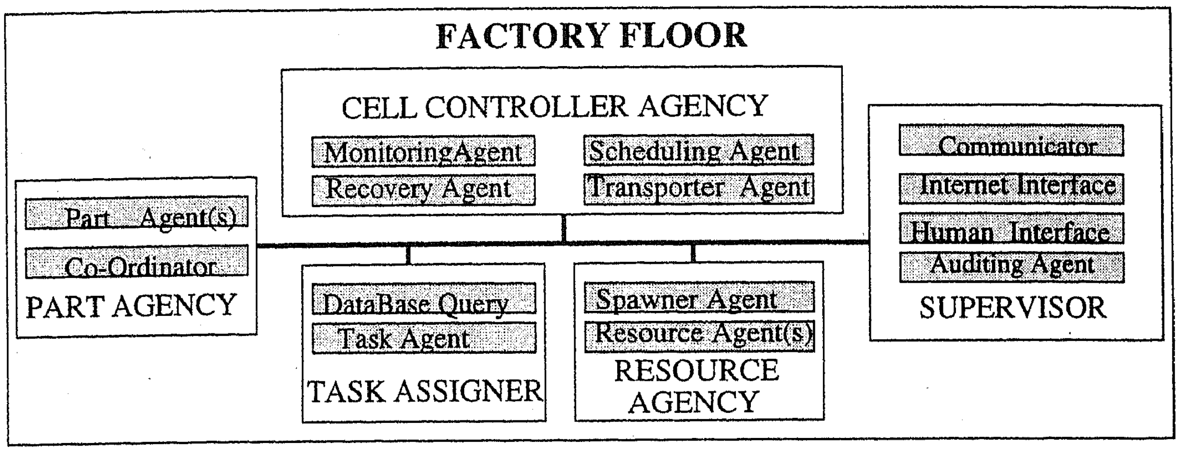

Figure 3. Control system agencies.

part agents have collected the necessary information about the available and compatible servers for execution of the process requirements, they pass on this information to the super part agent. This super part agent then contacts the scheduling agent (part of resource allocation agency) with this information for a detailed process plan for the entire batch. One this is worked out, each part agent is loaded with its detailed process plan and they start executing them.

4.2 *Task-assignment agency*

Each part agent has its own process requirements. These process requirements must be translated to a process plan. A process plan is basically a task graph with information about how each task (necessary for completing the job) is performed. In effect, the process plan consists of a sequence(s) of tasks and the machine server type associated with each task.

This agency consists of a *database query agent* which first checks (a database consisting of process requirement to process plan mappings) whether the process requirements match that of an already available process requirement to process plan mapping. If so, the available process plan is retrieved.

In case, the process requirement is new, a *task agent* is spawned which builds a process plan using heuristic algorithms and/or human intervention. Human intervention might be required because automatic generation of process plans for parts is currently feasible only in manufacturing research laboratories (Upton 1992). Once a new process requirement to process plan mapping is made, it is added to the database mentioned previously.

4.3 *Cell controller agency*

The factory floor has many resources such as machine servers, tool magazines, inspection centres, temporary storage units, and AGVs. Scheduling these resources is necessary so that optimal utilization takes place. In most available factory floor control systems, this resource allocation is closely built into the system resulting in loss of modularity (for

Table 1. Agency table.

Agency name	Functions
Part agency	(i) *Part agent*: The agent which is responsible for each member of a batch. (ii) *Co-ordinator*: This agent co-ordinates communication between the members of each batch and the outside world.
Task assignment agency	(i) *Database query agent*: Queries the system's database to check for batches with old task assignments. (ii) *Task agent*: Prepares a new process plan.
Cell controller agency	(i) *Monitoring agent*: Monitors the status of the system machine servers. (ii) *Scheduling agent*: Prepares schedules for each batch. (iii) *Recovery agent*: Calls recovery modules for repair and error correction. (iv) *Transporter agent*: In charge of the AGVS in the system.
Resource agency	(i) *Resource agent*: One each for the resource providers in the system. (ii) *Spawner agent*: Activated by the supervisor. Creates new resource agent.
Supervisor agency	(i) *Human agent*: The interface between human operators and the system. (ii) *Auditing agent*: Keeps track of the activities of the system. (iii) *Communicator agent*: The interface of the supervisor with other agents. (iv) *Internet interface agent*: The interface of the system with the Internet.

reasons mentioned in the introduction). In our architecture we have proposed a very modular structure. This *cell controller agency* consists of four agents, *monitoring, scheduling, recovery and transporter* agents.

The monitoring agent maintains a database about all available resources and their current status. When a part agent approaches it for a list of a particular type of resource, the monitoring agent queries its database to provide this information. If a particular resource is accepted by a part agent at any point of time, it communicates this information to the monitoring agent and hence the monitoring agent is able to keep an updated database all the time. Moreover whenever a resource fails to service a part agent (because of breakdown, lack of appropriate resources etc), the monitoring agent is intimated so that it can update its database on the non-availability of the service of that particular server. Moreover, when a new resource is added to the system, the new resource agent intimates the monitoring agent.

Once the process requirement and the resource availability is known, scheduling of the part agents of a particular batch has to be done. This is done by a generic scheduling agent. This is an intelligent agent which based on the requirements of the part agents and the availability of the resources works out an optimal schedule. The conversation is done between the super-part agent and the scheduling agent. In case the due-date fixed by the *supervisor agency* cannot be satisfied, the super-part agent negotiates with the *supervisor agency* for a possible modification of the due-dates.

Failures are very common in a manufacturing system. A flexible system should be capable of handling such failures gracefully without necessitating stoppage of the entire system (failures should have only local impact as far as possible). Failures detected by any of the part agents or others are intimated to the monitoring agent and the recovery agent which then calls the appropriate error-recovery (machine and/or human) modules.

The routing agent is a special type of monitoring agent except that it maintains information about the AGVs in the system. Since AGV routing is an added responsibility, we decided on a separate agent for this purpose. This agent maintains a database on the availability of AGVs and also provides routing information. The routing algorithms can be based on any of the conventional ones followed in present day routing systems (Viswanadham & Narahari 1992) for the AGVs when they are required to transport material on the factory floor.

4.4 *Resource agency*

As already mentioned before, a factory floor consists of many resources like machine servers, load/unload stations, inspection units, storage facilities etc. Each resource has its own capabilities and limitations. The *resource agency* proposed in this architecture has a *resource agent* for each service provider. It is assumed that each resource agent has a Java virtual machine running on it. Each resource agent has the necessary intelligence to ensure that tasks assigned to the resource server are completed – basically the intelligence to run the server. If the server has some problem, this resource agent terminates itself. The resource agents are spawned by a *spawner agent* whose function is to activate new resources. This *spawner agent* gets its input from the *supervisor agency.*

The resource agents maintain databases used for compatibility checks by part agents. Any resource server cannot be blindly selected because it might have some constraints either permanent or temporary. Part agents will have to do a compatibility check before generating a list of available servers. Part agents query the agent which makes its decisions based on the information available in its database.

In case some server loses some functionality due to some minor failure, the resource agent for that server terminates. But before doing so, it intimates the monitoring agent and the recovery agent. The recovery agent in turn calls the supervisor agency which then intimates the spawner agent to spawn a new resource agent which has updated (maybe reduced) attributes.

4.5 *Supervisor agency*

It is a recognized fact that the functions for a manufacturing control system are so demanding that there exists no known optimal control solutions. Use of heuristics is a time-honoured practice to overcome the complexity of the problems. The effectiveness of a heuristic depends on the context of its use in the configuration of the system and is generally determined by trial and error. These heuristic algorithms need fine tuning by human operators. The human operator should be capable of accessing the status information about the system as well as the individual servers before a decision can be made. Moreover the operator should be able to do that without bringing the entire system down. Also, if a particular server is giving problems, only that server is handled leaving the other servers unaffected. This agency is capable of accessing the various machine servers and effecting changes in their programs or other parameters. It also provides an effective interface for the human controllers to the entire system.

The *supervisor agency* sets the due-dates of the new batches. When the super-part agent cannot satisfy this (after the schedule has been worked out by the scheduling agent), the supervisor has the power to modify the due-date. Moreover it can recall part agents or entire batches by deactivating them (after altering their process plans so that they move out of the system).

The supervisor agency is made up of four agents: *human agent, auditing agent, the communicator agent and the Internet interface agent.* The human agent is the human interface to the entire system. It can be a convenient GUI for a human operator to monitor the system. The auditing agent maintains details of the factory floor events. Every agent is required to inform this agent whenever it makes a decision (like selection of machine server, AGV etc.). The communicator is the supervisor agency's interface to the other agents in the system. Any message emanating from the supervisor is sent via this agent.

In the futuristic scenario, the whole factory floor will be connected to the outside world through the Internet. Customers might be allowed to monitor the progress of their order or change specifications mid-way. Maybe someday some of the agents mentioned in this paper will be services offered by external parties dedicated to some application. In such a scenario, security of the system becomes paramount. In that case, it will be better to have a single point of entry (like a firewall) into the factory system. Customers can possess authentication tokens like "smartcards" or some biometric means to authenticate themselves before entering the system. The *Internet interface agent* is responsible for the interaction between the factory floor system and the outside world.

4.6 *Work-flow*

In this section, we describe the control system for handling material and information movement in a factory floor consisting of a number of computer controlled flexible machines and automated guided vehicles. In particular, we are concerned with the design of a coordinator architecture for interaction among various intelligent agencies mentioned below (figure 3.)

The flow-chart (figure 4) describes the detailed work-flow in the system. The process by which a client completes a job consists of several sequential steps. The process is first generated and for each task in the process plan, selection of servers, scheduling and completion follow the following steps.

4.6a *Sequence of steps:*

- A new batch enters the factory floor system. The supervisor agency (the human control) sets the due-date for the entire batch at this stage. A new part agency along with the co-ordinator part agent are created by the supervisor agency. Part agents for the members of the batch (identical members have only one part agent) are created and loaded with their process requirements.
- Each part agent contacts the task assignment agency for individual process plan (which includes the sequence of tasks and the type of machine servers where the tasks can be completed).
- The part agent then contacts the monitoring agent for the list of resource providers available for the tasks it needs to complete.
- The part agent does the compatibility checks with the resource providers.

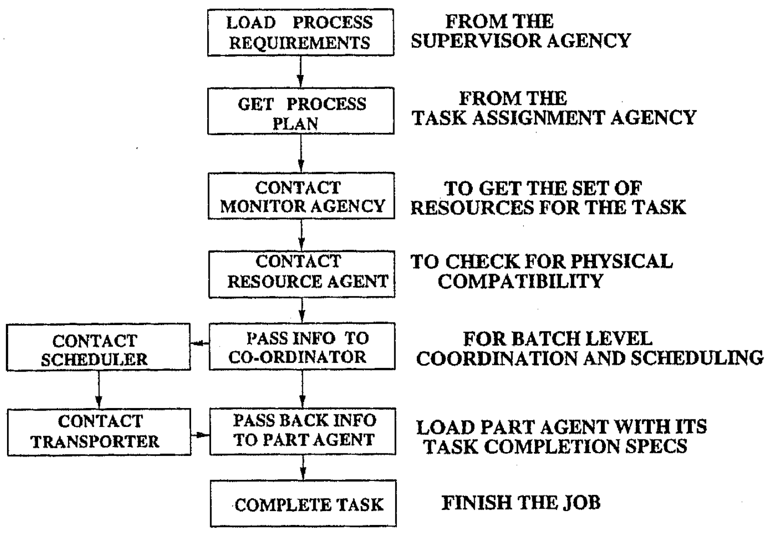
LOAD PROCESS REQUIREMENTS
FROM THE SUPERVISOR AGENCY
GET PROCESS PLAN
FROM THE TASK ASSIGNMENT AGENCY
CONTACT MONITOR AGENCY
TO GET THE SET OF RESOURCES FOR THE TASK
CONTACT RESOURCE AGENT
TO CHECK FOR PHYSICAL COMPATIBILITY
CONTACT SCHEDULER
PASS INFO TO CO-ORDINATOR
FOR BATCH LEVEL COORDINATION AND SCHEDULING
CONTACT TRANSPORTER
PASS BACK INFO TO PART AGENT
LOAD PART AGENT WITH ITS TASK COMPLETION SPECS
COMPLETE TASK
FINISH THE JOB

Figure 4. Work-flow diagram of a part-agent.

- Then the part agents pass on this information to the co-ordinator part agent which in turn contacts the scheduling agent scheduling the entire batch.
- The transporter agent contacts by the co-ordinator part agent for the routing information for each of the part agents.
- The process plan, the resource providers to go to and the routing plan are then loaded on the individual part agents. They would then initiate the process of completion of the job.

4.6b *Failures:* Of course, the above steps will work provided there are no breakdowns in the system. But on a factory floor, failures should have to be taken into consideration. The architecture proposed above easily accommodates failure recovery. Assume that a resource provider fails. The part agent using that provider or the next part agent which approaches this provider will inform the monitoring agent which updates its database. If the resource provider has just lost some functionality, the old resource agent is terminated and a new one with updated capabilities is spawned (by the spawner agent). Obviously a breakdown means that the schedule might have to be changed. The monitoring agent activates the failure recovery agent and informs the supervisor agency. This in turn activates the part agencies. The co-ordinator agent of each batch has knowledge about the part agents affected since it has the scheduling information of each part agent in its batch. These part agents are then rescheduled.

5. Conclusion

The main advantages of the distributed agent based control system presented in this paper are as follows.

Simple framework: The framework adopted in the new architecture is simple and logical. It accommodates present-day heirarchical architectures and at the same time processes the advantages afforded by the heterarchical structures because of distributed processing made in the new system. This new scheme can also accommodate heterogeneous systems.

The number of levels of interaction between agents before a decision is made is lesser compared to other models (Lin & Solberg 1992; Upton 1992) (in a heriararchical system, control messages have to pass many levels from the controlling computer to the entity whose action is being controlled) simplifying the design. The machines servers are independent with no agent supervising them. This control system also possesses all the advantages of heterarchical systems described in earlier sections.

Global knowledge: Global knowledge is retained in this system thereby avoiding the main disadvantage of completely heterarchical architectures. Heterarchical structures do not possess global knowledge and hence tend to be very sub-optimal in their performance.

Fault-tolerance: Any breakdown of a resource provider is easily handled without much loss of functionality of the entire system. This avoids the total lack of knowledge in fully distributed systems (knowledge of a breakdown is not propagated to all the members efficiently) as well as the elaborate schemes for fault detection usually employed in completely hierarchical schemes. Though it is understood that the entire system might come to a standstill if one of the agencies fail, it is a small price to pay to avoid the inherent disadvantage of heterarchical architectures.

Modularity: The entire system is very modular and hence addition or removal of resources does not pose any problems. The whole system need not be shut down to make minor changes. This makes the system highly reconfigurable.

Why software agents: The controller we have designed is based on the fast developing area of intelligent agents. Agent abstraction complements the classical object-oriented paradigm. The intention is that by focusing overall attention on the co-ordination of agents and their interactions, while treating the specific details of computation and information structuring at the level of objects, we can derive a greater power of abstraction. This helps the developers of the system better understand the dynamics of distributed application software (Cheong 1992). Moreover an agent-based architecture has the inherent advantages of decentralized concurrent processing and also helps in capturing the real world scenario faithfully and with less ambiguity, while at the same time ensuring that the right information is available at the required time.

The modified version of the above system was implemented satisfactorily for order management in manufacturing enterprises (Viswanadham *et al* 1998). Though the application was different, the agent and agency architecture was retained. This also goes to show the flexibility of our new system.

The main aim of this paper is to show that agent-based control systems is a simple logical way of obtaining flexibility both short-term as well as long term without sacrificing on performance. Since agents allow ease of changing heuristics without much disturbance to the system, the system becomes highly flexible and reliable. The paper discusses a new method based on agents similar to those suggested by (Lin & Solberg 1992; Upton 1992) which has very good potential for practical applications. We consider that agent-based architectures will become more dominant as the need for more flexibility becomes pressing. Future areas of research include heuristics that have to developed taking into consideration system specific knowledge and other related issues.

References

Cheong F-C 1992 *An agent-oriented programming language for heterogeneous distributed environments*. Ph D thesis, Dept. of Computer Science and Engineering, University of Michigan, Ann Arbor, MI

Dilts D M, Boyd N P, Whorms H H 1991 The evolution of control architectures for automated manufacturing systems. *J. Manuf. Syst.* 10: 79–91

Duffie N A, Chitturi R, Mou J-I 1988 Fault-tolerant heterarchical control of heterogeneous manufacturing system entities. *J. Manuf. Syst.* 7: 315-328

Finn T *et al* 1993 Draft specifications of the KQML agent communication langauge. http://www.cs.umbc.edu/kqml/kqmlspec/spec.html

Fox M S, Chionglo J F, Barbuceanu M 1993 The integrated supply chain management system. http://www.ie.utoronto.ca/EIL/public/iscm-intro.ps

Genesereth M R *et al* 1992 Knowledge interchange format version 3 reference manual. Stanford University. Logic Group, Standford, CA

Genesereth M, Singh N, Syed M A distributed and anonymous knowledge sharing approach to software interoperation. Stanford University. http://logic.stanford.edu/sharing/ papers/fgcs.ps.

Lin G Y -J, Solberg J J 1992 Integrated shop-floor control using autonomous agents. *IIE Trans.* 24(3): 57–71

Neches R, Fikes R, Finn T, Gruber T, Patil R, Senator T, Swartout W R 1991 Enabling technology for knowledge sharing. *AI Mag.*

Shaw M J 1988 Dynamic scheduling in cellular manufacturing systems: A framework for networked decision making. *J. Manuf. Syst.* 7: 83–94

Upton D M 1992 A flexible structure for computer-controlled manufacturing systems. *Manuf. Rev.* 5:

Upton D M, McAfee A 1996 The real virtual factory. *Harvard Business Rev.* (July–August)

Viswanadham N, Narahari Y 1992 *Performance modeling of automated manufacturing systems* (Englewood Cliffs, NJ: Prentice Hall)

Viswanadham N, Lakshminarayanan A, Deshpande M, Srinivasa Raghavan S R 1998 A distributed agent based controller for order management in manufacturing enterprises (submitted)

Woolridge M, Jennings N R 1994 Intelligent agents: Theory and practice. *Knowledge Eng. Rev.* (submitted)