

Distributed Scheduling and Control in Job Shop Manufacturing Systems: a Multiagent Architecture

José Alberto Araúzo,
Universidad de Valladolid,
Valladolid, Spain

Juan José De Benito
Universidad de Valladolid
Valladolid, Spain

Ricardo del Olmo
Universidad de Burgos
Burgos, Spain

Pedro Sanz
Universidad de Valladolid
Valladolid, Spain

Abstract - In the current global market, traditional hierarchical approaches for manufacturing scheduling and control don't produce the desired results. It occurs especially in Job-Shop manufacturing configurations. In order to overcome the limitations that hierarchical and centralized architectures show in new manufacturing systems, many decentralized schemes have been proposed. Most of those proposals are based either on dispatching rules or on market mechanisms. While dispatching rules approximations provide a good solution to manage large number of disturbances, market approaches provide efficiency. This paper presents a multiagent architecture which combines both types of ideas.

Keywords: MultiAgent systems, Shop Floor Control, Job-Shop, Scheduling, Holonic Manufacturing Systems, Lagrangian relaxation

1.0 Introduction

In order to make products, the elements of a manufacturing system need to perform a series of tasks that must be perfectly coordinated. This coordination is achieved by means of a programming and controlling system called shop floor control (SFC). So far, SFC has been designed on hierarchical or centralized schemes that have been implemented efficiently by software such as SCADA for continuous productive configurations. Unfortunately this is not true of discreet productive systems as Job-Shop manufacturing configurations.

Job Shop manufacturing systems have a set of features, which make the problem quite difficult. A wide range of products must be made in small number of units by means of different low specialization machines. The sequence in which the products pass through the machines and the number of the machines needed to make them will vary depending on the type of product. The assembly of parts and alternative routes of production are also possible. A priority also exists for each product, depending on the due dates agreed on with the customer. The programming and controlling of this type of productive systems is very complex and it is responsible for the fulfilling of the due dates, the uses of resources, stocktaking and, therefore, the efficiency of the system.

Hierarchical and centralized architectures are not flexible enough to adapt themselves to the dynamism and complexity needed in Job Shop productive configurations. That is why successive proposals have appeared to improve the shop floor control of that type of manufacturing systems. The recent paradigm of MultiAgent Systems (MAS), which offers new techniques to face complex unsolved problems, can help to find promising solutions. As a matter of fact, it is currently a very active field of investigation [1]. MAS provide methods to replace a central controller by several controllers, usually referred to as autonomous agents. Each agent is related to a small portion of the system (a machine, a part, etc.)

This paper presents information about the experience obtained during the development of a small prototype of shop floor control system based on

agents. Therefore, we propose to analyze the difficulties that lie in the control of Job-Shop manufacture environments. Secondly we will describe both the traditional solutions and the current proposals. The agent technology provides a perfect frame to develop many of these proposals, and this is dealt with in the following point. Next we will show the prototype we ourselves have developed and finally we will mention the most relevant conclusions we have arrived at.

2.0 Job-Shop Floor Control Problems

Obviously it is difficult to control any complex system. Even so, there are three aspects regarding Job-Shop plants that are worth pointing out, a type which is particularly difficult to deal with: the programming of production, the unexpected disruptions that might arise, and the evolution of the structure of the system through time [2]. It is essential to arrange planning beforehand so that not only the productive resources can be prepared properly, but also the tasks can be coordinated. These plans are production programs that include all the activities each resource must carry out, and also the dates on which each activity starts and finishes. Although there are many possible programs, not all of them succeed in making the most of the resources. Nor do they achieve all the aims. Selecting an ideal program among them all proves to be a very complicated task, since it is a np-complete problem. In this kind of problems a combinatory explosion takes place: the number of all possible programs increases in huge proportions as the number of machines and operations to be carried out increases as well. On many occasions the calculation time which is necessary to find the best solution is beyond reasonable limits. Fortunately there is software available which is based on heuristic and meta-heuristic methods, which guarantee reasonable programs in reasonable time.

The programming of operations would not be a big problem if it were not connected with the dynamism of the industrial environment. The production systems are subjected to continuous disruptions, which makes it rather difficult to comply with the fixed programs. Phenomena like machine failure, changes in the orders or mistakes in the manufacturing program forecast might invalidate the manufacturing program itself after it has been issued. Therefore, under these conditions the control system must deal with any eventuality and modify plans in real time in the best possible way.

There is one more problem that must also be taken into account: the dynamism of the structure of the system itself. The technological advances in the production system, the incorporation of machines or the modification in the managerial strategies make it possible for the manufacturing system to evolve. In order to program and control manufacturing, the SFC must have a model of the production system. If the latter changes, the SFC system must include all the modifications that have been carried out. As a result, the control system might prove inoperative depending on the extent of this change.

3.0 Proposed Solutions

Throughout the second half of the twentieth century, necessities in the industrial sector favored the development of quite efficient control systems. Even so, on many occasions these systems do not offer the desirable performance, due to the evolution of the industrial environment. Owing to the globalization of the markets and the technological obsolescence customers demand products made to order. Consequently, the range of products companies have to make is becoming bigger and bigger, and in smaller quantities. As a result of this and the dynamism of the current markets, the SFC systems have to face up to more and more complex systems.

3.1 Evolution of control architectures

The control of manufacturing systems has been traditionally carried out according to centralized or hierarchical multi-level architectures [3]. The former is based on a mainframe computer, which sets the program of activities, sends the orders to the plant, captures data, upgrades the databases and reprogram if necessary. The latter is more complex and

results from the evolution of the former. There are controllers at different levels that centralize the control of different areas in the plant. Although the latter is no doubt better than the former, both of them lack in robustness, have a slow response and are too rigid.

By introducing relations between elements, which are at the same hierarchical level, it is possible to improve the robustness and response time of the system. Even so, this type of structures, called modified hierarchical, fails to improve adaptability and extendibility. Their main drawback is that the hierarchical relations are very difficult to modify.

This tendency towards elimination of hierarchical structures inexorably leads to the so-called heterarchical or transversal architecture, in which there are no higher levels that hierarchy decisions. In order to achieve coordination it is necessary to create a model of negotiation that allows the controllers to get the production planning by mutual agreement. This makes the system more flexible, scalable, robust and it adapts to manufacturing necessities constantly and perfectly. It has a problem, though: the elimination of hierarchies makes it more difficult to achieve global objectives.

3.2 Current proposals

Owing to the physical structure of the manufacturing systems and the distributed location of the information in these systems, one of the proposals consists in the use of distributed architectures that are not so hierarchical so as to carry out the SFC systems. Many authors, inspired by the organization of the natural systems (living beings, human societies, etc.) propose original control systems. Among them, bionic manufacturing systems, fractal manufacturing or holonic manufacturing systems are worth mentioning [4].

Although these proposals start from different ideas, they share a feature that makes them attractive to researchers. They are self-organized systems where behavior emerges from the interaction between the autonomous entities. Obviously it is somewhat difficult to design and implement these systems. It is in this aspect that software agent technology can be useful, since it provides the appropriate tools for the design and implementation of complex systems.

4.0 Shop Floor Control Multiagent Systems

There are many proposals of multiagent systems for SFC, which are briefly dealt with next [5][6]. Like any system based on agents, they can be described in general terms by means of **agent models** (features of each type of agent, instances, etc.) and **agent society models** (structural relations and relations between agents).

4.1 Agent Models

From a traditional point of view of the software engineering, the systems are structured in modules by means of a functional division. According to this idea, the information systems of manufacturing companies are divided into modules used for functions such as maintenance, task scheduling, material management or design. Modules can also be divided by means of physical correspondence, in which there would be a correspondence between software modules and physical entities of the system. The latter division is the one that is most commonly used in manufacturing agents, since it makes the manufacturing system much more extensible and adaptable.

Although some functional agents appear in many studies, most of them (including the ones that refer to holonic manufacturing), correspond with the idea of physical division and propose two kinds of basic agents: '**product or order agents**', and '**resource agents**'. These agents have local information of the element they represent and they do not need to know the rest of the system completely in order to carry out their task. The order agents represent the orders that must be manufactured. They are created when a new order is placed and contain all the information about it: the due date, the product that has to be made and all the operations that are necessary to make it, restrictions in the sequence in which these operations are carried out, etc. The resource agents represent the elements of the productive system: (machines, robots, warehouses, etc.). Like the

product agents, they contain local information of the element they represent [figure 1].

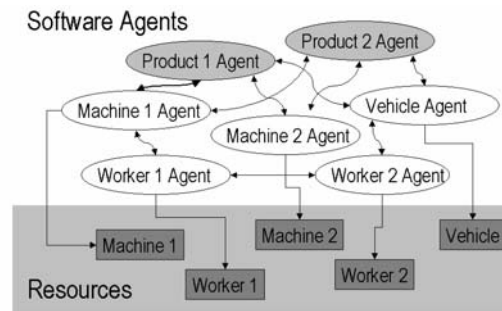


Figure 1: Multiagent System for Shop Floor Control

4.2 Agent Society Models

When it comes to establishing the relations between agents, the proposals are not as homogeneous as they are when establishing the type of agent. After revising the related literature, it is possible to clearly distinguish between two types of manufacturing agent societies: systems with programmer agent and systems without programmer agent. The existence of this type of agent marks the hierarchical character of the system, and therefore, the relations between its components [7].

In the first type of society there is a functional agent that carries out the programming of the necessary operations in order to get the production of the pending orders. In these systems the order agents ask the programmer agent to make them. The latter asks the resource agents for information about their capacity and availability, and prepares a manufacturing program with them. According to this program, different instructions are sent to the resource agents so that they start to operate.

In those systems with no programmer agent, the schedule emerges from the interaction between the order agents and the resource agents. The main question that arises when it comes to designing these systems is how to establish the interaction. Although there are many proposed solutions, none of them is definite or has been widely accepted. Among these we can point out the systems based on the market, dispatching rules or pull systems.

5.0 Prototype Design and Implementation

According to the ideas taken from literature, we have created a system prototype based on agents to carry out the *SFC* of the Job-Shop manufacturing systems. The multi-agent technology offers many tools, but we have selected *JADE* as development framework [8], and *GAIA* as analysis and design methodology [9]. Bearing in mind that the *FIPA* specifications are accepted throughout the world, we have developed the prototype according to their recommendations [10].

5.1 Shop Floor Scenario

A real manufacturing system is a very complex environment. Since in this study we intend to develop a first approach to the control system, it is advisable to design a simplified manufacturing system that can be simulated by means of virtual techniques. Its simplification will affect to the structure of the manufactured products (items) and the structure of the productive system (resources). In our model we do not consider assembling the parts we manufacture.

The productive system must manufacture these items following production orders, which consist of an order specifying type of item, number of units, priority, and delivery date. In order to manufacture each item it is necessary to carry out a series of operations, which have to be executed in the correct sequence. This order depends on restrictions called precedence relations between operations (for example, the operation of coating cannot be carried out prior to executing the operation of sanding). Thus, the technical data of the product consist only of a list of operations and a

list of precedence relations that limit the order in which all the operations can be carried out.

As far as the resources are concerned, we take it for granted that there will be unlimited availability of raw material, tools, personnel and transport. In this way the only resources arranged by the system will be the work center or machines. Although in a real environment a machine usually needs a lot of information to execute any operation (tools needed, human resources, etc.), in the manufacturing system we propose, the information is limited to a list of operations the machine can carry out, and the time needed for each operation.

5.2 Agent Models

As it has been mentioned previously, an essential aspect of any multi-agent system is the specification of the agents that take part of the system. We have defined four kinds of agents: *Database Agent*, *SFCCoordinator Agent*, *Order Agent* and *Machine Agent*.

The agents of the type *Database Agent* are those which have access to the information of a database that contains the technical information of the products and the orders to be manufactured with arranged delivery dates. The users of the system can use a window that is linked with this agent by means of which they can record and read in the database. The main task of this agent is to get the pending orders from the database and tell the *SFCCoordinator Agent* to order the manufacturing of these orders. Although there can be various agents like this in the system, in ours there is only one.

Another type of agent that appears in the system is the *SFCCoordinator Agent*. This agent receives the orders from the *Database Agent* and creates as many *Order Agents* as orders have to be manufactured. As the operations of the different orders are carried out, the *Order Agents* inform the *SFCCoordinator Agent* of the stage it is in the manufacturing process. When an order has been finished the *SFCCoordinator Agent* informs the *Database Agent* that it has been finished in order that the latter records it in the database.

The *Order Agents* represent the number of orders to be manufactured, and there are as many of them as pending orders. Their main task is to negotiate with the *Machine Agents* which machine (and when) is going to carry out all the necessary work to finish the product. Besides these agents keep the *SFCCoordinator Agent* informed about the stage of the manufacturing process.

The *Machine Agents* are related to the machines there are in the manufacturing system. That is why there are as many of them as machines. These agents negotiate with the *Order Agents* on the operations that the machine they are associated with must carry out. Once a certain operation for an *Order Agent* has been arranged, the *Machine Agent* sends the necessary instructions to the machine so that it starts to operate. The *Machine Agents* checks how the manufacturing is going and informs the *Order Agents*. Through a window, users can change the parameters of this type of agents, such as the communication with the machine, the operations that the machine can carry out, etc.

5.3 Agent Society Models

Most of the tasks the agents carry out are communicative acts. The regulation of these communications is essential so that the system will accomplish its objective. So as to guarantee the adequate exchange of messages, *FIPA* has defined a communication system called *ACL* (Agent Communication Language). A very important aspect of this communication system is the control of the conversation, which is determined by the communication protocols. *FIPA* establishes some of these, two of which we have implemented: *Request-Protocol* and *Contract-NetProtocol*.

The *Request-Protocol* is used to ask other agents for services. The first agent asks the second one to perform a task. The second one confirms whether it is going to carry it out or not, and once it has been carried out, it sends a report with the result of the operation. We have also used this protocol to implement the request for manufacturing orders that the *Database Agent* sends to the *SFCCoordinator Agent*. In the same way this protocol is used to implement the request that the *SFCCoordinator Agent* makes from each *Order Agent*.

The *Contract-Net Protocol* is used when the first agent intends to hire a second agent to carry out a certain task, and so it starts the conversation with a request for offer. The second agent finds it satisfactory, it accepts it and sends confirmation of acceptance to the second agent. Once the latter has carried out the task, it sends a report to the first agent.

This protocol has been proposed by some researchers to implement the negotiation between orders and machines [11][12]. The sequence of operations that the machines carry out and, consequently, the efficiency of the system depends on the way this protocol is carried out (which is the first agent, how the offer is selected, etc.). In our system the agents which start the process are the *Order Agents*, which ask the *Machine Agents* to perform the operations.

The figure 2, show our implementation of the Contract-Net Protocol. It starts with one or several *Order cfp* (Call For Proposal), where each order agent asks a machine agent for a time interval to perform a certain operation. After that, each *Machine Agent* selects only one *cfp*, and it sends a proposal to the *Order Agent* that sent the selected *cfp*. When an *Order Agent* gets a proposal, it can accept or reject it. If it accepts the proposal, then the *Order Agent* and the *Machine Agent* will add this piece of information in their databases.

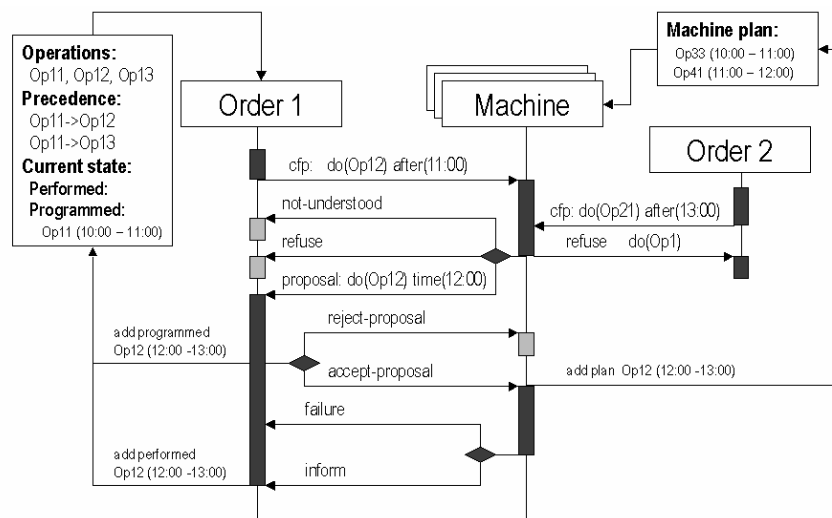


Figure 2: Contract-Net Protocol implementation.

The previous implementation have three points where certain agents must make decision: (1) the order agents have to decide what operation must be included in its *cfp*, (2) and the machine agents have to decide what *cfp* must be considered. In order to make these decisions, many works have proposed to make use of dispatching rules. Although this choice provides a good solution to manage a large number of disturbances, it does not provide global efficiency.

In order to improve the global behavior of the system, we have implemented another negotiation level, which is based on a Lagrangian relaxation mechanism [13]. That negotiation level runs constantly and parallel with the 'contract-net protocol', and it provides the necessary information to make decisions. The Lagrangian relaxation mechanism acts according to the following method. (1) Machine agents make public their available time interval prices. (2) After that, every order agent proposes a local schedule to agent machines to make their pending operations. In order to calculate the local schedule, order agents probe all possible schedules, and they take the cheapest. (3) The global schedule that is constructed by means of local schedules combination may not be feasible (many time intervals of machines may be shared for some order agents). If it happens, machine agents will publish new prices (higher prices for highly demanded time intervals), and the process will start again. Although local schedules can not form a feasible global schedule, they can provide good information to make decisions during the contract-net protocol.

6.0 Conclusions

After studying the literature, we can establish that there are numerous proposals for the implementation of the multi-agent system to develop SFC systems. The most widely accepted idea, which is the one we have followed in our study, consists in building an agent society. In this society the physical agents that represent the real resources interact among them and with other software agents without physical correspondence. These agents negotiate according to certain schemes, and the result of this interaction will be an acceptable performance of the system.

According to these ideas we have developed a small prototype in *JADE* platform based on the *GAIA* design as it has mentioned above. By developing this prototype we intend to verify the adaptation of these techniques to develop SFC systems based on agents. From what we know after all our experience, it seems that this is possible. From the development of the work itself and the conclusions, several questions arise: Have the best tools been used or are there others that can adapt to this problem better? If the manufacturing system were more complex, would the results be equally promising? And in a real flexible manufacturing system?

7.0 References

- [1] W. Shen, Distributed Manufacturing Scheduling Using Intelligent Agents, *IEEE Intelligent Systems*, 17(1), 2002, pp. 88-94.
- [2] D. C. Bussmann, D. C. McFarlane, Rationales for Holonic Manufacturing Control, 2nd Workshop on Intelligent Manufacturing Systems, Leuven, Belgium, 1999, pp. 177-184.
- [3] M. D. Dilts, N. P. Boyd, H. H. Whorms, The Evolution of Control Architectures for Automated Manufacturing Systems, *Journal of Manufacturing Systems*, 10(1), 1991, pp 79-93.
- [4] A. Tharumaraja, A. Wells, L. Nemes, A comparison of the bionic, fractal and holonic manufacturing concepts, *Int Journal of Computer Integrated Manufacturing*, vol. 9, 1996, pp. 217-226.
- [5] H. Van Dyke Parunak, A Practitioners' Review of Industrial Agent Applications, *Autonomous Agents and Multi-Agent Systems* 3:4 (2000) pp 389-407.
- [6] W. Shen, D. H. Norrie, Agent-Based Approaches for Intelligent Manufacturing: A State-of-the-Art Survey, *Knowledge and Information Systems*, an International Journal, 1(2), 1998, pp 129-156
- [7] R. W. Brennan, D. H. Norrie, Metrics for Evaluating Distributed Manufacturing Control Systems, *Computers in Industry*, 51, 2003, pp 225-235.
- [8] *JADE* web site, 2005, <http://sharon.cselt.it/projects/jade/>
- [9] M. Wooldridge, N. R. Jennings, D. Kinny, The Gaia Methodology for Agent-Oriented Analysis and Design, *Autonomous Agents and Multi-Agent Systems*, 3, pp 285-312
- [10] *FIPA* web site, Foundation for Intelligent Physical Agents, 2005, <http://www.fipa.org/>.
- [11] A. Saad, K. Kawamura, G. Biswas, Performance Evaluati3n of Contract Net-Based Heterarchical Scheduling for Flaxible Manufacturing Systems, *Intelligent Autonomous and Soft Computing*, 3(3), 1996, pp 229-248
- [12] J. Barata, L. M. Camarinha-Matos. Implementing a Contract-based Multiagent Approach for Shop Floor Agility, *Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA'02)* 2002, IEEE Computer Society.
- [13] Christos A. Kaskavelis and Michael C. Caramanis. Efficient Lagrangian relaxation algorithms for industry size job-shop scheduling problems, *IIE Transactions* 1998 30, pp 1085-1097