MAS Technology for Flexible and Adaptive Production Programming

M^a Emilia Garcia, Soledad Valero, Estefania Argente, Adriana Giret and Vicente Julian

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valéncia, Spain
{svalero, mgarcia, eargente, agiret, vinglada}@dsic.upv.es

Abstract. One of the main critical problems in manufacturing systems domains is the production scheduling process because obtaining an agile and reactive production planning and scheduling system is essential in manufacturing. The production scheduling process is a complex problem in which finding a suitable production scheduling can extremely increase the effectiveness of highly flexible production processes. Nevertheless, this high flexibility makes the production scheduling and acquisition of relevant data quite complicated. Therefore, there is a strong demand for a universal and flexible tool for production scheduling capable of increasing the utilization of resources and that supports a decision making process for the selection of production orders. In this paper, a Flexible and Adaptive Scheduling Tool to develop an adaptable, fault tolerant and scalable scheduling system for a manufacturing environment is proposed. The proposal is based on multi-agent systems because provides a natural way to solve problems in domains of this kind.

Keywords: multi-agent systems, production scheduling, agent-based tool.

1 Introduction

The future of the manufacturing sector in the world will be determined by how it meets the challenges of "new manufacturing". Such manufacturing systems will need to satisfy fundamental requirements [18, 30], such as: enterprise integration, distributed organization, heterogeneous environments, interoperability, open and dynamic structure, cooperation, integration of humans with software and hardware, agility, scalability and fault tolerance.

Manufacturing requirements impose important properties on modelling manufacturing systems [18]. These properties define functional attributes and specific requirements for the system structure and the system development process which must be considered by the methodology. Bearing these requirements in mind we have studied software engineering methodologies which are best suited for problems of this kind. We conducted a study [36] on Object-Oriented Methodologies, Enterprise Modelling Language and Methodologies and Agent-Oriented

Methodologies. This study has demonstrated that MAS methodologies are good candidates to work with.

Lastly, an increasing amount of research has been devoted to holonic and agent based manufacturing over a broad range of both theoretical issues and industrial applications. We can divide these research efforts into two groups [22]: Control Architectures and Control Algorithms. Some examples of control architectures are: PROSA [37], the agent based architecture of Bussmann [7], agents and function blocks of Deen and Fletcher [11], MetaMorph [21], INTERRAP based architecture [10]. The developments about Control Algorithms range over [22] : Planning and Scheduling [13, 5], Execution and Shop Floor Control [12, 16], and Machine and Device Control [32, 33]. In spite of the large number of developments reported in these areas, there is very little work reported on modelling manufacturing systems with a Software Engineering Methodology, although its benefits. To date, many of the developments in the manufacturing field have been conducted in an almost ”empirical way”, without any design methodology.

Current manufacturing enterprises desire being flexible, responsive, adaptive and able to cope with the variability of demand. Decisions need to: be made faster, be more formalized, deal with vast amounts of data, fit with business objectives, be right, etc. The Intelligent Manufacturing Systems (IMS) programme aims to fulfill these requirements. One of the promising approaches in this field is the Agent-Based Manufacturing System paradigm [30] which have proved to be a successful tool [1, 4–6, 8, 12, 14, 15, 17, 20, 24, 27–29, 31, 33, 35] to implement systems of the ”new manufacturing” era.

Over the last years, a number of researchers have used agent technology in attempts to resolve the manufacturing scheduling problem, whereas only a few testbeds and real industrial applications have been developed and reported. For example, in [25] agent-based systems for shop floor scheduling and machine control are applied. The prototype systems supported only three industrial scenarios sponsored by its project members AMP, General Motors and Rockwell Automation/Allen-Bradley. Another example is ExPlanTech, a multiagent support for manufacturing decision making [26]. This framework provides technological support for various manufacturing problems and comprises different components, which can be assembled to develop a system that supports a user's decision making in different aspects of production planning. In [23] CAMPS is presented, another example of integrating agents with constraint satisfaction, where a set of intelligent agents tried to coordinate their actions for satisfying planning and scheduling results by handling several intra-agent and interagent constraints. Finally, a general methodology of aged-based manufacturing systems scheduling, incorporating game theoretic analysis of agent cooperation is presented in [3] to solve the n-job 3-stage flexible flowshop scheduling problem.

Despite the cited applications, new non-domain depended and flexible enough tools for production scheduling [1] are needed, which should be able to increase utilization of resources and supports a decision making in a selection of orders.

[1] In the rest of the paper we use the terms production programming and production scheduling interchangeably.

The flexibility of these tools should allow the system to adapt itself to changes in the production structure, or to allow using the system by another customer. In this way, we present FAST (Flexible and Adaptive Scheduling Tool): a Multi-Agent based tool to develop a flexible, fault tolerant and scalable scheduling system for the manufacturing area.

The rest of the paper is structured as follows: section 2 focuses on the production programming problem; section 3 describes the proposed toolkit, showing its multi-agent based architecture and the development process; section 4 presents an application example over a real factory and, finally, some conclusions are explained in section 5.

2 Production Programming Problem

Almost all of the manufacturing enterprises are facing the problem of how to set the optimal production programming to effectively exploit the company resources while reaching the highest turnover. To keep business running and to avoid such critical situations when the company production is not balanced (e.g. the company does not receive an optimal number of production orders able to manufacture) an extra attention to production programming must be paid.

The production programming process involves two main steps: (i) definition of the plan, in which production volumes and boundaries for the next period are defined; and (ii) production scheduling (weekly) and sequencing (daily), in which orders are completely assigned to resources and times. Initially, the production department defines the production volumes for the next period (plan), based on a compromise between the available production capacity and the sales request for production, also taking into account forward sales and stock level requirements. Production programming meetings are normally held on a monthly basis, and they typically determine the production programme for the next 3 months. Later, the production department (Figure 1) uses *Planning* data, information about *Raw Materials* availability, *Resources Availability*, factory and external *Events*, *Work Orders*, *Enterprise Goals*, etc., to generate the production programming schedule, in which start time and resource allocation for specific lot productions are specified. Therefore the orders are assigned to certain build weeks and days in the different plants according to the available production capacity.

Nowadays, the production programming process is characterized by: (i) weak automatization, where a large number of schedules are static; (ii) not reactive, where scheduling systems do not face the set of several events happening in a period (break down, supplier fault, environmental impacts such as humidity, temperature, etc.); (iii) not taking advantage of singularities, so schedules are based on global models that do not consider neither the own peculiarity of each stage of the production process nor the differentiated states of each stage; (iv) not distributed/no distribution, as schedules are executed on a single, centralized computer; (v) myopic, as models are based on one simple objective function.

Therefore, the aim of this work is to make available a tool for supporting new scheduling system developments, in order to improve the current production programming process, making it more reactive, dynamic and automatic, so the production system and customer services are enhanced. Thus, FAST provides with a powerful framework to develop flexible, fault tolerant and scalable scheduling systems. Furthermore, this tool supplies the necessary services to track the scheduling progress, making possible to detect alterations and prevent problems.

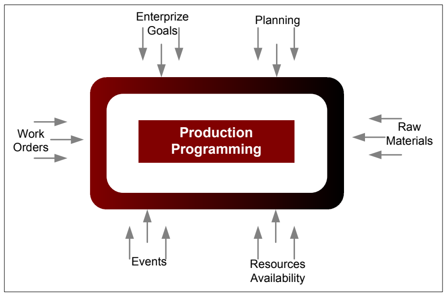


Fig. 1. Production Programming Information Needs

3 The FAST Toolkit

The FAST toolkit is a multi-agent based tool which allows developing flexible, fault tolerant and scalable scheduling systems in the manufacturing environment. It can be considered as a configurable scheduling tool which supports the creation, modification and monitoring of production programming schedules. Obviously, each manufacturing system has its own particularities. Therefore, the FAST toolkit has been conceived as a helping tool for the developer adding a specific development process. Thus, this process facilitates the construction of an integrated production programming system. In next sections, the multi-agent approach and the suggested development process are described.

3.1 Multi-agent approach

Automated workflow management systems have proved to be the driving force behind successful decision making in industry. Multi-agent systems (MAS) technology offers a convenient platform for workflow modelling. A framework where each agent represents a real information unit of the modelled enterprise is an appropriate model for optimization and visualization of flows of material, work, and information. The main features and the most interesting properties of using the multi-agent system approach in contrast with conventional software systems are scalability, modularity, flexibility and online reconfigurability.

Every production programming process (no mater the particular manufacturing enterprise in which it is defined) has to address the schedule creation, schedule modification (re-programming), and schedule execution control problems. Therefore the FAST toolkit provides a set of agents that cover all those activities (figure 2): (i) *Schedule Creation Controller* agent, that oversees the information about a new schedule order; (ii) *Schedule Modification Controller* agent, which maintains information about changes needed for adjusting the schedule because of failures in the manufacturing process; (iii) *Scheduler* agent, who has the ability to schedule tasks and resources; (iv) *Plant Wrapper* agent, that maintains and provides information about all restrictions and features of each machine and plant element; (v) *Lot Planner* agent, that manages all information about task sequences needed to manufacture a given lot; (vi) *Schedule Execution Monitor* agent, which supervises actual execution of a schedule in a specific plan; (vii) and *Manager* agent, that maintains the integrity of the system and regulates the cooperation among its different agents.

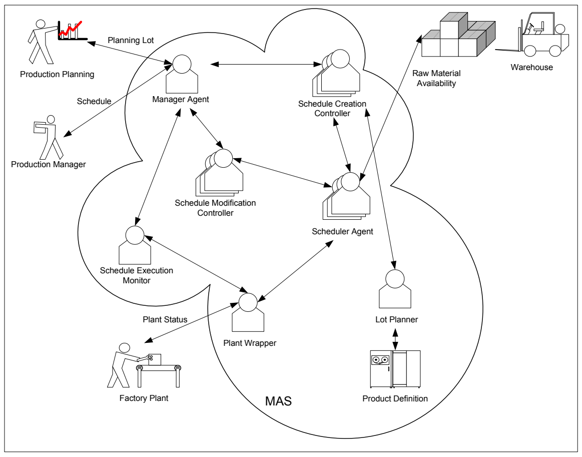


Fig. 2. MAS architecture

Three main scenarios can be identified: (i) schedule creation; (ii) schedule execution monitoring; and (iii) schedule modification scenarios. In the ***schedule creation scenario***, the Production Planning Department provides the *Manager* agent with all new planning lots that need to be programmed. Then the *Manager* agent starts an instance of a *Schedule Creation Controller* (SCC) agent, which is in charge of controlling the creation of the new required program. This SCC agent starts an interaction with the *Lot Planner* (LP) agent, which informs about the appropriated task sequences and the requirements needed to manufacture a given lot. After this, the SCC agent will send a request to the *Scheduler* (or *Scheduler group*, formed by several Scheduler agents) to obtain the best possible schedule for the required production lots. During the schedule creation, the *Scheduler* agents must interact with the adequate *Plant Wrapper* (PW) agent to get the current plant status and specific information about each machine inside the plant. Once the *Scheduler* has a valid allocation task proposal, it informs the corresponding SCC agent which, after checking the proposal, forwards the new schedule to the *Manager Agent* and finishes its execution.

In the ***Schedule modification scenario***, when the *Manager* agent receives a schedule modification order from the *Schedule Execution Monitor* (SEM) agent, it activates a *Schedule Modification Controller* (SMC) agent whom will be in charge of the management of the scenario and responsible for getting a new schedule with the proper re-programmed tasks. This SMC agent requests the *Scheduler* to determine start times and resource allocations for that part of the schedule that needs modification. So the *Scheduler* initiates a scheduling process, in which it requests the *Plant Wrapper* agent to provide the updated information about the *Plant Status*. It also queries the *Warehouse* system in order to determine *Raw Material Availability*. With all this information the *Scheduler* assigns, if possible, manufacturing tasks to resources. When the scheduling process finishes, it communicates the modified schedule to the SMC or indicates the impossibility to produce it because of unrecoverable plant failures.

In the ***Schedule execution monitoring scenario***, the *Manager* activates the *Schedule Execution Monitor* (SEM) agent (one agent for each plant), and asks it for information about the current schedule in execution. Then the SEM agent requests the *Plant Wrapper* (PW) agent to inform about the actual state of this schedule. Each time the PW agent detects an error, it notifies the SEM agent, which evaluates whether this problem can affect other subsequent schedules. If so, the SEM agent informs the *Manager* that a subsequent schedule has to be modified because of errors produced during the actual schedule execution.

Some advantages of MAS for our specific approach are: (i) the system is highly flexible, enabling to use different models and methods to solve the scheduling problem in every stage of the manufacturing process and to easily change them dynamically or even employ them concurrently; (ii) it may integrate and optimize a range of scheduling goals related to different processes (such as to optimize production programming to reduce stock levels in warehouses and in transit; to keep service level at acceptable grade; to achieve a higher responsiveness to competitor activity and reduction in lost sales opportunity; to ensure that the schedule is realistic in terms of resource utilization and required dynamic switches); (iii) it offers intelligent reasoning about the enterprise resources with the aim to produce accurate estimation of project's deadline and costs; (iv) it maintains system state updated and can adapt to changes in the environment while still achieving overall system goals (online reconfigurability), also allowing for re-planning, i.e. maintenance of the plans created so far in such a way that they are dynamically updated with respect to eventual changes in co-operating agents (e.g. resource/agent breakdown); (v) it improves reactivity to events and enables dynamic scheduling problem resolution; (vi) it offers rapid response to new system requirements through the addition of new modules or reconfiguration of existing ones; (vii) it enables to dynamically integrate new agents, remove existing ones or upgrade agents; (viii) agents operate asynchronously and concurrently, which results in computational efficiency; (ix) communication between agents is independent of message content (thanks to FIPA [2] protocols); (x) the system is also prepared for Just-in-Time production by using a real-time scheduler and providing real-time information to the agents of our platform.

3.2 FAST Development Framework

The FAST toolkit provides a development framework which is formed by a set of fixed system-independent modules and a set of libraries and files which should be used and adapted by the system developer. This toolkit offers different utilities as pre-built modules: the agent execution framework, the set of agents commented above, error control, predefined interaction processes and a simple scheduling method which can be adapted or easily replaced by more specific scheduling algorithms. The tool has been developed using the JAVA implementation language and JADE [9, 19] platform for agent creation and management.

This framework can be considered as a flexible skeleton of a production programming toolkit which can be adapted with the suitable scheduling algorithms, product specifications and material definitions as needed. The FAST development process tries to make easier the developer work. The different steps of this process are (see Figure 3):

A. *Resources/Plant/Materials Definition*: the developer must define the structure of the production plant, the available resources and the materials used in the production process. The process of acquiring knowledge from the given domain is described in this stage, which applies traditional methods such as analyses of domain texts, expert interviews, and questionnaires. The terminology used by domain experts is defined in this stage. FAST employs an object/relational persistence and query service which provides transparent persistence keeping the resulting application portable to all SQL databases. The object/relational mapping tool requires metadata that governs the transformation of data from one representation to the other (and vice versa). The mapping metadata is declared in standard XML text files.
B. *Scheduling Process Definition*: FAST offers a simple scheduling algorithm implemented as a plug-in. Thus, it is a flexible system and any change in the algorithm applied to implement the scheduling process, or even the introduction of new algorithms, does not affect the whole FAST system functionality.
C. *Events/Heuristics Definition*: FAST needs a global and updated state view of the production plant and each of the machines involved in the manufacturing process. This state view allows the system to detect events that affect the current execution of a schedule in a specific plant. To detect and control these events in a suitable way, the Schedule Execution Monitor Agent must be completed with the definition of these problematic situations. The suggested toolkit incorporates examples of how to update several methods used by the Schedule Execution Monitor Agent.

[2] http://www.fipa.org

D. *Output Interface*: The system developer may need to implement specific interfaces for the environment under development. FAST basically produces an output formed by the programmed (or re-programmed) orders using a XML text file. FAST also offers a simple visual interface which can be easily reimplemented and adapted for specific purposes.

The overall process is conceived as an iterative process. The basic idea behind this iterative development is to develop the FAST system incrementally, allowing the developer to take advantages of what has been learned through this development process.

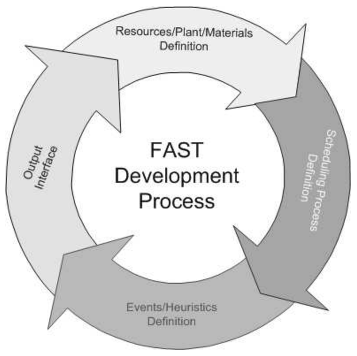

Resources/Plant/Materials Definition
Scheduling Process Definition
Events/Heuristics Definition
Output Interface
FAST Development Process

Fig. 3. FAST Development Process

4 Applying FAST

The FAST toolkit has been employed to obtain, in a easy and fast way, a system that improves the current production programming process in a Ceramic Tile Factory. The main objective is to make it more reactive, dynamic and automatic, so then the factory enhances its production system and customer services.

The Ceramic Tile sector is very competitive. This competition is finally reflected in an increase of the variety of products and services, together with a decrease of the production costs. Moreover, the improvement of the customer service is also needed to stand up to the rising foreign enterprises. In this way, in the Ceramic Tile Factory under study, different products with diverse sizes, designs and compositions are produced. Moreover, the factory has to deal with two kinds of clients: building firms and wholesalers.

Several business processes can be distinguished in a Ceramic Tile Factory. Firstly, a design department defines which ceramic products will be produced in the current season. Then, a prediction of sales and orders forecast is taken by the commercial department, based on historical sales data, orders, etc. Later, medium term production orders are defined, sequencing the different product lots to be produced. This sequence configures the *Master Plan*, which is normally used as the major input data to generate the production programming, that includes activities such as determining start time and resource allocation for a specific lot production. The production department uses *Master Plan*, information about *Raw Materials* availability, *Plant Status*, etc., to generate the production programming schedule. Finally, all tasks related with the production and final storage of the different product lots are carried out. Usually, the Ceramic Tile production process (Figure 4) is represented as a three stage hybrid flow shop with sequence dependency in which three stages can be identified: press and glass lines (first stage), kiln (second stage), and classification and packed lines (third stage) [2]. Each stage is a productive phase with different times, resources and objectives. Finally, the commercial department sells factory products and manages orders from clients.

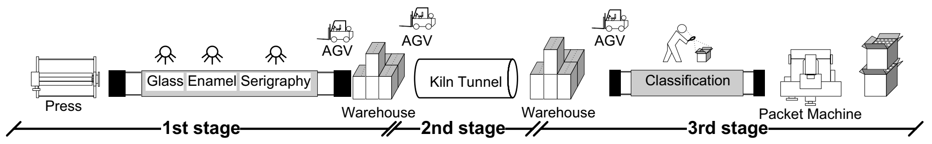


Fig. 4. Ceramic Tile Production Process: three stage hybrid flow shop

Notice that in a job shop manufacturing environment, things rarely go as expected. An optimal schedule might become unacceptable because of unforeseen dynamic situations on the shop floor (unavailability of resources and materials, power system failures, new jobs, etc). If so, a new schedule must be generated to restore performance.

4.1 Implementation

In this section, we explain the main features of the implemented prototype for the production scheduling of a ceramic tile factory. The toolkit has been successfully employed for this specific environment.

The prototype development process has followed these steps:

A. Resources/Plant/Materials Definition. In this step the knowledge acquisition and conceptualization is done. In our case, an analysis of the system processes was firstly performed on the real factory to collect a glossary of concepts (classes) for the domain. Secondly, an expert interview was conducted. Then, the domain concepts, instances, relationships and properties were identified, presented and associated with domain terms. After this, the ontology (figure 5) was formally represented using the Ontology Web Language (OWL) [38] and employing Protege [34] as the ontology toolkit. This stage involves the formalization of each term and the constraints used by the ontology. Terms are represented through classes, relations, functions and instances.

As a result, there exists three main databases: the order database which sets the production goals; the product definition database which explains the activities and resources needed to the manufacturing of an specific ceramic tile; and the plant database which specifies the available resources and provides information about the state of the plant.

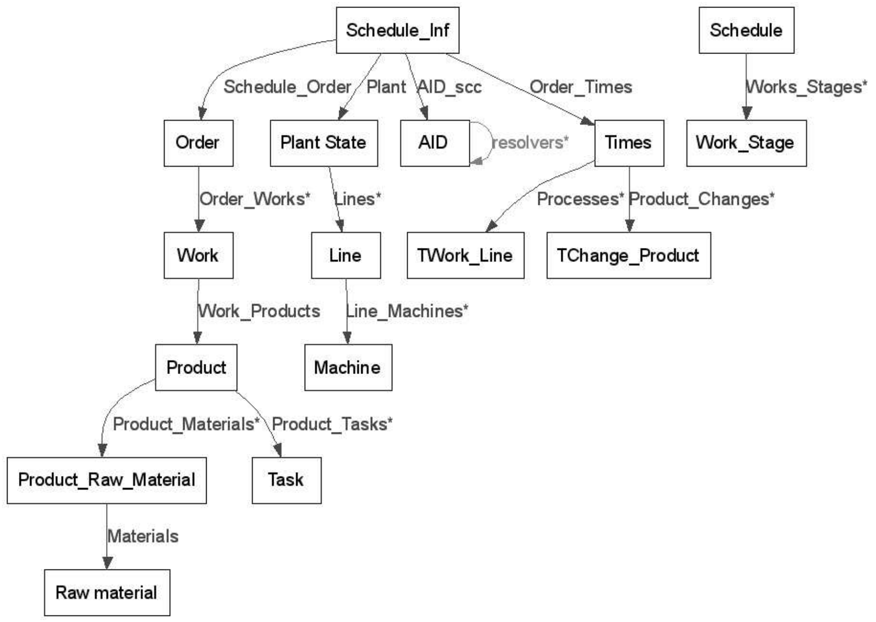


Fig. 5. Ontology concept example for the production scheduling of a ceramic tile factory

B. Scheduling Process Definition. After the manufacturing data model is completed, it is necessary to identify the set of choices the scheduler must make to derive a schedule or schedules that will meet the order deadlines. In this case, a group of agents in charge of the schedule creation/modification processes has been modelled. This agent group creates a feasible schedule for the workshop taking into consideration the existing factory restrictions. It is composed of four agents: one coordinator agent and three scheduling stage agents. These stage agents are specialized in constructing partial schedules for each production step (glazing, kiln and classification), taking into account its particular constraints and requirements. For this purpose, they dispose of a set of algorithms that are capable of obtaining the best objective function results. So, each stage agent executes simultaneously different suitable algorithms under diverse circumstances, selecting the most appropriate result.

Following, we present a brief description of the agents mentioned above.

The *Glazing Agent* faces the problem of high sequence dependent setup costs in the glazing production lines. Therefore, it aims to minimize those changes. Besides, the stability of this particular stage is of high importance because of the stage dependency on external providers and a high number of human resources. Additionally, the *Glazing Agent* should consider that not all jobs can be processed on all machines. It uses two algorithms: one based on dispatching rules, and the other one on set recovering techniques.

The *Kiln Agent*'s goal is to minimize changes of the kiln's temperature curve. It uses three algorithms, one based on dispatching rules, another on genetic algorithms and the third one on constraints propagation.

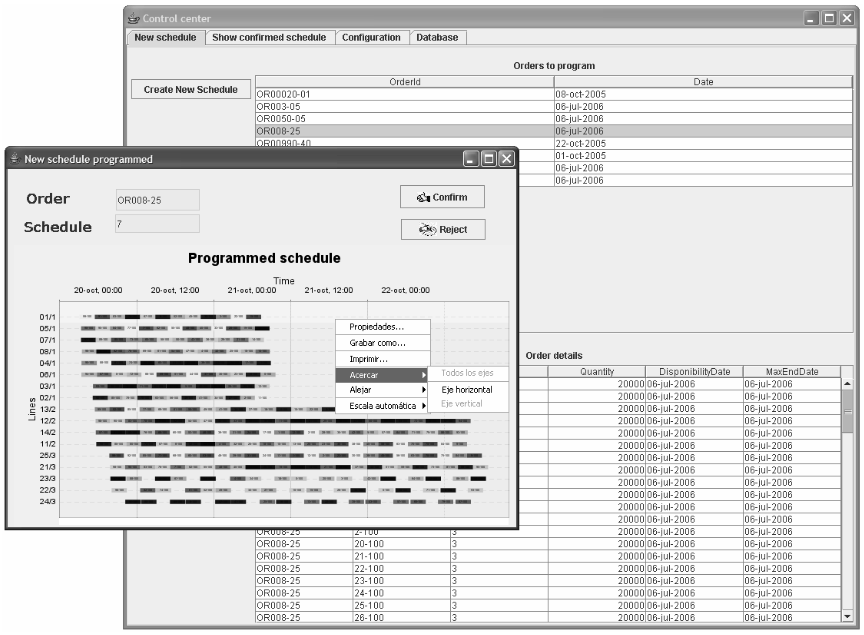
The *Classification Agent* aims to minimize the job processing times in order to reduce the number of human resources. It uses two algorithms: one based on dispatching rules and the other one based on genetic algorithms.

Finally, the *Coordinator Agent* aims to obtain feasible production schedules for the whole workshop while considering the jobs proposed by the *Lot Planner* agent and the partial schedules for each production stage. It contains algorithms to prevent infeasibility whenever this is possible. If not, it initiates a negotiation process with each stage agent, supervising them. The *Coordinator Agent*'s production programming is based on a predictive-reactive strategy (it does a foreseen schedule and reacts to events) and a mixed rescheduling policy (cyclic and event driven). The rescheduling process always takes into account the partial program performance and its longtime stability. Therefore, the *Coordinator Agent* is in charge of offering the scheduling service to the rest of agents in the system, representing the scheduling group.

C. Events/Heuristics Definition At this point, it is necessary to identify events that could affect the current execution of a schedule in the ceramic tile plant. The monitorized events can cause a simply schedule modification (solved by updating predicted start or end times) or a rescheduling (revisiting some or all of the scheduling decisions). This event identification covers situations like equipments going out of service, new orders arrival, or possible production deadlines non-fulfillment due to delays in process completion times. The schedule updating is at least as important as generating the original schedule. Many existing scheduling systems have been discarded because those systems have failed to seriously address updating.

The main types of events identified are the following: (i) variations in process stage duration; (ii) business orders or production orders that may be cancelled or added; (iii) manual adjustments, as scheduler users must be able to manually edit the schedule; and (iv) unexpected events, such as breakdowns in plant resources, incorrect or incomplete order information, lack of materials. The scheduling system must let managers reschedule after unexpected events to minimize disruption.

D. Interface We have developed an own interface for our problem, taking into account specific features of a ceramic tile factory. The new GUI let users inspect the proposed schedule from different views and edit it as necessary. Figure 6 shows a line oriented view of different programmed orders. The FAST toolkit architecture makes it easy for developers to add new views to meet the needs of particular applications.



Order: OR008-25

Schedule: 7

Orders to program

OrderId	Date
OR00020-01	08-oct-2005
OR003-05	06-jul-2006
OR0050-05	06-jul-2006
OR008-25	06-jul-2006
OR00990-40	22-oct-2005
	01-oct-2005
	06-jul-2006
	06-jul-2006

Order details

			Quantity	DisponibilityDate	MaxEndDate
			20000	06-jul-2006	06-jul-2006
			20000	06-jul-2006	06-jul-2006
			20000	06-jul-2006	06-jul-2006
			20000	06-jul-2006	06-jul-2006
			20000	06-jul-2006	06-jul-2006
			20000	06-jul-2006	06-jul-2006
			20000	06-jul-2006	06-jul-2006
			20000	06-jul-2006	06-jul-2006
			20000	06-jul-2006	06-jul-2006
			20000	06-jul-2006	06-jul-2006
			20000	06-jul-2006	06-jul-2006
			20000	06-jul-2006	06-jul-2006
OR008-25	2-100	3	20000	06-jul-2006	06-jul-2006
OR008-25	20-100	3	20000	06-jul-2006	06-jul-2006
OR008-25	21-100	3	20000	06-jul-2006	06-jul-2006
OR008-25	22-100	3	20000	06-jul-2006	06-jul-2006
OR008-25	23-100	3	20000	06-jul-2006	06-jul-2006
OR008-25	24-100	3	20000	06-jul-2006	06-jul-2006
OR008-25	25-100	3	20000	06-jul-2006	06-jul-2006
OR008-25	26-100	3	20000	06-jul-2006	06-jul-2006

Fig. 6. New Creation Process Interface

4.2 Evaluation and results

In order to evaluate this experimental simulation prototype, different simulation tests have been executed in which the operating conditions have been analyzed. These tests give a measure of the system reliability, robustness, flexibility and efficiency not only under normal conditions, but also under nonstandard operating circumstances. In this section some of these tests are explained. More specifically, tests on functionality and efficiency of the system; concurrent programming capacity; system reaction to unexpected events such as machine failures; and agent failures are carried out.

The objective of the first test was to check the correct functionality and efficiency of the system, so many trials for measuring the necessary time to program a new schedule were done. The generated instances were characterized by the order size (number of works that compose an order). When the size increases, the system has to manage more information so the whole process consumes more time. This situation is shown in Figure 7. Although time increases, the system can manage the orders independently of their size and with a little and unimportant delay, specially from the point of view of the tile world.

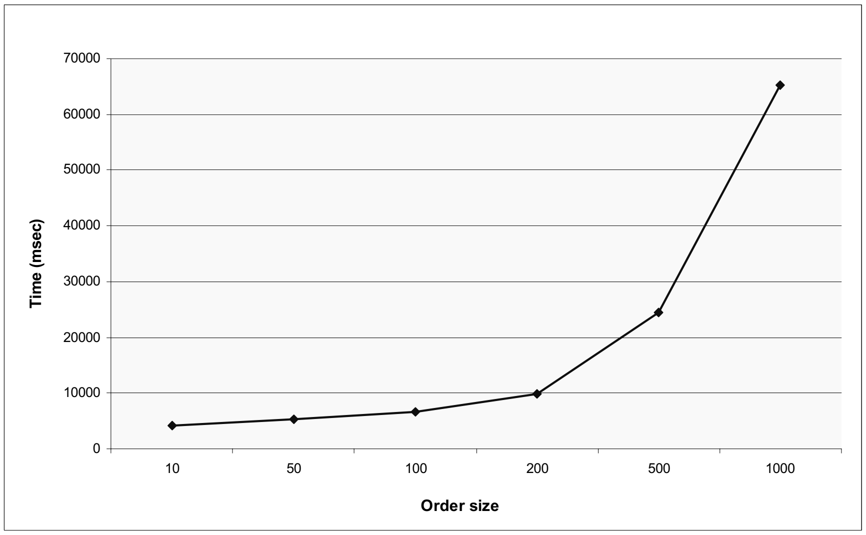


Fig. 7. Time needed for a new schedule creation process, depending on the order size

In the following test, the concurrent programming capacity of the system is analyzed, as the user is able to launch several orders to be programmed at the same time. For each new order launched there is an independent process in charge of it, so all orders are processed concurrently. Figure 8 shows a graphic which represents the needed time to program one specific order when there are one, two, four or ten programming processes launched at the same time. As shown, this time does not increase significantly when the user programmes other orders concurrently. This is possible thanks to algorithm and process concurrency.

On the other hand, the prototype reacts fast, dynamically and automatically when an event happens. For example the system detects when a work is delayed. If one work should be finished accordingly to the confirmed schedule, but has not finished yet, then the user receives a notification. It is very important to have a fast reaction to this situation because it means that there are failures in the confirmed schedule. Figure 9 shows the needed average time to detect and inform the user about a delayed work. As expected, the graphic shows that when the number of works delayed at the same time increases, the reaction time augments.

Another interesting event to consider is a line or machine failure. Just like before, it is very important to react faster to this kind of events. Thus, if there is any work programmed in this line or machine, then it should be reprogrammed and

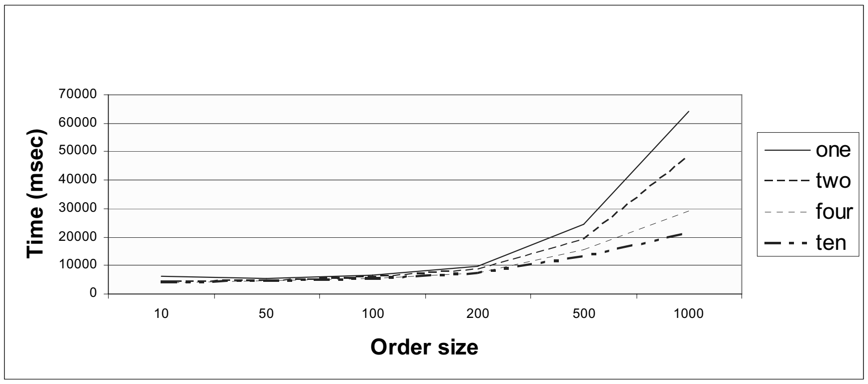


Fig. 8. Concurrent programming capacity of the system, when several programming processes are launched concurrently

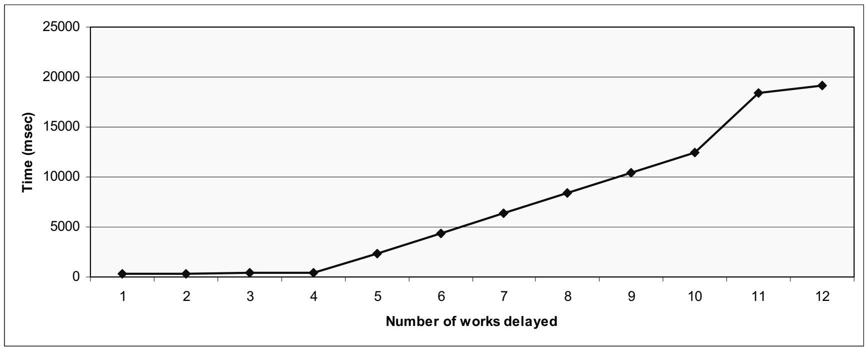


Fig. 9. Needed time for detection of delayed works

user must also be informed. The *Plant Wrapper* agent, who has all information about plant states, detects this failure and sends this information to the *Scheduler Execution Monitor* (SEM) agent. The SEM evaluates whether there is any work that needs to be reprogrammed. If not, the process finishes. Otherwise, the *Manager* agent is informed and the confirmed schedule is reprogrammed, taking into account the failed line or machine. In our test, lots of trails were launched to determine how much time each part of the process needed. The results (figures 10 and 11) shown some variability depending on the state of the system and on which processes were running at that current moment. The way to reduce this variability is by using different threads, but this solution was not implemented because the needed time for the production programming process is rather low, even in the worst case.

In Figure 10 and 11, there are some significant results. These graphics represent the time elapsed since an error happens until the *Plat Wrapper* agent detects it (*"PW detects"*); then until the SEM agent checks whether the confirmed schedule must be reprogrammed (*"SEM evaluates"*); and finally until the *Manager* agent receives notification and launches the reprogrammation process (*"Manager launches reprogrammation"*). When the error does not affect the confirmed schedule (Figure 10), the process usually takes around 0.02 seconds from the failure happening until the SEM decision that the error does not affect. But as shown in figure 10, when the Plant Wrapper or the SEM agents are busy, the needed time increases up to 0.25 seconds. On the other hand, when the error affects the confirmed schedule (Figure 11), the process usually takes around 0.32 seconds from the failure happening until the Manager launches the reprogrammation. But like before, there are some cases when time increases because agents are busy (tests 1, 6 and 9 in figure 11).

The created prototype is very flexible and adaptable. It allows adding, modifying and deleting agents during execution and also changing the scheduling algorithm or even using several algorithms currently, each of them in one active *Scheduler* agent. Once the schedules are programmed, each Scheduler agent provides the user with its Gantt diagram so then user decides which schedule must be confirmed and saved in the database. In order to check this system functionality and measure if the needed time to create a new schedule increases too much when the system employs several scheduling algorithms, some trials where done launching one, two and four active Scheduler agents each time. The results can be seen in Figure 12. As expected, the prototype is able to work with several scheduling algorithms at the same time and this does not affects too much the process time because all algorithms are executed concurrently.

Finally, when an agent fails, the prototype detects it and launches another agent to replace the failed one. This functionality gives the system great robustness and reliability because not only it setups the agent configuration, but it also is able to control that all launched processes finish successfully. For example if the Scheduler agent fails during a new scheduling creation process, then when new Scheduler agents will be active again, this same process will be automatically launched, allowing the schedule to be created successfully. The time elapsed

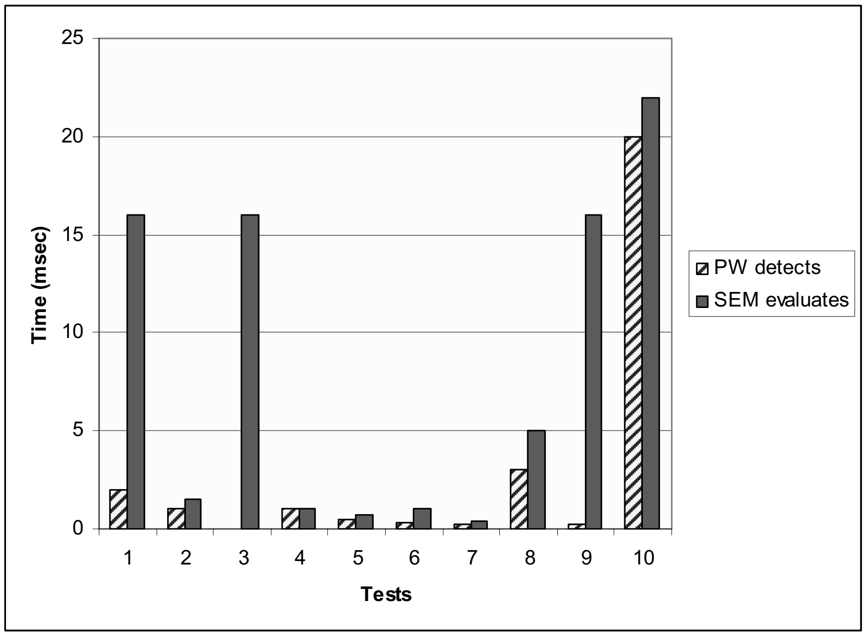


Fig. 10. Needed time for error recovery when error does not affect the calculated schedule

from an agent failing till it is operative again ranges from 0.10 to 3 seconds. This variability can be shown in Figure 13, in which some trials displayed. This variability is due to the exact time in which the agent fails. The system checks that all agents are alive in a periodic way, so it detects the fail after 2 seconds at the most.

5 Conclusions

In this paper, we have answered to the current demand for universal and flexible tools that enable improving the production scheduling complex problem in manufacturing systems, by means of the proposed FAST tool, which is a Flexible and Adaptive Scheduling Tool based on the multi-agent system paradigm for a manufacturing environment. It can be considered as a configurable scheduling tool which supports the creation, modification and supervision of production programming schedules in a easy and fast way.

We have also applied the FAST framework to an specific application example in a ceramic tile factory, obtaining a suitable prototype for the production programming process of this factory in a satisfactory way. The created prototype is reliable, robust, flexible and efficient not only under normal conditions, but also under nonstandard operating circumstances, as it is shown in the evaluation tests.

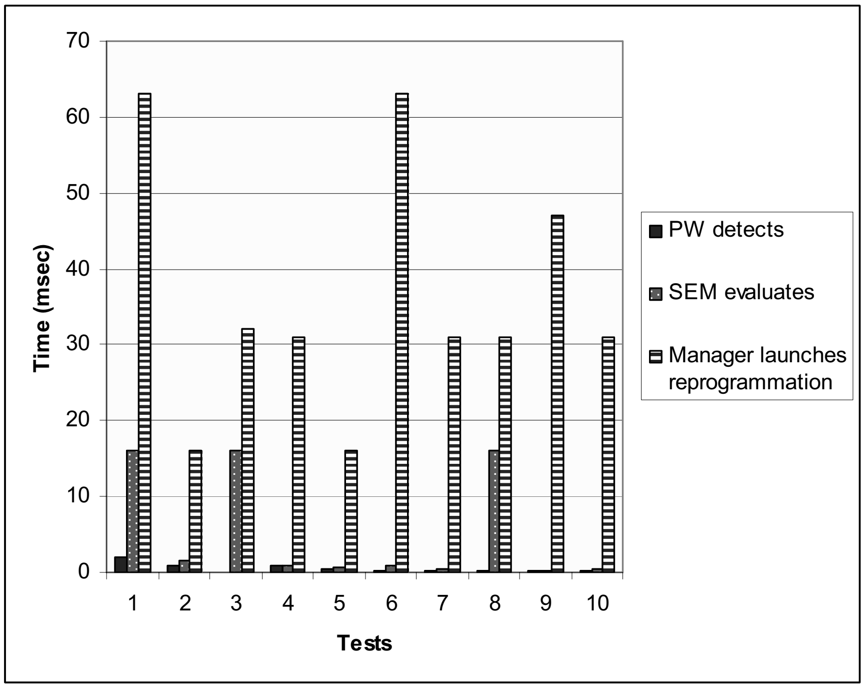


Fig. 11. Needed time for error recovery when error affects the calculated schedule

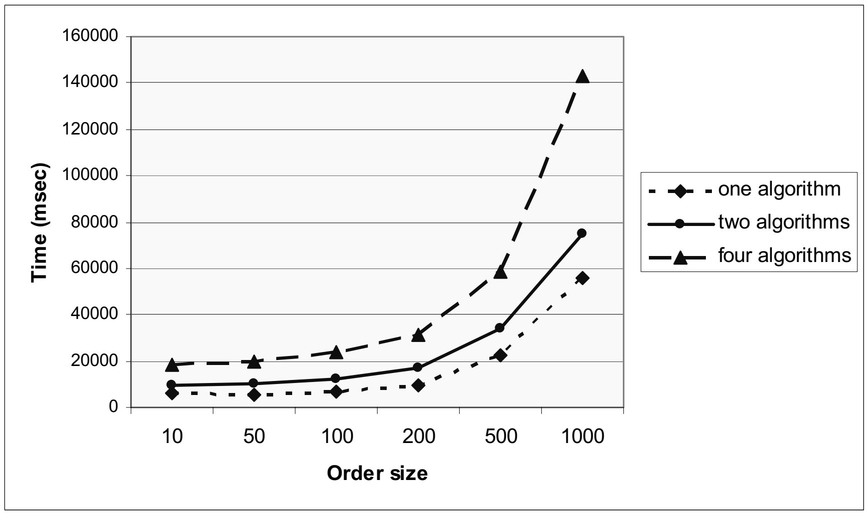


Fig. 12. Needed time for creating new schedules when using several scheduling algorithms

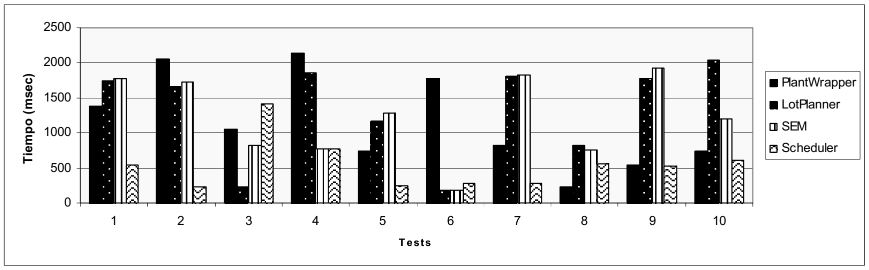


Fig. 13. Time elapsed from agent failure till agent operativeness

As future works, the FAST tool should be extended with new utilities. Currently, we are adding new pre-built scheduling algorithms into the toolkit. Moreover, we are improving the interface definition stage, providing the user with the possibility of modifying the proposed schedules on-line.

6 Aknowledgments

This work is partially supported by the GV06/315 project of the Valencian government and by the Polytechnic University of Valencia under grant PII-UPV 5574.

References

1. J. Agre, G. Elsley, D. McFarlane, J. Cheng, and B. Gunn. Holonic Control of Cooling Control System. In *Proceedings of Rensslaers Manufacturing Conference*, 1994.
2. C. Andrés. *Production Scheduling in Hybrid Flow Shop with Sequence Dependent Setup Times. Models, Methods and Algorithms. A Ceramic Tile Enterprise Application.* PhD thesis, Universidad Politécnica de Valencia, 2001.
3. A. Babayan and D. He. Solving the n-job 3-stage flexible flowshop scheduling problem using an agent-based approach. *International Journal of Production Research, Taylor & Francis Ltd.*, 42(4):777–799, 2004.
4. A. Bengoa. An Aproach to Holonic Components in Control of Machine Tools. *Annals of CIRP*, 45(1), 1996.
5. G. Biswas, B. Sugato, and A. Saad. Holonic Planning and Scheduling for Assembly Tasks. *TR CIS-95-01, Center for Intelligent Systems, Vanderbilt University*, 1995.
6. J. Brown and B. McCarragher. Maintenance resource allocation using decentralised cooperative control. Technical report, ANU, 1998.
7. S. Bussmann. An Agent-Oriented Architecture for Holonic Manufacturing Control. *Proc. of 1st Int. Workshop on Intelligent Manufacturing Systems, EPFL*, pages 1–12, 1998.
8. S. Deen. A cooperation framework for holonic interactions in manufacturing. *In Proceedings of the Second International Working Conference on Cooperating Knowledge Based Systems (CKBS'94*, 1994.

9. Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADE - A FIPA-compliant Agent Framework. In *Proc. of the PAAM'99*, pages 97–108, Apr. 1999.
10. K. Fischer. An Agent-Based Approach to Holonic Manufacturing Systems. *in L. M. Camarinha-Matos, H. Afsarmanesh and v. Marik (Eds.) Intelligent Systems for Manufacturing. Multi-Agent Systems and Virtual Organisations*, pages 3–12, 1998.
11. M. Fletcher and M. S. Deen. Fault-tolerant holonic manufacturing systems. *Concurrency and Computation: Practice and Experience*, 13(1):43–70, 2001.
12. N. Gayed, D. Jarvis, and J. Jarvis. A Strategy for the Migration of Existing Manufacturing Systems to Holonic Systems. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pages 319–324, 1998.
13. L. Gou, T. Hasegawa, P. Luh, S. Tamura, and J. Oblak. Holonic Planning and Scheduling for a Robotic Assembly Testbed. In *Proceedings of the 4th Rensselaer International Conference on Computer Integrated Manufacturing and Automation Technology*, 1994.
14. L. Gou, P. Luh, and Y. Kyoya. Holonic Manufacturing Scheduling: architecture, cooperation, mechanism, and implementation. *Computers in Industry*, 37:213–231, 1998.
15. T. Hasegawa, L. Gou, S. Tamura, P. Luh, and J. Oblak. Holonic Planning and Scheduling Architecture for Manufacturing. *In Proceedings of the 2nd International Working Conference on Cooperating Knowledge-based Systems*, 1994.
16. T. Heikkila, M. Jarviluoma, and J. Hasemann. Holonic Control of a Manufacturing Robot Cell. Technical report, VTT Automation, 1995.
17. T. Heikkila, M. Jarviluoma, and T. Juntunen. Holonic Control for Manufacturing Systems: Design of a Manufacturing Robot Cell. *Integrated Computer Aided Engineering*, 4:202–218, 1997.
18. P. R. HMS. *HMS Requirements*. HMS Server, http://hms.ifw.uni-hannover.de/, 1994.
19. JADE. Java Agent DEvelopment Framework. *http://jade.tilab.com/*, 2006.
20. A. Marcus, T. Kis Vancza, and L. Monostori. A market approach to holonic manufacturing. *Annals of CIRP*, 45:433–436, 1996.
21. F. Maturana and D. Norrie. Distributed decision-making using the contract net within a mediator architecture. *Decision Support Systems 20*, pages 53–64, 1997.
22. D. McFarlane and B. S. *Agent-Based Manufacturing. Advances in the Holonic Approach*, chapter Holonic Manufacturing Control: Rationales, Developments and Open Issues, pages 301–326. Springer-Verlag, 2003.
23. K. Miyashita. "camps: A constraint-based architecture for multi-agent planning and scheduling". *Journal of Intelligent Manufacturing*, 9(2):147–154, Apr 1998.
24. A. Ng, R. Yeung, and E. Cheung. HSCS - the design of a holonic shopfloor control system. In *Proceedings of IEEE Conference on Emerging technologies and Factory Automation*, pages 179–185, 1996.
25. V. "Parunak. Workshop report: Implementing manufacturing agents. *National Center for Manufacturing Sciences*, 1996.
26. M. Pechoucek, J. Vokrinek, and P. Becvar. Explantech: Multiagent support for manufacturing decision making. *IEEE Intelligent Systems*, pages 1541–1672, Jan/Feb 2005.
27. C. Ramos. A holonic approach for task scheduling in manufacturing systems. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 2511–2516, 1996.
28. L. Rannanjarvi and T. Heikkila. Software Development for Holonic Manufacturing Systems. *Computers in Industry*, 37(3):233–253, 1998.

29. A. Saad, G. Biswas, K. Kawamura, M. Johnson, and A. Salama. Evaluation of Contract Net-Based Heterarchical Scheduling for Flexible Manufacturing Systems. *In Proceedings of the 1995 International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 310–321, 1995.
30. W. Shen and D. Norrie. Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. *Knowledge and Information Systems, an Internatinal Journal*, 1(2):129–156, 1999.
31. N. Sugimura, M. Hiroi, T. Moriwaki, and K. Hozumi. A study on holonic scheduling for manufacturing systems of composite parts. In *Proceedings of Japan/USA Symposium on Flexible Manufacturing*, pages 1407–1410, 1996.
32. P. Tanaya, J. Detand, and J. Kruth. Holonic Machine Controller: A Study and Implmentation of Holonic Behaviour to Current NC Controller. *Computers in Industry*, 33:325–333, 1997.
33. P. Tanaya, J. Detand, J. Kruth, and P. Valckenaers. Object-Oriented Execution Model For a Machine Controller Holon. *European Journal of Control*, 4(4):345–361, 1998.
34. S. University. Protege ontology editor. *http://protege.stanford.edu.*
35. P. Valckenaers, H. Van Brusel, L. Bongaerts, and F. Bonneville. Programming, Scheduling and Control of Flexible Assembly Systems. *Computers in Industry*, 26:209–218, 1995.
36. S. Valero, E. Argente, A. Giret, V. Julian, and V. Botti. Goodness and lacks of mas methodologies for manufacturing domains. In *LNAI 3690. CEEMAS'05*, pages 645–648. Springer, 2005.
37. H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. Reference Architecture for Holonic Manufacturing Systems: PROSA. *Computers In Industry*, 37:255–274, 1998.
38. W3C. Owl web ontology language overview. *http://www.w3.org/TR/owl-features/*, 2004.