www.revistadyo.com

Gaia Analysis and Design of a Multi-agent-based Shop Floor Control System
Análisis y Diseño Gaia de un sistema multiagente de control de planta

José Alberto Araúzo [1], Ricardo del Olmo [2] y Juan José Laviós [2]

[1] Departamento de Organización de Empresas y CIM. Escuela de Ingenierías Industriales. Universidad de Valladolid. Pº del Cauce, 59, 47011, Valladolid, España.
[2] Departamento de Ingeniería Civil. Escuela Politécnica Superior (Edificio La Milanera). Universidad de Burgos. C/ Villadiego s/n, 09001, Burgos, España.
arauzo@eii.uva.es, rdelolmo@ubu.es, jjlavios@ubu.es

Fecha de recepción: 21-10-2013
Fecha de aceptación: 4-11-2014

Abstract: To achieve competitiveness in the today's manufacturing plants; many researchers suggest agent based manufacturing systems, with scheduling and control activities distributed on several entities called agents. To develop this kind of systems some promising agent-oriented software methodologies, as Gaia, have been proposed. They provide the concepts and procedures to define the main features of multi-agent systems. In this paper we show how we used the Gaia methodology to analyze and design a multi-agent-based shop floor control system that has been further implemented by JADE. We also show preliminary computational experiments that show as our approach has a good performance in flexible environments.

Key Words: shop floor control, scheduling and control, multi-agent systems, software engineering, Gaia analysis and design.

Resumen: Para mejorar el desempeño de las actuales plantas de producción, muchas investigaciones sugieren la implementación de sistemas de fabricación basados en agentes, donde las actividades de programación y control de la producción están distribuidas sobre varias entidades denominadas agentes. Para desarrollar este tipo de sistemas, durante las dos últimas décadas se han propuesto varias metodologías que proporcionan los conceptos y procedimientos para definir las principales características de los sistemas multiagente. En este artículo se muestra el proceso de análisis y diseño de un sistema multiagente de control de planta mediante Gaia, una de las metodologías de desarrollo de sistemas multiagente más aceptadas en la actualidad. El sistema ha sido posteriormente implementado mediante JADE. También mostramos algunos resultados experimentales que muestran que el desempeño de nuestra propuesta en entonos flexibles es satisfactorio.

Palabras clave: control de planta, programación y control de la producción, sistemas multiagente, ingeniería de software, análisis y diseño Gaia.

1. Introduction

Agent based manufacturing systems apply the concepts, methods and techniques introduced by the multi-agent systems (MAS) computing paradigm, towards the development of more competitive production systems to the to-day's market.

According to this paradigm, an agent is an autonomous and flexible software entity that is situated in an environment, and is capable to act autonomously to achieve its design objectives (Wooldridge and Jennings 1995). Since other agents can be also be situated in the same environment, each one should be able to interact to each other by means of communication, that is to say, they should possess social abilities. These characteristics offer the software developers the opportunity to create collaborative agent communities that look for a global shared objective. These features provide methods to design and implement distributed complex systems in a natural way.

Agent based manufacturing systems try to integrate programming, execution and control activities, in a software made up of a group of autonomous, proactive and reactive entities (agents) which can interact using their social abilities. Although the system could be composed of agents performing dedicated tasks as programming, monitoring, or on-line control, the most usual composition in agent based manufacturing systems is physical. In a physical composition the control software structure is based on the real manufacturing environment. Environment changes are easily implemented in the software and the system improves its flexibility, scalability and reconfigurability (Giret 2008).

In this paper we show the analysis and design of a multi-agent system for shop floor control (SFC). The

SFC is a manufacturing subsystem that includes the production planning (scheduling) function and the control on the operational level. The work involve solving some questions as -what kinds of agents should the system have?-, -which objective should each agent have?-, -what should their behaviours be like?-, or -what should their interactions be like?-. We are going to show how we use the Gaia methodology to solve those questions. Gaia has been selected because of the following reasons: (1) it is quite easy to learn; (2) it proved to be flexible and robust; (3) it is a general methodology that can be useful for any multi-agent-based develop tool; and (4) it is quite accepted for the multi-agent developer community.

The rest of the paper is organized as follows: in section 2 and 3 we introduce respectively the agent based manufacturing systems and Gaia methodology; in section 4 we provide our system requirements and the resulting of the Gaia analysis phase; the resulting model of the design phase is presented in section 5; section 6 provides some computational experiments, and finally, we present the conclusions in section 7.

2. Agent based manufacturing systems

Current competitive markets force companies to shorten their product-life cycles, reduce time-to-market, increase product variety, while maintaining high quality and reducing costs. Consequently, manufacturing systems must become more flexible in order to cope with constant product changes and must exhibit more robustness in order to maximize equipment utilization. But flexible and robust manufacturing systems require an intelligent control that classical technologies cannot provide (Bussmann, Jennings, and Wooldridge 2004).

Software agents are a suitable technology to achieve the above requirements. A decentralized manufacturing system can be modelled following this paradigm, where resources are allocated dynamically by a distributed coordination process. Although, in this kind of systems, decisions may not be optimal, they are based on updated information which leads the system to a better performance.

Probably the first application of multi-agent concepts in manufacturing field was the prototype YAMS (Parunak et al.), a decentralized factory control system based on the Contract-Net Protocol (Smith 1980). Since then, many publications have emerged using agent-based technologies in different fields of manufacturing (Monostori 2006): engineering design (Brissaud and Zwolinski 2004 , Chen and Tseng 2005), process planning (Amara, Dépincé, and Hascoët 2004), production planning and resource allocation (Bruccoleri et al. 2005) and shop floor control (Leitão 2009).

Two kinds of agents appear in most of the works about production planning, resource allocation and shop floor control agent based approaches (Arauzo et al. 2003): the order agents that manage the order production and the resource agents that schedule and control the resource activities. In this type of systems, two main cooperation mechanisms have been used:

(1) Order agents agree with resource agents the execution of operations. The most common procedure to allocate operations to resources in a totally distributed way is the Contract Net Protocol (Parunak 1986, Hsieh 2008). But there are also another distributed approaches as pull-based systems (Ounnar and Pujo 2012), or the mechanisms proposed by Blanc et al. (2008) and Roulet-Dubonnet and Ystgaard (2011).

(2) An agent of centralized nature develops efficient production programs that are used as a guide for the decision-making of the other agents. Most sophisticated agent based architectures propose agents of centralized nature (coordinator agent) that collect information permanently about the orders and resources state. With this information, the coordinator agent develops or updates a global program that is used for the decision making in the order and resource agents (Bongaerst 1998). Within this approach we highlight the systems that use auctions as allocation mechanisms (Gou et al 1998, Wonga et al 2006, Leitão and Restivo 2008).

3. The Gaia methodology

Software development methodologies are frameworks to structure, plan, and control the process of developing information systems. Since the emergence of multi-agent systems, several methodologies have been proposed specifically to enable their development. Some of these methodologies are: MAS-CommonKADS (Iglesias and Garijo 2005), AUML (Odell et al. 2001), Ingenias (Pavón and Gómez-Sanz 2003) or Gaia. They provide the concepts, notations, techniques and guidelines to develop such systems.

Gaia, as many other agent-based methodologies, provides the two levels of design in MAS: the individual agent structure and the agent society (Zambonelli, Jennings and Wooldridge 2003). It defines the structure of MAS as a role model: MAS are a set of autonomous and interactive agents that live in an organized society in which each agent plays one or several roles and interact with other agents by means different protocols (Moraitis and Spanoudakis 2006).

The Gaia modelling methodology is a two-phase process: the analysis phase and the design phase in which the global structure and the agent details are specified (figure 1).

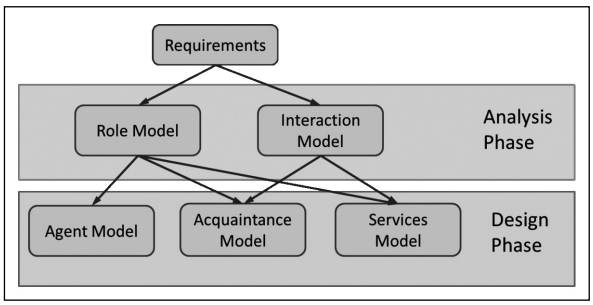

Figure 1
Models and relationships in the Gaia methodology

In the analysis phase two models must be defined: the role model and de interaction model. The role model identifies the roles that agents play within the MAS. Each role is characterized by four attributes: responsibilities, permissions, activities and protocols. Responsibilities determine the functionality and they can be classified into two types: liveness properties that determine the activities the role has to do, and safety properties that indicate something the role must prevent. Permissions represent the information the role can create, read or modify. Activities are a list of the tasks that the role performs without interactions. Protocols are a list interaction patterns. These protocols are detailed later, in the other model of the analysis phase: the interaction model, in which each protocol is described by four attributes (figure 2): the interaction initiator role, the responder role, the necessary information to initiate the interaction (input), and the resulting information (output).

The design phase concludes with three models: the agent model, the acquaintance model and the services model. The agent model identifies agent types in the system, the roles that each agent type plays, and the number of instances for each agent type. The acquaintance model is a graph that represents the agent relationship. Finally, the services model shows the services that are offered for each agent type. A service can be viewed as a function that the agent can perform. They are derived from the list of protocols, activities, responsibilities, and the liveness properties of the agent played roles.

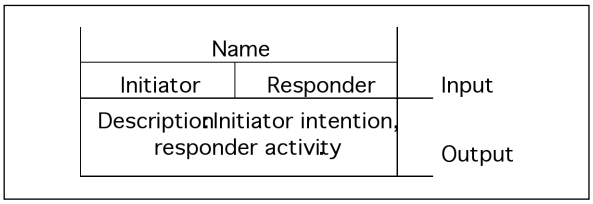

Figure 2
Protocol representation

4. Analysis phase

We have applied the Gaia methodology to the definition of a Shop Floor Control based on the Multi Agent Systems paradigm. In this section we present the resulting models of the analysis phase. To reach them we start with the requirements, and then, we followed the four steps proposed in (Wooldridge et al. 2000): (1) identifying the roles in the system, (2) for each role, identifying and document the associated protocols, (3) refining the role model using the interaction model, and (4) iterate stages 1,2 and 3.

4.1. Requirements

Shop floor control (SFC) is a system of computers and/or control tools used to schedule, dispatch and track the progress of orders through manufacturing system.

Our SFC has the following inputs: machine features (technological possibilities and capacities), items production process (operations and precedence relationship for each item) and manufacturing orders (item and due date). The SFC will be responsible for executing the manufacturing orders as efficiently as possible. To do so, it must: allocate and schedule tasks on machines, dispatch the operation execution to the machines, and monitor the status of the plant. Fur-

thermore, the system has to react to the disturbances and must reschedule when necessary.

Unlike traditional approaches, in our manufacturing system, operations are not initially allocated to any machine. The allocation is chosen by the system at runtime, depending on the efficiency of the different machines and their availability. The operation execution times are also calculated by the system for each machine in accordance with the operation parameters and the machine features.

Another feature of the proposed system is the scheduling procedure. In traditional systems, scheduling is done centrally, while in our system it is done in a distributed manner. We will use an auction-based method similar to that proposed by (Laviós et al. 2010 and Arauzo et al. 2013). Under this method, each work order creates a local schedule according to the prices of resource time units. These prices have been obtained previously through an auction that is driven by an auctioneer. Finally, local programs are integrated by the auctioneer.

4.2. Role Model

The analysis phase permits the identification of four external or user roles (Table 1) and nine system roles. The last ones are those that are played by the software agents and they will be detailed in the analysis process. They are described briefly in the table 2.

Table 1
User roles

Role	*Description*
MACHINEMANAGER	It sets machine parameters: capacity and technical possibilities and specifications
PROCESSDESIGNER	It sets products can be manufactured in the system. For each product, it designs their manufacturing processes (operations and precedence relationships).
PLANNER	It defines and dispatches orders. Moreover, it analyses the system efficiency.
SHOPFLOORMANAGER	It sets and monitors the system.

Table 2
System roles

Role	*Description*
MACHINEOPSPC	(Machine Operation Specification). It assists the user MACHINEMANAGER to set the machine. It also assists the role PROCESSDSASS for the process specification by stating whether a machine is able to perform operations and by calculating its duration.
PROCESSDSASS	(Process Design Assistant). It assists the user to define the process manufacturing. It asks the role MACHINEOPSPC for information about the possibility of performing operations in the machines and the durations of these operations.
PLANNERASS	(Planer Assistant) It assists the user PLANNER to define the production orders. It also calculates parameters for the system performance measurement.
ORDERDISPATCHER	(Order Dispatcher) It dispatches the orders defined by the PLANNER.
DISPLAYER	It displays de system state (orders and machine state) and its evolution.
PRICECALCULATOR	(Order Manufacturing Manager). It calculates the price of resource usage in each time unit of the scheduling horizon.
LOCALSCHEDULER	It schedules the order operations according the prices that the role PRICECALCULATOR has calculated.
ORDMANUFMANAGER	(Order Manufacturing Manager). It manages the execution of operations of each production order according to the schedule that is proposed by the role LOCALSCHEDULER. When an operation has to be executed, this role asks the role OPEXMANAGER for the execution. This role is also responsible of the precedence constraints compliance during the execution phase.
OPEXMANAGER	(Operation Execution Manager). It is responsible of the operation execution. It controls the physical machine.

All these roles should be detailed with their permissions, responsibilities, activities and protocols. For reasons of space, we only detail the role MACHINEOPSPC (figure 3) to illustrate the methodology. To distinguish activities and protocols, activities are denoted by underlining. Liveness responsibilities are specified via a liveness expression (Zambonelli, Jennings and Wooldridge 2003). They express the execution sequence of activities and protocols by means of operators such as "+", "•" or "w". In the figure, "X•Y" expresses "X followed by Y", "X|Y" expresses "X or Y occurs", "X+" expresses "X occurs 1 or more times" and "XW" expresses "x occurs infinitely often". The complete role model can be found in (Araúzo 2012).

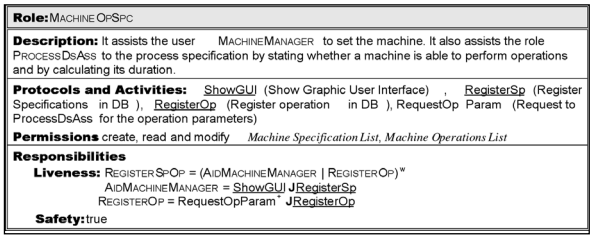

Figure 3
Properties of the role MACHINEOPSPC

4.3. Interaction Model

In an organization with multiple roles, interactions occur inevitably. In the role model these interactions or protocols have already appeared as part of the liveness responsibilities of the initiator role. In the figure 4, the protocol RequestOpParam of the role MACHINEOPSPC (figure 3) is described by using the special template shown in the figure 2. The main protocols of the model are described briefly in the table 3. For more information see (Araúzo 2012).

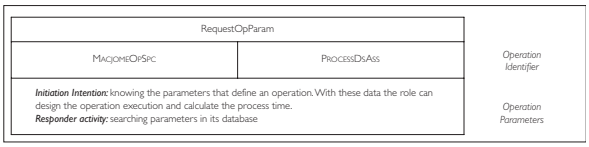

Figure 4
Representation of the protocol RequestOpParam

5. Design phase

During this phase, the agent model, the acquaintance model and the services model were achieved. The agent model, in which the roles are assigned to the agents, is presented in the table 4. To package several roles in a single agent, different criteria can be used: interaction with a single user role, same location in a network, or accessing to the same information.

Figure 5 shows the acquaintance model, which is derived from the role model, the interaction model and the agent model. It is the simplest model and shows

Table 3
Interaction model

Protocol	*Initiator/Responder*	*Description (Initiator Intention)*
RequestOpParam	MACHINEOPSPC/PROCESSDSASS	Knowing the parameters that define an operation to design the execution and calculate the process time.
RequestOpProc	PROCESSDSASS / PROCESSDSASS	Generating the operation execution process in a machine and knowing the process time.
RequestItems	PLANNER / PROCESSDSASS	Knowing the items with a defined process to create manufacturing orders.
RequestOrManuf	PLANNER / ORDERDISPATCHER	Creating the ORDMANUFMANAGER to manage the production order
RequestProcess	ORDERDISPATCHER/PROCESSDSASS	Knowing the order manufacturing process to create the ORDMANUFMANAGER.
RequestOpEx	ORDMANUFMANAGER/OPEXMANAGER	Executing an operation
RequestLocSch	PRICECALCULATOR/LOCALSCHEDULER	According to the proposed scheduling method, calculating the prices of resource usage. It is necessary to have the local programs of each order. With this protocol, the role PRICECALCULATOR gets those schedules.
RequestMachEv	DISPLAYER / OPEXMANAGER	Knowing the state changes of the machines to monitor the system. The DISPLAYER has to know the operation start time and conclusion time.

Table 4
Agent model

Agent	*Planner*	*Process Designer*	*Coordinator*	*Order Manager*	*Machine Manager*
Number of instances	1	1	1	Number of orders in the system	Number of machines
Roles	PLANNERASS	PROCESSDSASS	DISPLAYER ORDERDISPATCHER PRICECALCULATOR	ORDMANUFMANAGER LOCALSCHEDULER	OPEXMANAGER MACHINEOPSPC

the communication links between agent types. Its purpose is identifying potential communication bottlenecks.

The last model of the design phase is the service model. It identifies the services associated with each agent. The services model is too detailed to be included in this paper; it can be consulted in (Araúzo 2012).

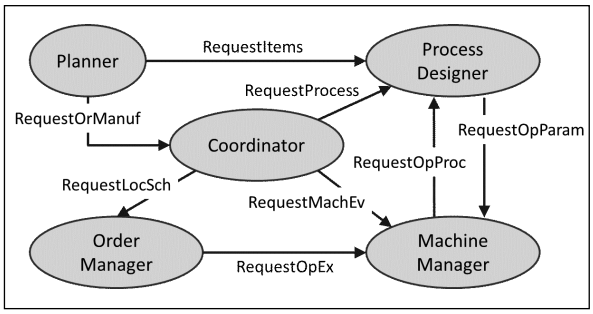

Figure 5
Acquaintance model. Arrows represent interaction protocols. They go from de initiator to the responder

6. Computational experiments

The system has been implemented by JADE and proved in different scenarios. The first scenario shows a simple case to illustrate the main features of the system, (see next subsection). This is followed by some more complex scenarios to evaluate the system performance. The analysis focuses on the ability of our multi-agent system to use the process flexibility to improve the shop floor performance.

6.1. Simple scenario

We consider a manufacturing system with three machines (M1, M2 and M3) in which tree orders must be manufactured. The orders are defined by the Planer agent by specifying the item (product), weight (delay cost) and due date (table 5). Each item has a production process that is defined by the Process Designer agent (table 6). Our SFC agent based system must schedule, execute, and control the production orders as efficiently as possible. More formally, this problem can be formulated as a job shop scheduling problem (JSSP) with three machines and three orders.

Table 5
Simple scenario orders

Order	*Item*	*Weight*	*Due Date (seconds)*
OrA	A	2	60
OrB	B	1	60
OrC	C	5	60

Table 6
Simple scenario process. The items are manufactured by means two or three operations. Each operation is specified by the machine that can execute the operation and its duration

Item (X)	*Process (operations)*					
	OpX1		*OpX2*		*OpX3*	
A	M3	40	M2	50	M1	20
B	M3	20	M1	50	-	-
C	M1	50	M2	40	M3	50

As we have explained above, the allocation procedure is based on an auction. An auctioneer (the Coordinator Agent) proposes prices for each time units of each machine from the current time to the end of the scheduling horizon, and order agents try to find

a set of time units of resources to execute their operations while incurring the minimum possible local cost. Continuously the auctioneer updates the price charged for the resources time units with the purpose of reducing resource conflicts and maximizing their revenue. By means of this procedure the system attempts to minimize a delay-based objective function [1].

$$T^2 = \sum_{i=1}^{N} w_i \text{ - } T_i^2$$

$T_l\, i$ = max [[(0, max ⊤ i [[($c_l ij$) – $D_l i$]]) → Delay of the orden i]]
w_i → Weight of the order i
D_i → Due date of the order i [1]
c_{ij} → Conclusion time of the operation j of the orden i
N → number of orders

The Figure 6 shows the system monitoring that is provided by the coordinator agent in the end of the experiment. It displays the tasks performed by each machine and the prices of the time units after finishing the simulation. In the upper area of the figures, the relative duality gap evolution is presented. Mathematically, the prices of time slots are the solution of the dual problem and the duality gap is a measure of the difference between the primal and dual objective functions, so it quantifies the quality of the solution (Laviós et al. 2010). A small relative duality gap means that the prices are representative of the system state; therefore a good solution is achieved. The lower part of the figures presents the resources charts. These charts show the tasks that each machine has performed (lower area of the resource charts) and the time units prices (upper area of the resource charts).

The Figure 7 shows the system monitoring for a flexible simple scenario derived from the above case. In this new scenario the operation OpC2 can be executed by the machines M1 or M2 in 40 seconds and the operation OpB2 can be executed by the machine M1 in 50 seconds or in the machine M2 in 60 seconds. Although the duration of the operation OpB2 is smaller in machine M1 than in M2, now the system has reallocated in real time this operation to M2. In this experiment, the orders OrA and OrB are concluded before than in the first experiment, so the objective function changes from 55200 to 48600. This shows that the system is capable to use the flexibility to improve in real time the global performance.

The prices we can see in the figures 6 and 7 show the importance of each machine at each unit of time. For instance, in the figure 7, the prices of resource M1 are very high during the first 100 seconds. This means that the machine is very valuable (bottle neck) at those moments. On the other side, prices of machine M3 are small, although it has been working in different operations during the experiment.

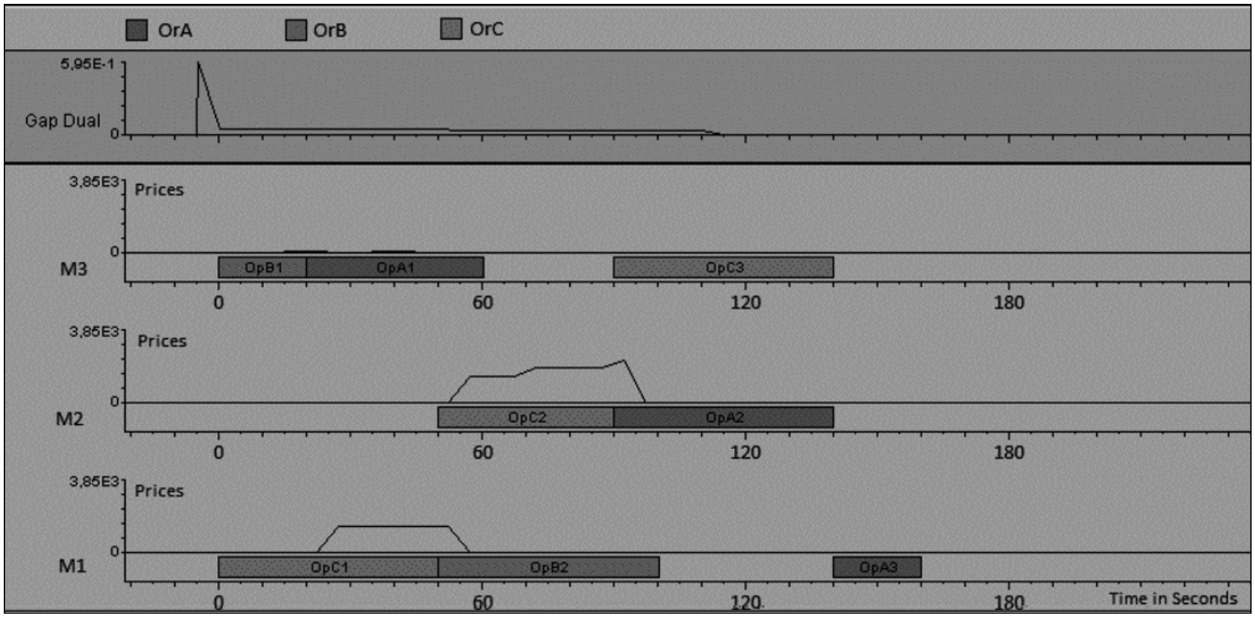

Figure 6
Gantt char of simple scenario. Opij denotes Operation /j/ of order Or/i/

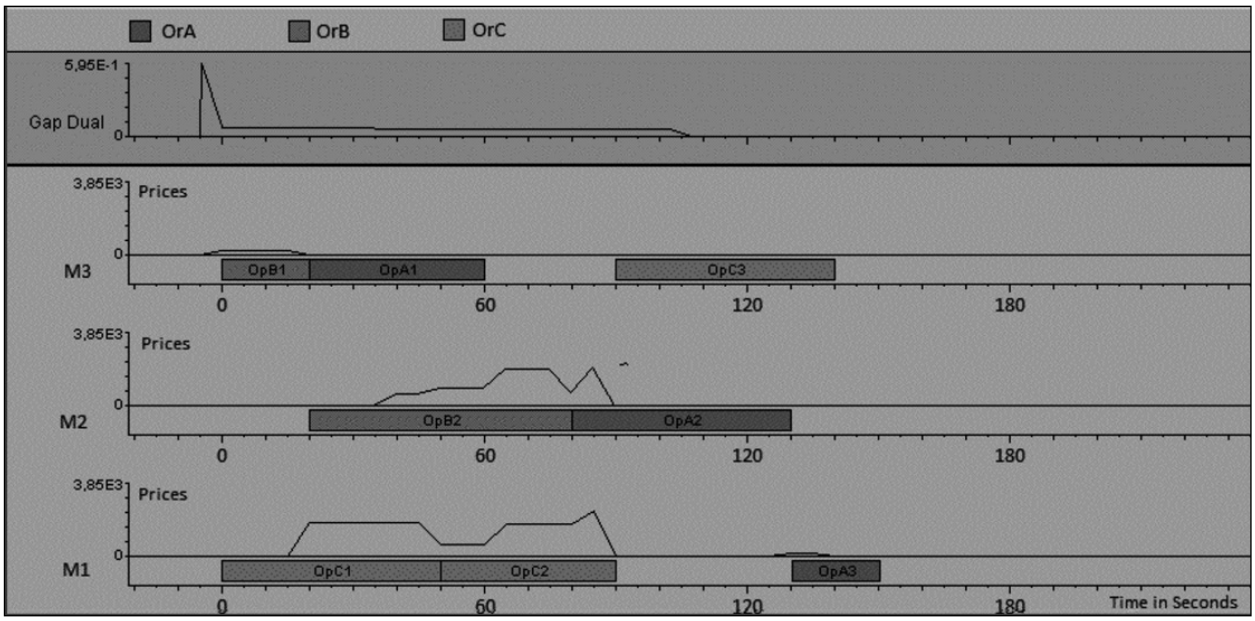

Figure 7
Gantt char of flexible simple scenario

6.2. Complex scenarios

In order to evaluate the system performance and its ability to manage the process flexibility efficiently we define four job shop cases with five machines and seven orders. Manufacturing processes have between three and five operations. For each operation, the machine is selected randomly among the five ones; and the duration is generated randomly between 20 and 140 seconds.

We have defined six versions of each problem with different degree of flexibility. This is done gradually by adding a new machine for the execution of some operations, increasing the degree of flexibility. To measure the degree of flexibility of a problem we use the following expression [2].

$$FI_p = \frac{1}{I_p} \sum_{i=1}^{Ip} f_{ip}$$

$FI_p \rightarrow$ flexibility index of problem p
$I_p \rightarrow$ number of orders of the problem p [2]
$f_{ip} = R_{ip} - 1 \rightarrow$ flexibility of order i of problem p
$R_{ip} \rightarrow$ number of possible routes of order i of problem p

For each case (problem and flexibility) we compare the results that we have obtained by our system and the results that we have obtained by a classical dispatching method with a slack-based heuristic, which is the most frequently used to deal with this kind of problems in dynamic and stochastic environments, thus it can be used as reference to evaluate our proposal. To schedule operations according to the dispatching procedure two steps are necessary: (1) if the operation can be executed in several machines we allocate the operation on most efficient and most available machine, and (2) when several operations can be scheduled in a machine we schedule the one that belongs to the most urgent order (the one with the least weighted slack per pending operation).

In the figure 8 we can see the value of the objective function [1] for each case (problem and flexibility) and procedure: Multi-Agent System (MAS) or Heuristic Dispatching (HD). We can see that the MAS procedure has always better performance than the HD procedure. Moreover, while the MAS procedure is able to use the process flexibility to improve the performance for every problem, the HDP procedure only does it for the problem 2.

7. Conclusions

In this paper we have presented the Gaia analysis and design process for a multi-agent system for shop floor control. This is the first phase of a development process that finished with a successful JADE implementation. In general, the process of developing our system has been agile and robust. Gaia allows an incremental development process, documenting the different models and facilitating the subsequent JADE implementation.

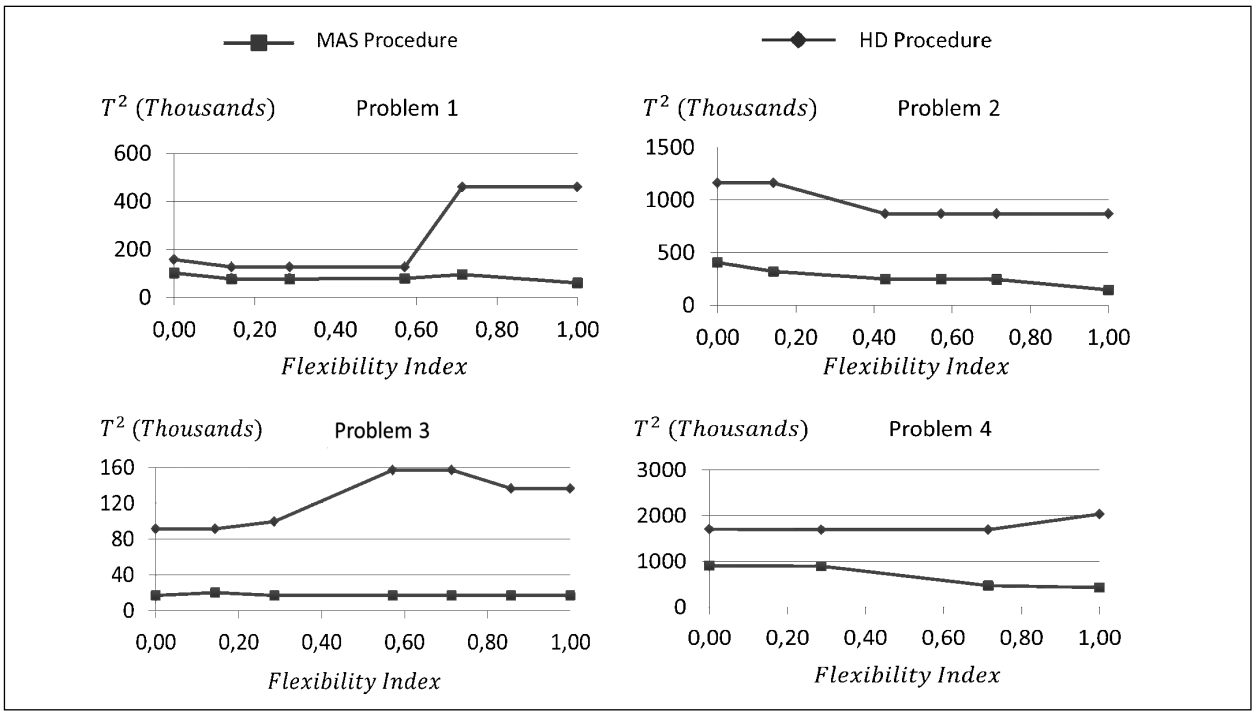

Figure 8
Objective function value for each case (problem and flexibility) and procedure

The preliminary tests show that our approach works suitable in the proposed scenarios. We obtain better results by means our system than by means the traditional dispatching methods. The flexibility management is better by means our approach as well.

8. References

AMARA, H., DÉPINCÉ, P., HASCOËT, J-Y. (2004). «A Human-Centered Architecture for Process Planning». *CIRP J. of Manufacturing Systems,* 33(4), pp. 363-372.

ARAÚZO, J.A. (2012). *Análisis y diseño Gaia de un sistema de fabricación basado en agentes*. Working paper. Departamento de Organización de Empresas y CIM, Universidad de Valladolid.

ARAÚZO, J.A., DE BENITO MARTÍN, J.J., DEL OLMO MARTÍNEZ, R. (2003) «El control de planta: una aproximación basada en agentes». *V Congreso de Ingeniería de Organización.*

ARAÚZO, J.A., DEL OLMO, R., LAVIÓS, J.J. (2013). «Subasta combinatoria para la programación dinámica en sistemas de fabricación distribuidos». *Dirección y Organización,* 51, pp. 55-64.

BLANC. P., DEMONGODIN, I., CASTAGNA, P, (2008). «A holonic approach for manufacturing execution system design: An industrial application». *Engineering Applications of Artificial Intelligence,* 21(3), pp. 315-330.

BONGAERST, L., (1998). *Integration of scheduling and control in holonic manufacturing systems.* Ph.D. Dissertation. Leuven: PMA Division of Leuven K.U.

BOTTI, V., GIRET, A., (2008). *Anemona: A Multi-agent Methodology for Holonic Manufacturing Systems.* Springer.

BRISSAUD, D., ZWOLINSKI, P. (2004). «End-of-Life-Based Negotiation Throughout the Design Process». *Annals of the CIRP,* 53(1), pp. 155-158.

BRUCCOLERI, M., LO NIGRO, G., PERRONE, G., RENNA, P., NOTO LA DIEGA, S. (2005), «Production Planning in Reconfigurable Enterprises and Reconfigurable Production Systems». *Annals of the CIRP,* 54(1), pp. 433-436.

BUSSMANN, S., JENNINGS, N.R., WOOLDRIDGE, M. (2004). *Multiagent Systems for Manufacturing Control: A design methodology.* Series on Agent Technology, Springer-Verlag: Berlin, Germany.

CHEN, S., TSENG, M. (2005). «Defining Specifications for Customer Products: A Multi-Attribute Negotiation Approach». *Annals of the CIRP,* 54(1), pp. 159-162.

GOU, L., LUH, P.B., KYOYA, Y. (1998). «Holonic manufacturing scheduling: architecture, cooperation mechanism, and implementation». *Computers in Industry,* 37(3), pp. 213-231.

HSIEH, F.S. (2008). «Holarchy formation and optimization in holonic manufacturing systems with contract net». *Automatica,* 44(4), pp. 959-970.

IGLESIAS, C.A., GARIJO, M., (2008). «The agent-oriented methodology MAS-CommonKADS». *Agent-Oriented Methodologies,* Idea Group, Hershey, Pa, USA, pp. 46-78.

LAVIOS, J.J., OLMO-MARTÍNEZ, R., ARAUZO, J.A., ORDAX, J.M. (2010). «Price Updating in Combinatorial Auctions for Coordination of Manufacturing Multiagent Systems». *Trends in Practical Applications of Agents and Multiagent Systems,* Springer: pp. 201-207.

LEITÃO, P. (2009). «Agent-based distributed manufacturing control: A state-of-the-art survey». *Engineering Applications of Artificial Intelligence,* 22(7), pp. 979-991.

LEITÃO, P., RESTIVO, F., (2008). «A holonic approach to dynamic manufacturing scheduling». *Robotics and Computer-Integrated Manufacturing,* 24(5), pp. 625–634.

MONOSTORI, L., VÁNCZA, J., KUMARA, S. R. (2006). «Agent-based systems for manufacturing». *CIRP Annals-Manufacturing Technology,* 55(2), pp. 697-720.

MORAITIS, P., SPANOUDAKIS, N.I. (2006). «The Gaia2Jade process for multi-agent systems development». *Applied Artificial Intelligence* 20(2-4), pp. 251-273.

OUNNAR, F., PUJO, P., (2012). «Pull control for job shop: holonic manufacturing system approach using multicriteria decision-making». *Journal of Intelligent Manufacturing,* 23(1), pp. 141-153.

PARUNAK, H.V.D., WHITE, J.F., LOZO, P.W., JUDD, R., IRISH, B.W., KINDRICK J. (1986). «An architecture for heuristic factory control». *Proc. of the American Control Conference,* Seattle, WA, USA, pp. 548-558.

PAVÓN, J., GÓMEZ-SANZ, J.J. (2003): *Agent Oriented Software Engineering with INGENIAS. 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003,* pp. 394-403.

POGGI, A., RIMASSA, G., TURCI, P., ODELL, J., MOURATIDIS, H., MANSON, G., (2004). «Modelling Deployment and Mobility Issues in Multi-agent Systems Using AUML». *Lecture Notes in Computer Science,* 2935, pp. 69-84.

ROULET-DUBONNET, O., YSTGAARD, P., (2011). «An application of the holonic manufacturing system to a flexible assembly cell». *Lecture Notes in Computer Science,* 6867, pp. 29-38.

SMITH, R.G. (1980). «The contract net protocol: high-level communication and control in distributed problem solving». *IEEE Transactions on Computer,* C29 (12), pp. 1104-1113.

WONGA, T.N., LEUNGA, C.W., MAKA, K.L., FUNGB, R.Y.K. (2006). «Dynamic shopfloor scheduling in multi-agent manufacturing systems». *Expert Systems with Applications,* 31(3), pp. 486-494.

WOOLDRIDGE, M., JENNINGS, N.R. (1995). «Intelligent Agents: Theory and Practice». *The Knowledge Engineering Review,* 10/2, pp. 115-152.

WOOLDRIDGE, M., JENNINGS, N.R., KINNY, D. (2000). «The Gaia Methodology for Agent-Oriented Analysis and Design». *Autonomous Agents and Multi-Agent Systems,* 3, pp. 285-312

ZAMBONELLI, F., JENNINGS, N.R., WOOLDRIDGE, M. (2003). «Developing Multiagent Systems: The Gaia Methodology». *ACM Transactions on Software Engineering and Methodology,* 12(3), pp. 317-337.