J Intell Manuf
DOI 10.1007/s10845-008-0119-z

Intelligent and collaborative Multi-Agent System to generate and schedule production orders

Omar López-Ortega · Virgilio López-Morales · Israel Villar-Medina

Abstract The authors present a Multi-Agent System for constructing and releasing *production orders*. In a manufacturing enterprise, the generation of production orders consists in a set of coordinated tasks among departments. This has been achieved traditionally as a module of the Production Activity Control (PAC) system. However, classic PAC modules lack collaborative techniques and intelligent behaviour. Moreover, in real-life situations experienced planners take over traditional PAC systems, since the range of possibilities to actually build production orders increases exponentially. To contribute to production planning, we present an intelligent and collaborative Multi-Agent System (MAS), having coordinated two forms to emulate intelligence. The learning capability is achieved by means of a Feed-forward Artificial Neural Network (FANN) with the back-propagation algorithm. The FANN is embedded within a *machine agent* whose objective is to obtain the appropriate machine in order to comply with requirements coming from the sales department. Also, an expert system is provided to a *tool agent*, which in turn is in charge of inferring the right tooling. The MAS also consists of a *coordinator* and a *spy*. The *coordinator agent* has the responsibility to control the flow of messages among the agents, whereas the *spy agent* is constantly reading the Enterprise Information System. Finally, a *scheduler agent* schedules the production orders. The resultant MAS improves the current form to plan production in a factory dedicated to produce *labels*.

Keywords Multi-Agent Systems · Production Planning · Neural Networks · Expert Systems · Supervised learning

O. López-Ortega (✉) · V. López-Morales · I. Villar-Medina
Centro de Investigación en Tecnologías de Información y Sistemas, Instituto de Ciencias Básicas e Ingeniería, Universidad Autónoma del Estado de Hidalgo, Carretera Pachuca—Tulancingo km. 4.5, Pachuca Hidalgo, México
e-mail: lopezo@uaeh.reduaeh.mx; olopez27@prodigy.net.mx

V. López-Morales
e-mail: virgilio@uaeh.edu.mx

I. Villar-Medina
e-mail: villar_israel@yahoo.com

Introduction

Collaboration in the manufacturing field occurs constantly. Fine examples are found not only in the supply chain (Julka et al. 2002), but also in the events and actions within a manufacturing company, which force enterprises to adjust manufacturing systems and comply with dynamic markets. Manufacturing systems are understood as a structured composition of software and hardware, aimed at organizing, planning and executing activities in order to produce the required value-added item. In this setting, software applications are grounded on diverse philosophies to facilitate collaboration.

The most common approach to support collaboration is the MRP and its extensions to the supply chain. MRP incorporates a module known as the Production Activity Control (PAC), which is paramount because it is a force that unites planning and manufacturing (Browne et al. 1992). This union is materialized by the construction of *production orders*, providing information regarding the machine and tools to be employed. However, as clients' orders tend to be far from homogeneous, the information system supporting the PAC module is mostly unfit to deal with the exponential number of combinations to generate the *right* production order. Therefore, the PAC module must be upgraded with software that reflects collaboration and incorporates flexible ways to generate a production order.

Arguably, the Multi-Agent System (MAS) has emerged as the major swift in the supporting systems for manufacturing.

Moreover, the addition of capabilities to emulate intelligence in agents is no longer an aspiration but a reality. For instance, it is possible to create agents with Expert Systems as presented in López-Ortega and López-Morales (2006). However, some other topics regarding cognitive abilities are left to be explored. Discussion on Artificial Intelligence (AI) has been focused on algorithms that mirror the supervised learning process. This is achieved primarily through Feed-forward Artificial Neural Networks (FANN), with the backpropagation algorithm. Even though this technique provides flexible ways of making inferences, it has not been widely exploited in Multi-Agent Systems to make Production Planning more flexible. Consequently, modelling and implementing a MAS that employs co-ordinately learning and decision making capabilities, becomes a great challenge to provide flexible mechanisms to cope with changing market conditions, and accordingly accommodate a production order. Thus, the main contribution we present in this paper is the design and implementation of a collaborative Multi-Agent System to improve Production Planning. To elaborate on the Multi-Agent System, this article is organized as it is explained next. A brief analysis of the related work is presented in the second section. The study case is divided into two sections. Third section contains the case study description and the transition to an MAS, while fourth section "The case study—part 2: the MAS illustrated" illustrates the actual execution of MAS by employing data from a real situation. Next, a discussion on the scheduling problem as a necessary step to achieve production control is presented in fifth section "Scheduling and closed-loop PAC". Finally, the conclusions of the project are drawn. We do not include theoretical insights on neither Expert Systems nor Artificial Neural Networks, but interested readers might consult Russel and Norvig (2002) and Jang et al. (1997).

Related work

We observe a solid trend to incorporate agent technology in order to resemble the collaborative nature of manufacturing systems. Marík and Lazanský (2007) coined the dramatic term *agentification of manufacturing systems* to summarize this trend. More explicitly, agent technology has been implemented to help communicate data among supporting applications, such as CAD or CAPP. An example is shown by Feng (2005), who created an MAS to improve collaboration between design and manufacturing departments. Kornienko et al. (2004) presents a MAS aimed at optimizing resource assignments by analyzing process plans. On a similar approach, Wang and Shen (2003) tackles the process planning problem, and extends the results to the scheduling problem (Wang et al. 2006), in an effort to optimize decisions before manufacturing execution actually takes place. In López-Ortega and López-Morales (2006), a semantic network is modelled as a necessary step to provide intelligence to a MAS dedicated to process planning. We further employed such findings to determine a rule-based system to create intelligent agents that decide which machines and tools adhere to a process plan (López-Ortega and López-Morales 2006). Similarly, in Feng et al. (2005a) a rule-based system is provided to agents to plan manufacturing processes. It can be argued that collaboration is facilitated by adding intelligence to agents.

An activity characterized by intense collaboration, which can be improved by using AI techniques and agent-based systems, is that of enterprise resource planning. Frey et al. (2003) implements an MAS to calculate the lot size, the operations to be performed, and the time-span between jobs. The information is fed to a simulator, which determines alternative courses of action before the production is initiated. Ostensibly, collaboration between different software applications seems to be more suitable with the Multi-Agent System approach.

Regarding collaboration between individual firms, several Multi-Agent Systems have been implemented. In Symeonidis et al. (2003), a Multi-Agent System includes data mining techniques to set up the profile of resources in a supply chain. The proposed MAS takes advantage of the information already stored in a commercial Enterprise Resource Planning (ERP) system. Julka et al. (2002) reports a MAS whose objective is to minimize stocks along a supply chain. In Lea et al. (2005), an MAS is used to integrate three standalone ERP modules. The Multi-Agent System consists of the following agents: *coordinator*, *task*, *data* and *GUI*. Both the *task agent* and the *coordinator agent* make partial decisions in order to simplify the resource planning process. Although the evidence of agent technology to boost enterprise collaboration is vaster than our brief review, we can draw some partial conclusions.

The incorporation of agent technology is based on the exploitation of existing applications and the ability to include intelligence. Rather than promoting the entire substitution of commercial applications, it is important to notice that agent technology might act in the *backstage*, helping to communicate results among different software systems. Also, with the advent of agent technology, the inclusion of algorithms to emulate intelligence is a way to provide flexible software systems. In this sense, rule-based systems are the main AI technique that has been added to software agents. Although the related research found in literature is highly valuable, this present paper pioneers on how to incorporate supervised learning in software agents to deal with complex decisions in a manufacturing environment. This paper also illustrates a realistic MAS, in which two different AI techniques are employed coordinately to accomplish production plans. The design and implementation of the Multi-Agent System we created serve to break boundaries within a manufacturing

enterprise, thus enhancing collaboration among the following departments: sales, production planning and manufacturing.

The case study—part 1: agentification of production planning

The environment is a factory dedicated to manufacture *labels*. Labels are used almost everywhere: On bottled products such as wine or sodas, candies, jeans, bar-coding, CD's, etc. They provide information about a given product. Consequently, their demand in the market place is high, yet prone to changes. Minor variations in size, colors, or raw material impact on the entire manufacturing system. Few companies have the capacity and expertise to produce them, and those that manufacture labels face tremendous production planning problems, because systems based on the MRP approach are simply not suitable to handle such complexity.

The typical flow of data in our case study is presented in Fig. 1. The labels are produced according to clients' requirements. An external application is used by the sales department to input clients' data, such as label design, color, dimensions, material and specific attributes. These data are stored in the Enterprise Information System (EIS). Once the client's order is processed by the sales department, the production planning department must elaborate a *production order*, on which the information to produce the label is comprised. The production order must be accurate and error-free so that the manufacturing department can perform adequately.

The possible combinations to produce a label are numerous. For example, the following raw material can be employed: glued-paper, non-glued paper, several types of cloth, plastic, cardboard, to name but only a few. Moreover, the raw material type determines how to acquire it, which can be in the form of continuous cylinders or in batches (also called master). Another complexity arises when a client sets the colors to be used (i.e. blue letters on white, glued-paper). Thus, the combination of tints is fixed according to the client requirements. The production planning manager must establish what operations (SU1, SU2 and SU3) are to be executed to comply with the client's required design of the label. On such information, the manager must determine the number of tools (up to three). Once these pieces of data are available, the production planning manager must provide the right machine (out of three options) on which the label will be produced. The machine depends on the following variables: raw material family, number of tints, resolution (dots per inch), type of finishing, and number of tools. This creates great difficulty to construct consistent production orders. Normally, the production planning manager relies on his/her experience; however, production orders normally contain imprecise data, as the manager shows inconsistent behavior about his/her decisions. The production manager either assigns the wrong number of tools for a given machine, or launches a production order for a machine that is not suitable to deal with a specific design requirement. This situation might occur due to fatigue, or the inability of a human-being to manage larger number of combinations. Therefore, the process of

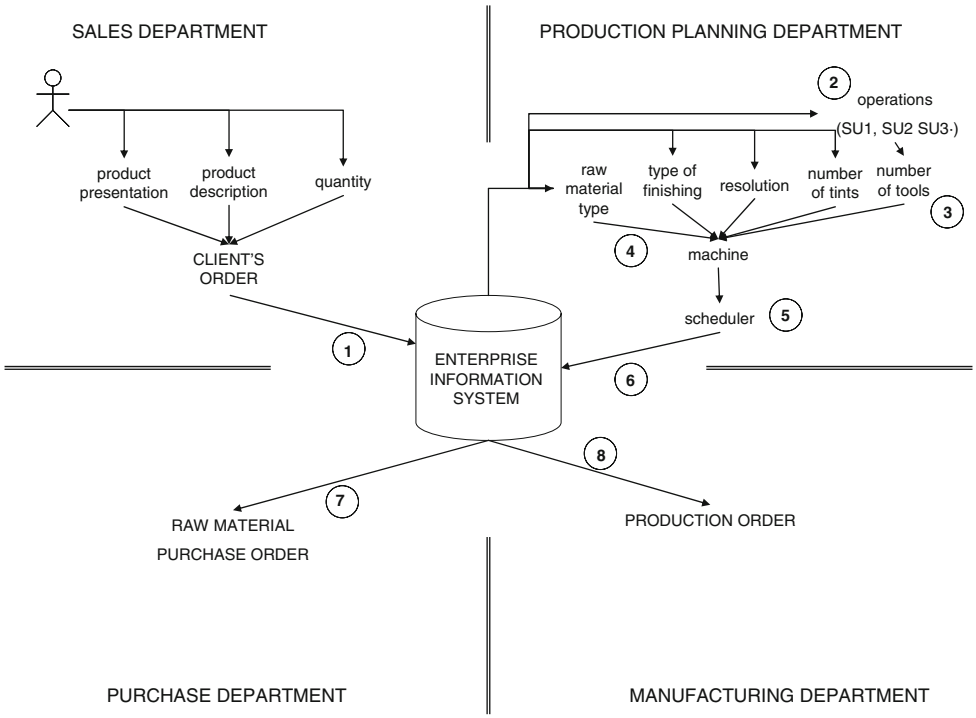

Fig. 1 The current situation

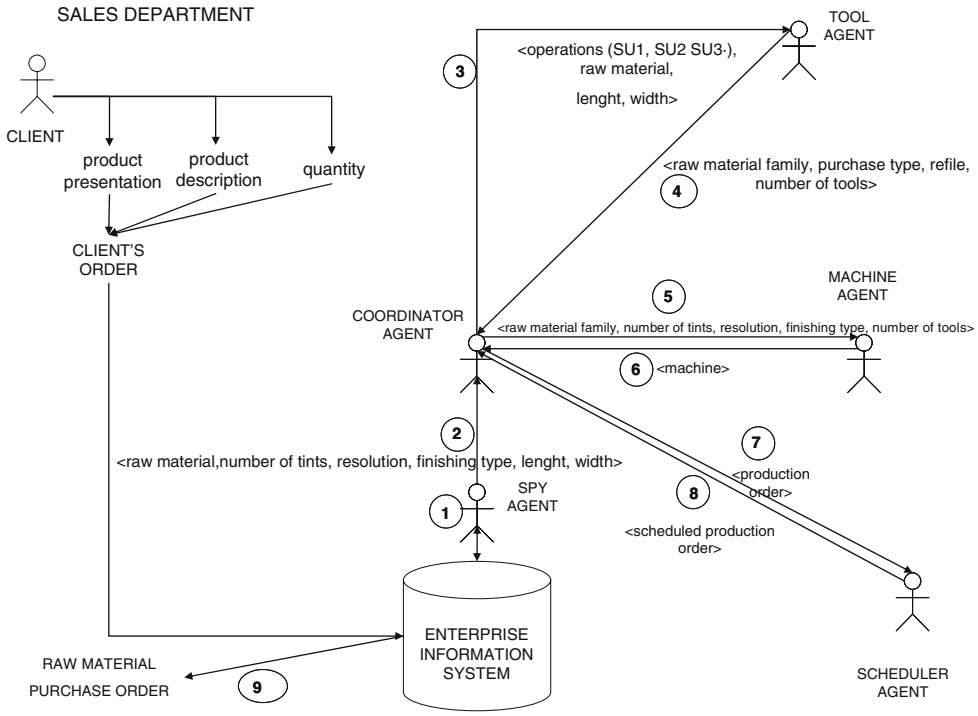

Fig. 2 General design of the Multi-Agent System

generating a production order must be automised, employing a flexible and robust approach.

Transition to a collaborative and intelligent system

The Multi-Agent System is a proposal to increase the quality of generating and releasing production orders. Our solution is to decompose the decision process carried by the human manager in a series of tasks performed by software agents (Fig. 2). These tasks are:

1. To read the EIS for newly created sales orders, and to acquire relevant variables.
2. To determine all the necessary tooling information.
3. To determine the correct machine.
4. To establish a sequence for launching production orders to the manufacturing department.
5. To maintain data consistency and robustness along the process.

The primary goal of the MAS is to construct a production order. In the design we propose a *coordinator agent*, which is responsible for maintaining data consistency along the process by controlling the flow of messages among the agents. A *spy agent* must read the EIS in order to acquire client's orders, and send the information to the *coordinator*. Two individual agents are in charge of making decisions regarding tools and machines. A *machine agent* obtains the appropriate machine, whereas the *tool agent* is responsible for providing the right tooling. As soon as the machine, tools and other data are established, a priority is assigned to the production order. To do so, we include a *scheduler agent*. When the production order gets the priority, it is then launched to the manufacturing department. Given the particular features of the planning problem at hand, we faced the dilemma of selecting algorithms so that each intelligent agent obtains valid conclusions.

For an agent to settle the number of tools to be used during production of the label, it was possible to model the manager's knowledge in the form of explicit rules. Such rule-base is also employed to delineate the raw material family and how to purchase it. The rule-base contains knowledge to decide on the right tooling. The *tool agent* is, therefore, an expert in charge of exploiting the rule-based system and come up with the mentioned pieces of information.

On the other hand, determination of the machine where the label is to be produced is not straight forward. As it has been stated before, the number of possibilities grows exponentially. In this situation, it is obviously impractical to create a rule-based system with thousands of rules, at the risk of missing valid combinations, if the rule-base is incomplete. Nonetheless, Artificial Neural Networks are known to be good *classifiers*, so that an input pattern, which is likely to be unknown, is mapped to a value that resides within one of various output classes. For these reasons, the *machine agent* possesses a FANN, which gives it the skill to find out what machine fits best according to the actual combination of input values. Yet, it was necessary to set up a training matrix and employ a method of supervised learning. On the following, we elaborate on both, the communication protocol

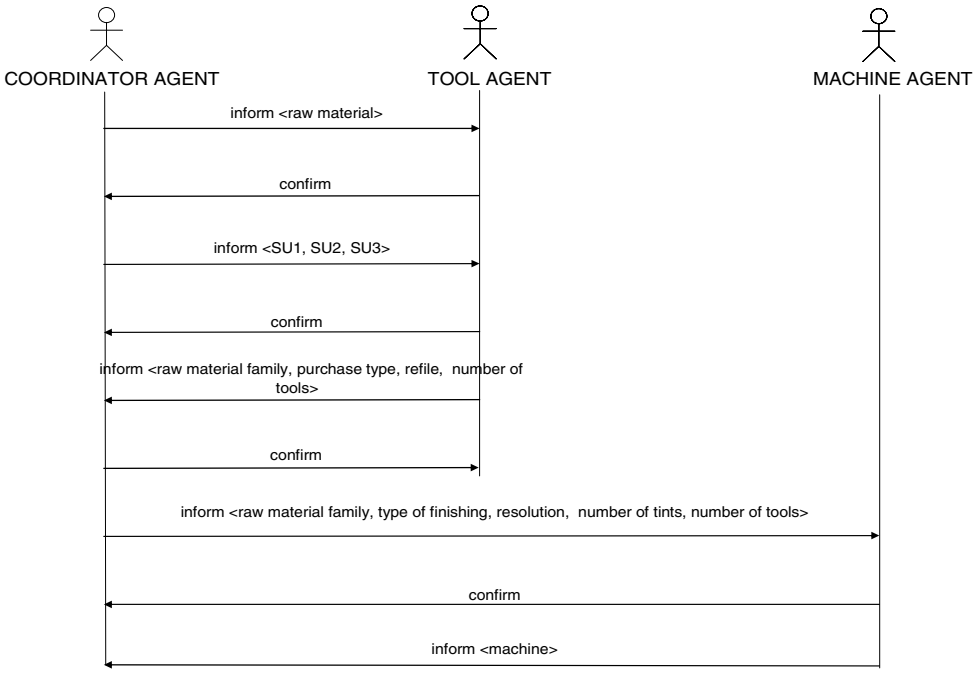

Fig. 3 The communication protocol

and the intelligent capabilities implemented to provide flexible mechanisms of collaboration. Specific data for this case study is provided in the fourth section "The case study—part 2: the MAS illustrated".

The communication protocol

The MAS is globally coordinated by means of a communication protocol. In Fig. 3 we present a simplified version of the proposed model. The left side shows the protocol between the *coordinator* and the *tool agents*. On the right side, a similar model is used between the *coordinator* and the *machine agents*. According to the AUML notation (Bauer and Odell 2005), the solid arrows at the end of each message represent synchronous exchanges. For the purpose of simplicity, we show several contents in a single message, even though this is not standard notation. Figure 3 does not illustrate the update of the Enterprise Information System, action that is performed by the *coordinator*. The communication between the *coordinator* and the *machine agent* is not initiated until the *tool agent* informs what results it has achieved. The entire communication process ends as soon as the *coordinator* updates the value of the machine in the EIS. When this occurs, the production order is on hold to be scheduled. The actual message content that is uttered by both, the *tool agent* and the *machine agent*, comes from the emulation of intelligent behaviour.

The rule-base embedded in tool agent

The *tool agent* has to settle the number of tools, as soon as it receives data from the *coordinator*. Although this is the main task for the *tool agent*, we obtain intermediate data while the inference process is taking place. The *coordinator* sends the length and width of the label, as well as the material on which the label is to be produced. The *tool agent* loads the received values into the expert system, and initiates the forward chaining of rules. The rule-base is programmed in a shell called *rule*, as this shell proved effective in our previous work on doting expert systems to agents (López-Ortega and López-Morales 2006). Specifically, the expert system embedded in the *tool agent* obtains the number of tools, based on the values of SU1, SU2 and SU3. This is exemplified by the following excerpt.

```
ruleT1: IF SU1 != ""none"" AND SU2 !=
   ""none"" AND SU3 != ""none""
THEN number_tools = 3
ruleT2: IF SU1 = ""none"" AND SU2 =
   ""none"" AND SU3 = "none"
THEN number_tools = 1
ruleT3: IF SU1 != "none" AND SU2 =
   "none" AND SU3· = "none"
THEN number_tools = 1
ruleT4: IF SU1 = "none" AND SU2 ! =
   "none" AND SU3 = "none"
THEN number_tools = 1
ruleT5: IF SU1 = "none" AND SU2 =
   "none" AND SU3 != "none"
THEN number_tools = 1
ruleT6: IF SU1 != "none" AND SU2 ! =
   "none" AND SU3 = "none"
THEN number_tools = 2
```

```
ruleT7: IF SU1 != "none" AND SU2 =
   "none" AND SU3 != "none"
THEN number_tools = 2
ruleT8: IF SU1 = "none" AND SU2 ! =
   "none" AND SU3 != "none"
THEN number_tools = 2
```

Based on the raw material, the *tool agent* determines the raw material family. Some rules read:

```
ruleRMF01: IF RAW_MATERIAL=
   "kimno" THEN RAW_MATERIAL_FAMILY =
   "kimdura"
ruleRMF02: IF RAW_MATERIAL=
   "kimsi" THEN RAW_MATERIAL_FAMILY =
   "kimdura"
ruleRMF03: IF RAW_MATERIAL=
   "kimte" THEN RAW_MATERIAL_FAMILY=
   "kimdura"
ruleRMF04: IF RAW_MATERIAL=
   "valer" THEN RAW_MATERIAL_FAMILY =
   "fabric"
ruleRMF05: IF RAW_MATERIAL=
   "poliester" THEN RAW_MATERIAL_FAMILY =
   "fabric"
ruleRMF06: IF RAW_MATERIAL=
   "nylon" THEN RAW_MATERIAL_FAMILY =
   "fabric"
ruleRMF07: IF RAW_MATERIAL=
   "auttr" THEN RAW_MATERIAL_FAMILY =
   "paper"
ruleRMF08: IF RAW_MATERIAL=
   "orono" THEN RAW_MATERIAL_FAMILY=
   "paper"
ruleRMF09: IF RAW_MATERIAL=
   "oroag" THEN RAW_MATERIAL_FAMILY =
   "paper"
ruleRMF10: IF RAW_MATERIAL=
   "cartd" THEN RAW_MATERIAL_FAMILY =
   "cardboard"
ruleRMF11: IF RAW_MATERIAL=
   "carts" THEN RAW_MATERIAL_FAMILY =
   "cardboard"
```

Then, the type of purchase is set according to the information gathered by the *tool agent*. Should raw material be purchased on continuous cylinders or on batches, it all depends on the raw material family. An excerpt of the rules to decide what type of purchase follows here:

```
ruleTOP1: IF RAW_MATERIAL_FAMILY =
   "fabric" AND RAW_MATERIAL != "nylon"
   THEN TYPE_OF_PURCHSE = "master"
ruleTOP2: IF RAW_MATERIAL_FAMILY =
   "fabric" AND RAW_MATERIAL = "nylon"
   THEN TYPE-OF_PURCHASE = "continuous
   cylinder"
ruleTOP3: IF RAW_MATERIAL_FAMILY =
   "cardboard" AND RAW_MATERIAL ! =
   "cartd"
   THEN TYPE_OF_PURCHASE = "master"
ruleTOP4: IF RAW_MATERIAL_FAMILY =
   "cardboard" AND RAW_MATERIAL =
   "cartd"
   THEN TYPE_OF_PURCHASE = "continuous
   cylinder"
ruleTOP5: IF RAW_MATERIAL_FAMILY =
   "kimdura"
   THEN TYPE_OF_PURCHASE = "master"
ruleTOP6: IF RAW_MATERIAL_FAMILY =
   "various"
   THEN TYPE_OF_PURCHASE = "master"
ruleTOP7: IF RAW_MATERIAL_FAMILY =
   "paper" AND RAW_MATERIAL != "dual"
   THEN TYPE_OF_PURCHASE = "master"
ruleTOP8: IF RAW_MATERIAL_FAMILY =
   "paper" AND RAW_MATERIAL = "dual"
   THEN TYPE_OF_PURCHASE =
   "continuous_cylinder"
```

The *tool agent* informs the *coordinator* about its conclusions. The *coordinator*, having received the information, initiates communication with the *machine agent*. The *machine agent* employs a FANN to obtain authoritatively the machine where the label should be produced.

Implementation of the FANN

We provide in this section details of the Feed-forward network to determine the right machine. This depends on the combination of six different data sets, as follows:

raw_material_family X tints X resolution X finish_type X number_tools− > machine

We capitalize on the power of Artificial Neural Networks in order to solve this combinatorial problem. The architecture chosen for the Artificial Neural Network is called

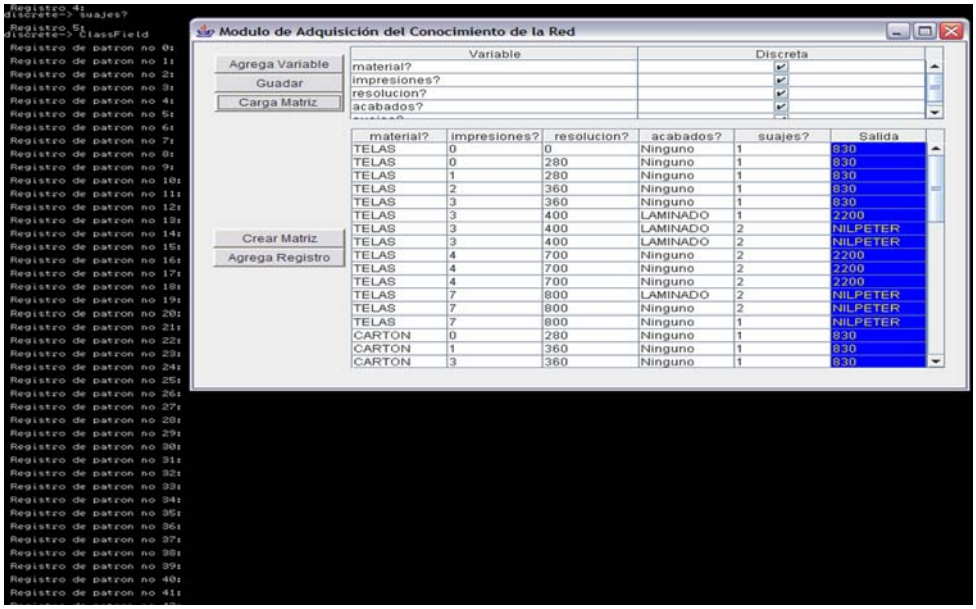

Fig. 4 Module to define the training matrix

feedforward, since the input values are transmitted along the layers of the net, until the output layer is reached. This type of net is trained by using the backpropagation of the error signal. This method is based on the gradient descent technique, so that the error value diminishes as the gradient of a given function i.e the logistic function, is calculated (Jang et al. 1997). In each training pass, the weights are updated. The training phase is complete once the error value is less than a given target value. When this occurs, the resultant weights are stored. They are later employed by the net to fix an output value, should an unknown input pattern arrive.

Although there are countless simulators of neural networks, none of them are suited to be used by a Multi-Agent System. Hence, it was necessary to incorporate supervised learning in the *machine agent* by means of Java programming. To comply rightly with the requirements to implement backpropagation, a number of technological innovations were carried out, adapting the work of Bigus and Bigus (2002) on Java coding of backpropagation. Thus, we created software to facilitate the implementation of the FANN:

- A module to define the training matrix, with different types of input values (Fig. 4).
- A module to track the error value.
- A module to acquire and store weights, once the training phase is over.
- A module to incorporate the resultant FANN into the *machine agent*.

The formal representation of the software modules to train and execute the FANN is presented in Fig. 5. The class diagram shows that the *machine agent* is implemented on the JADE platform. This agent incorporates a neural network by

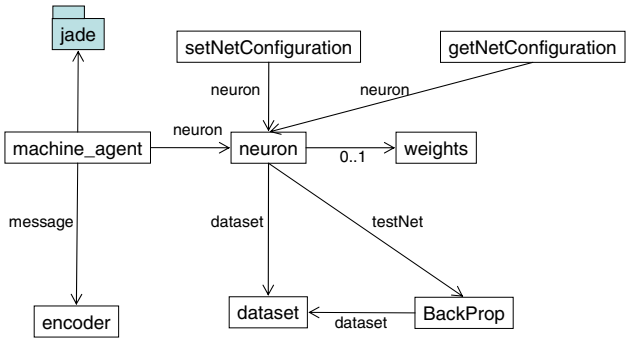

Fig. 5 Class diagram for the tool agent

instantiating the neuron class. To use effectively the backpropagation algorithm, symbols (actual values for the input and output of the net) must be encoded-decoded into a number in the interval [0,1]. This process is done via the `encoder` class. The `neuron` class in turn uses and determines the configuration of the net. The `weights` class is used to create and read the files *"pesos.dat"* where the resultant weight vector resides. The `BackProp` class possesses the necessary coding to compute the backpropagation algorithm.

The next step consisted in training the FANN. This was done by constructing a training matrix, which is a subset sufficiently representative of the entire range of possibilities to set a machine. Nonetheless, it was not a trivial task to obtain the most appropriate training matrix (with some of the training combinations the FANN stopped at a saddle point, or a local minimum). It was not after several tries, supervised by the production manager, that we could come up with a fiable training set. The training matrix is presented in Fig. 6.

Fig. 6 Training Matrix for the ANN

INPUTS					OUTPUT
RAW MATERIAL FAMILY	**NUMBER OF TINTS**	**RES (DPI)**	**FINISH TYPE**	**NUMBER OF TOOLS**	**MACHINE**
FABRIC	0	0	NONE	1	830
FABRIC	0	280	NONE	1	830
FABRIC	***1***	***280***	***NONE***	***1***	***830***
FABRIC	***2***	***360***	***NONE***	***1***	***830***
FABRIC	3	360	NONE	1	830
FABRIC	3	400	LAMINATED	1	2200
FABRIC	3	400	LAMINATED	2	NILPETER
FABRIC	3	400	LAMINATED	2	NILPETER
FABRIC	4	700	NONE	2	2200
FABRIC	4	700	NONE	2	2200
FABRIC	4	700	NONE	2	2200
FABRIC	7	800	LAMINATED	2	NILPETER
FABRIC	7	800	NONE	2	NILPETER
FABRIC	7	800	NONE	1	NILPETER
CARDBOARD	0	280	NONE	1	830
CARDBOARD	1	360	NONE	1	830
CARDBOARD	3	360	NONE	1	830
CARDBOARD	4	500	LAMINATED	1	2200
CARDBOARD	4	360	NONE	1	2200
CARDBOARD	5	600	LAMINATED	1	2200
CARDBOARD	6	700	NONE	2	NILPETER
CARDBOARD	5	800	NONE	3	NILPETER
CARDBOARD	6	700	LAMINATED	2	NILPETER
CARDBOARD	5	800	LAMINATED	2	NILPETER
KIMDURA	0	360	NONE	1	2200
KIMDURA	1	400	LAMINATED	1	2200
KIMDURA	1	400	LAMINATED	2	2200
KIMDURA	1	400	LAMINATED	1	2200
KIMDURA	5	360	NONE	1	NILPETER
KIMDURA	6	360	NONE	1	NILPETER
BOPP	0	280	NONE	1	830
BOPP	1	360	LAMINATED	1	830
BOPP	3	400	NONE	1	2200
BOPP	3	400	LAMINATED	1	2200
BOPP	5	500	NONE	1	2200
BOPP	6	700	NONE	1	2200
BOPP	7	800	NONE	2	NILPETER
PAPER	0	280	NONE	1	830
PAPER	0	280	NONE	1	830
PAPER	1	360	NONE	1	830
PAPER	2	360	NONE	1	830
PAPER	3	500	NONE	1	2200
PAPER	3	600	SULFATED	1	2200
PAPER	3	700	LAMINATED	2	2200
PAPER	3	600	LAMINATED	2	2200
PAPER	6	360	LAMINATED	1	NILPETER
PAPER	5	700	NONE	2	NILPETER
PAPER	6	800	NONE	3	NILPETER
OTHER	0	0	NONE	1	2200
OTHER	4	500	NONE	3	NILPETER

The actual configuration of the FANN is closely bound to the training matrix. Specifically, one processing unit is created for each value contained in the training set. For example, as Input 1 of the training matrix has six different values, six processing units are built. These processing units belong to the input layer of the net. Since Input 2 has six different values, then six processing units are also fixed in the input layer. The training module we set up is helpful to fix the number of processing units.

Once the processing units for the input layer are generated, then the intermediate layer is set. The module assigns the same number of processing units in the input layer to the intermediate layer. The Java code fixes only one intermediate layer. A FANN with a single intermediate layer is powerful enough to deal with non-linear problems. The output layer is then constructed.

The number of processing units in the output layer depends on the number of values that are provided as valid outcomes in the training set. In our training matrix, the output layer is given three different values, and three processing units are constructed. The number of weights to link together the processing units is the product of the resultant processing units. Specifically, our net possesses 25*25*3 weights which must be adjusted during the training phase and used during execution phase.

Thus, the FANN to be trained and further exploited consists of 3 layers, 25 processing units in both, the input and intermediate layers, and three processing units in the out-

put layer. The error value during the training phase was set at 0.004. Once the FANN descended below this value, the training phase was completed, and the weights values were stored in a binary file called "*pesos.dat*". This file is then used when the FANN is set in execution mode. The execution of the FANN has given positive results. We performed up to a hundred runs once the net was trained, and in 97% of the cases the *machine agent* provided a reliable result.

It is utterly complex to implement intelligence and to coordinate communication among agents. Extensive Java and JADE code was either adapted or newly created to provide the necessary intelligent capabilities. Also, the *behaviours* of the agents were set up properly in order to avoid interferences while the agents are exchanging information. Interested readers may consult Bellifemine et al. (2007) for insights on how to program agents on the JADE platform. In the following section we present the results obtained for the entire Multi-Agent-System.

The case study—part 2: the MAS illustrated

In this section, we illustrate the series of tasks and partial decisions achieved by the software agents. We deliberately use the command line to illustrate the entire process of message exchange between the agents. The process starts as soon as the sales department enters a client's requirement. The *spy agent* realizes that a new order has just entered the EIS, and it sends a series of messages to the *coordinator*, informing the values of the following variables:

Sales order: 594
Width: 112
Length: 1000
Raw material: NYLON
SU1: X007
SU2: " "
SU3: " "
number of tints: 0
resolution: 360
type of finishing: none

Figure 7a shows how the *spy agent* informs the *coordinator agent* about the previous data, organized in the sales order 594. When the *coordinator* acknowledges reception of messages from the *spy*, then it informs the *tool agent* regarding width, length, raw material, SU1, SU2 and SU3.

The *tool agent* assesses that it has enough data to initiate its reasoning process ("`Agente herramienta con suficientes datos`"), then it runs the expert system and obtains a conclusion (Fig. 7a, bottom). The following rules are triggered according to the input data of the current example:

ruleRMF06: IF RAW_MATERIAL = "nylon"
THEN RAW_MATERIAL_FAMILY = "fabric"
ruleTOP2: IF RAW_MATERIAL_FAMILY=
"fabric" AND RAW_MATERIAL = "nylon"
THEN TYPE_OF_PURCHSE = "continuous
cylinder"
IF SU1 != "none" AND SU2 = "none"
AND SU3 = "none" THEN number_tools = 1

Therefore, the *tool agent* reaches the next conclusion:

RAW MATERIAL FAMILY: Fabric
TYPE OF PURCHASE: Continuous cylinder
(rollo)
NUMBER OF TOOLS: 1

According to the communication protocol presented, the *tool agent* informs the *coordinator agent* about its conclusions. On the reception of messages from the *tool agent*, the *coordinator* sends the following data to the *machine agent* (Fig. 7b—"`la cadena entrante es: TELAS 0 360 Ninguno 1`"):

RAW MATERIAL FAMILY: Fabric
NUMBER OF TINTS: 0
RESOLUTION : 360
TYPE OF FINISHING: none
NUMBER OF TOOLS: 1

Based on this type of information, the FANN embedded within the *machine agent*, which is set into execution mode, obtains a valid machine, and sends a message to the *coordinator* ("`Agente coordinador recibi mensaje MR = 830`"). In this example, the obtained machine is known as "830" (again, this is factory-specific terminology). Readers are encouraged to compare the previous set of data with the two highlighted records on the training matrix in Fig. 6. On these two records the FANN was trained to establish the machine value as "830". In the example, the *machine agent* approximates its decision to the closest value on which it was trained, thus stating that machine "830" is the one to be used. For this production order, the assigned priority is 205. Figure 7c summarizes the results obtained by the MAS. The entire process is repeated every time the *spy agent* reads a new sales order in the EIA. As a result of such process, the construction and release of a valid production order is fully automised by the intelligent and collaborative system that we developed.

Scheduling and closed-loop PAC

The purpose of the MAS is clear: constructing a production order that integrates information to be employed during manufacturing execution, i.e. machine type, number of tools, tints. Data from sales department is passed top–bottom, and then it is transformed into a production order that finally reaches the manufacturing department. We can claim that

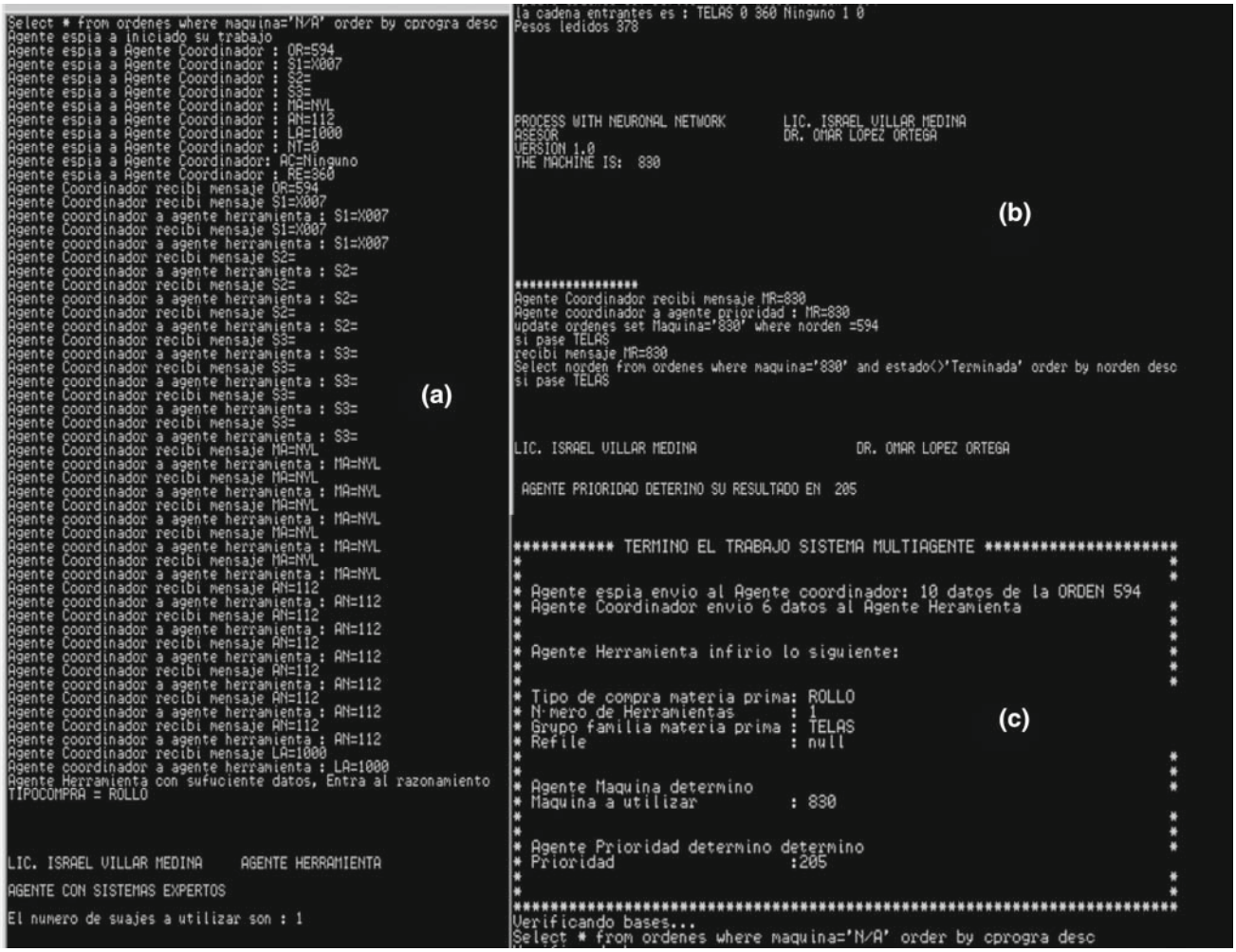

Fig. 7 Results obtained by the Multi-Agent System

production planning is improved by a collaborative and intelligent system. However, should manufacturing execution be more realistic, it is necessary to develop a solid scheduling policy. Our *scheduler agent* employs a FIFO policy, even though it is the simplest ordering/queueing mechanism. We are assessing results given in Wang et al. (2006) as future upgrades of the *scheduler agent*. We can asses that integrating planning and scheduling is not a trivial task, yet it opens up opportunities to actually close the loop between production planning and manufacturing execution. The current configuration of the MAS that we developed enhances production *planning*, yet production *control* is not fully covered. Thus, a dynamic control policy (Paternina-Arboleda and Das 2005) might be obtained when relevant variables, such as WIP inventory, backorder penalty cost, setup time and cost, processing and transportation cost and time, or machine disruption time, are sent back to planning for achieving production control. The loop is closed when events within the manufacturing department, such as machine failure or a delay in the setup time, are communicated in real-time to the planning department. Optimization can be achieved by using a scheduling policy that benefits from realistic data. Therefore, communication must be bidirectional, from production planning to manufacturing execution and vice versa. We suggest to design a hierarchy of agents, or a holarchy (Walter et al. 2006), to accomplish agent-based PAC. Communication, intelligence, and the capability to integrate data from external applications are key features that must be exploited thoroughly. Achieving such necessary feedback is an opportunity to continue research in the field of agent-based manufacturing.

Conclusions

Based on theoretical and practical results, we think that Multi-Agent Systems represent a major swift in supporting systems for manufacturing. We contribute to agent-based production planning by presenting a collaborative and intelligent MAS.

Particularly, the cognitive function of learning is possible by means of a feed-forward Artificial Neural Network with the back-propagation algorithm, embedded within a *machine agent*. We also employ another method to emulate intelligence by using an Expert System, which has been embedded in a *tool agent*. Collaboration is controlled by a *coordinator,* whereas a *spy* is constantly reading the Enterprise Information System in order to acquire client's orders, and send the information to the *coordinator*. Marík (*op cit*) acknowledges the inclusion of agent technology on the production planning level. The *agentification* process of the production planning department, he claims, provides an elegant mechanism for system integration, and supports the migration from centralized planning towards distributed and flexible architectures. The MAS we developed contributes to achieve this. To the best of the authors' knowledge, this is the first report on using rule-based systems and supervised learning coordinately, as mechanisms to improve Production Planning. Moreover, decisions obtained by individual agents reflect the various steps that must be performed in different departments in a real-life situation. We also point that existing applications are not required to be entirely substituted. Some of them still provide valid data, so agent technology is appropriate to process such data intelligently, and communicate with any software residing elsewhere (i.e. in another enterprise). In this sense, agent technology is mature enough to be fully exploited in real-world applications, as our MAS illustrates.

References

Bauer, B., & Odell, J. (2005). UML 2.0 and agents: how to build agent-based systems with the new UML standard. *Engineering Applications of Artificial Intelligence, 18*, 141–157.

Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. USA: Wiley and Sons.

Bigus, J., & Bigus, J. (2002). *Constructing Intelligent Agents using Java* (2nd ed.). NY: Wiley and Sons.

Browne, J., Harhen, J., & Shivnan, J. (1992). *Production Management Systems. A CIM perspective*. UK: Addison-Wesley.

Feng, S. C. (2005). Preliminary design and manufacturing planning integration using web-based intelligent agents. *Journal of Intelligent Manufacturing, 16*, 423–437.

Feng, S. C., Stouffer, K. A., & Jurrens, K. K. (2005a). Manufacturing planning and predictive process model integration using software agents. *Advanced Engineering Informatics, 19*, 135–142.

Frey, D., Nimis, J., Worn, H., & Lockemann, P. (2003). Benchmarking and robust multi-agent production planning and control. *Engineering applications of Artificial Intelligence, 16*, 307–320.

Jang, J.-S. R., Sun, C.-T., & Mizutani, E. (1997). *Neuro-fuzzy and soft computing. A computational approach to learning and machine intelligence*. USA: Prentice-Hall.

Julka, N., Srinivasan, R., & Karimi, I. (2002). Agent-base supply chain management-1: framework. *Computers and Chemical Engineering, 26*, 1755–1769.

Kornienko, S., Kornienko, O., & Priese, J. (2004). Application of multi-agent planning to the assignment problem. *Computers in Industry, 54*, 273–290.

Lea, B. R., Gupta, M. C., & Yu, W. B. (2005). A prototype multi-agent ERP system: an integrated architecture and a conceptual framework. *Technovation, 25*, 433–441.

López-Morales, V., & López-Ortega, O. (2005). A distributed semantic network model for a collaborative intelligent system. *Journal of Intelligent Manufacturing, 16*, 515–525.

López-Ortega, O., & López-Morales, V. (2006). Cognitive communication in a multi-agent system for distributed process planning. *International Journal of Computer Applications in Technology, 26*, 99–107.

Marík, V., & Lazanský, J. (2007). Industrial applications of agent technologies. *Control Engineering Practice*, *15*(11), 1364–1380.

Paternina-Arboleda, C. D., & Das, T. K. (2005). A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problems. *Simulation Modelling Practice and Theory, 13*, 389–406.

Russel, S. J., & Norvig, P. (2002). *Artificial intelligence: A modern approach* (2nd ed.). USA: Prentice-Hall.

Symeonidis, A. L., Kehagias, D., & Mitkas, P. (2003). Intelligent policy recommendations on enterprise resource planning by the use of agent technology and data mining techniques. *Expert Systems with Applications, 25*, 589–602.

Walter, S. S., Brennan, R. W., & Norrie, D. H. (2006). Experience and reflection on the development of a holonic job-shop scheduling system. *International Journal of Computer Applications in Technology, 26*, 15–27.

Wang, L., & Shen, W. (2003). DPP: An agent-based approach for distributed process planning. *Journal of Intelligent Manufacturing, 14*, 429–439.

Wang, L., Shen, W., & Hao, Q. (2006). An overview of distributed process planning and its integration with scheduling. *International Journal of Computer Applications in Technology, 26*, 3–14.