Agent Based Prototype for Interoperation of Production Planning and Control and Manufacturing Automation

Rui M. Lima
School of Engineering of University of Minho
Department of Production and Systems
Campus de Azurém, Guimarães, Portugal
rml@dps.uminho.pt

Rui M. Sousa
School of Engineering of University of Minho
Department of Production and Systems
Campus de Azurém, Guimarães, Portugal
rms@dps.uminho.pt

Abstract

This work describes a model for distributed dynamic Production Planning and Control (PPC) agent based system, which includes interoperation with manufacturing automation. It is presented a demonstration prototype involving distributed software agents and industrial equipment integration, which implements part of the developed model functionalities. Clients can send orders, and resources may apply for those orders fulfilment. Resources with orders allocated to, start automatically the required manufacturing operations. The prototype was implemented integrating several tools, including LabVIEW and LEGO Mindstorms components. This is useful to validate the integration, proposed by the dynamic PPC model, between production planning processes and manufacturing execution operations.

1. Introduction

Production Planning and Control (PPC) systems include planning functions like determining the quantity and timing of materials and capacity requirements in order to satisfy production system demand. Furthermore, the system could "tell" which production resources should be used to provide those capacity requirements. In this case a function related with detailed dispatching or scheduling would be responsible for those decisions. On scheduling functions, decisions are taken in advance considering some state of utilization of the production system and some criteria combination. In the case of dispatching functions, decisions are taken at the moment that the action should be executed, also considering a view of the state of the system and a combination of criteria. After those decisions, the system should be able to execute and control the shop orders.

Production shop orders are executed by production resources, i.e. individual or group of operators and machines. If the system is composed, even partially, by computer controlled equipment, some operations from the shop orders could be executed in a more automatic way interoperating PPC with Computer Aided Manufacturing (CAM) control equipment.

The dynamic interoperation between Production Planning and Control systems and Automation Equipment is being researched by several authors using models based in the software agent paradigm. Yanli *et al* [1] describe a manufacturing execution system based on the scheduling processes and execution control of tasks related with part orders received from the ERP/MRP system. This multi-agent system is based on a Director Facilitator agent in order to find adequate service agents and a Broker Agent to mediate the resource selection process. Wang *et al* [2] use a three layer structure based on Client agents, Facilitator agents and Resource agents to propose Virtual Computer Integrated Manufacturing (VCIM) architecture for small to medium enterprises. They present a prototype able to build schedule solutions for some case studies. In [3] it is presented a solution for product design and process planning, and also for scheduling and controlling the production execution. This system is based on six fundamental operational agents and one centralized control managing agent. Dynamic configuration is a problem considered by Bruccoleri *et al* [4] in their reconfigurable machine system (RMS) proposal. This model has four negotiation protocols to handle dynamic resource allocation and different types of exceptions, namely: machine breakdowns, machine deterioration and rush orders.

In this work, the dynamic selection, allocation and operation of computer controlled production resources, for faster and effective response of the production system, are the fundamental objectives. This is addressed by a prototype development based on the interoperation between a dynamic PPC system and manufacturing equipment.

2. Dynamic PPC model

Production Planning and Control (PPC) systems must be able to transform demand requirements into shop orders, and ultimately to control their execution. Usually this is achieved, using Hierarchical Production

1-4244-0826-1/07/$20.00 © 2007 IEEE

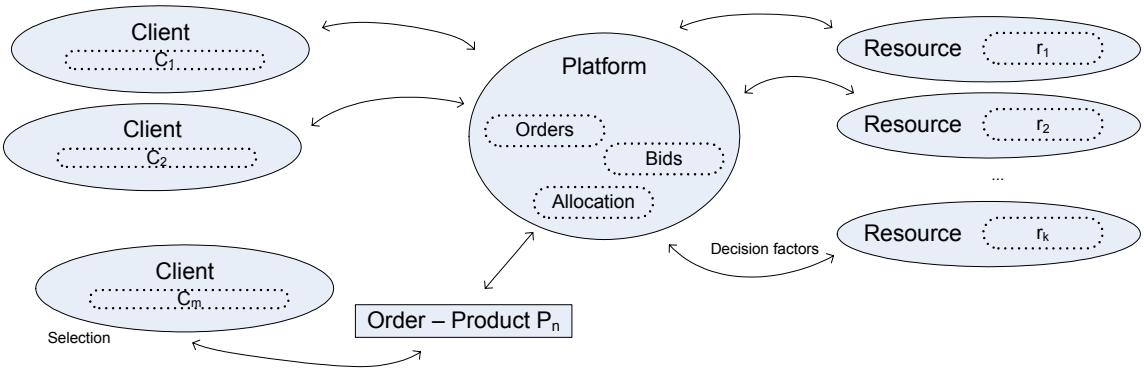

Figure 1: PPC Model Illustration.

Planning and Control, based on Manufacturing Requirements Planning (MRP) and Capacity Planning functions. This approach considers a predefined production organization structure in order to compute material requirements for defined demand during a planning horizon for each period of time. Material requirements are then used to compute capacity requirements that must me compared with planned shop orders to decide about the possibility of executing them. A dynamic PPC model could be more appropriate to deal with a fast changing environment based on shortened production flow time requirements imposed by increased customization of product orders.

The referred changing environment requires a dynamic PPC model able to reduce production flow time and deal with customization of orders. An adequate production system should be production to order type. Lima *et al* [5] proposed a distributed dynamically adaptable PPC system based on the production planning, control and execution of a production order of one product for each demand order. This model is based on the dynamic selection and allocation of resources to satisfy each order in the minimum possible time. In this model, illustrated by Figure 1, Clients and Resources communicate through a Platform that acts, mainly as a Blackboard, publishing each order of one product and available capable resources bid offers. Clients select resources for their orders and allocation can be done. This is a protocol of negotiation based on the Contract Net [6].

In this model an autonomous resource can completely execute a transformation in the production process in order to deliver an item of the product structure. Two extreme systems could be considered, one based on a global view of availability of resources and another based on local views of the resources. In the first one, resources have some restrictions on their autonomy because clients receive capability and free capacity of every resource, using that information to take a global decision of selection of resources for each item of the product structure. In the second case, the one explored in this work, interested resources present offers for each item order, and could act as a client creating orders for each material requirement for that item. In this way, the production system can be modelled to the desired extent, modelling the product structure.

The five basic stages for the resolution of problems of resource allocation for distributed manufacturing systems, according to Tharumarajah [7] are: (1) "decompose order(s) into operations"; (2) "assign operation(s)"; (3) "select machine"; (4) "allocate operation"; (5) "coordinate allocation & build schedule". In stage 1, orders are decomposed in operations, which are related with machines in stage 2. In stage 3 machines are selected for the execution of each operation. In stage 4 operations are allocated to machines. The coordination of the allocation of operations to machines results from stage 5, which includes the construction of the production schedule. In reality, problems solved in these stages have to be solved whenever it is necessary to allocate operations to resources and to build production schedules.

According to stage 1, in this model, the order is related with one product and the decomposition in operations is associated with the transformations of state between levels of the product structure. Each transformation could be formalized as:

$$t_j, j = 1,...,n .$$

In this model, stage 2 corresponds to the definition of the domain of available production resources (*res*) able to execute transformations of the product structure:

$$res_k, k = 1,...,r .$$

In stage 3, resources are selected for each transformation, according to a criteria defined for the developed system. In this, case minimum throughput time will be used. So, for each transformation a processing time T_{jk} is determined that depends on unit processing time c_{jk} and the number of times w_{jk} a resource must execute the transformation (number of units for each order). This is represented by the following equation:

$$\forall j \in \{1,...,n\}, \forall k \in \{1,...,r\} : T_{jk} = c_{jk} \cdot w_{jk} .$$

This transformation processing time is used by each resource to, according his free agenda, calculate the throughput time.

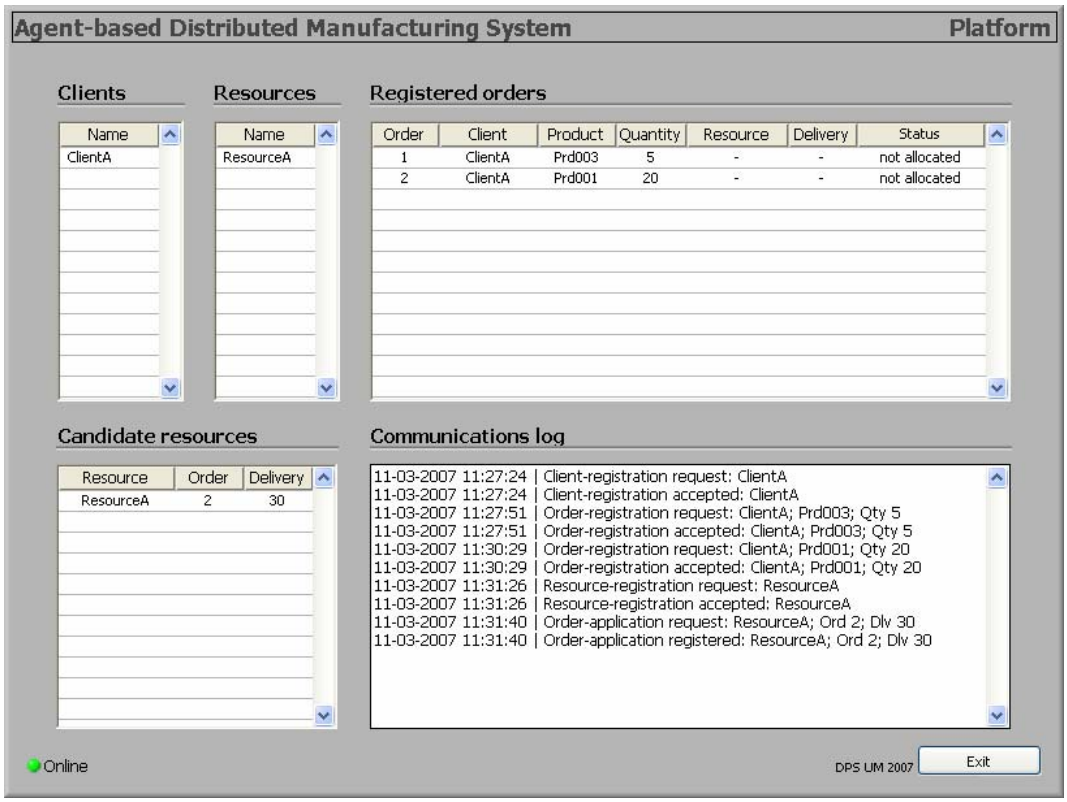

Figure 2: User interface for platform agent (main panel).

The solution will also be related with stage 5, because initial and final processing time for each transformation will be determined, and so a dynamic schedule solution will be obtained. Stage 4 and "coordination" aspects from stage 5 are related with operational strategies. In this work it is made a direct address of the problem from the point of view of the interoperation of the PPC processes with automation processes controlling production equipment.

Testing the validity of these concepts could be addressed by a functional prototype. Software agent paradigm was used for system development, as described in [5], to demonstrate the validity of the dynamic PPC concepts. Furthermore, the interoperation of production planning processes with physical production resources is an objective for the development of the system, described in this paper.

3. Implementation model

The developed prototype was designated as Distributed Manufacturing System (DMS), and implements part of the dynamic PPC model described on the previous section.

3.1. Basic structure

Three types of agents were implemented, one for each entity of the model: client, resource and platform (Figure 1). Software agents involved in this work are able to: execute some product transformation (autonomy); monitor the environment through the communication platform or the state of the resources that are represented by the agent (monitoring); and interact with the environment or other agents (action) based on design objectives (goals). These are characteristics of agents as defined in [8], [9] or [10].

Although, the existence of the platform agent could induce a complete centralization of management, that is not true (Figure 1). In fact, all the decision processes concerning orders, resource allocation, production, etc., are performed by client and resource agents. The platform manages the underlying services (e.g. client and resource register/login, order registration requests, etc.), and maintains a database containing all the relevant data (including a communications log). This platform agent, as relating the communications between agents, acts as a Blackboard. Figure 2 shows the main panel of the platform agent user interface.

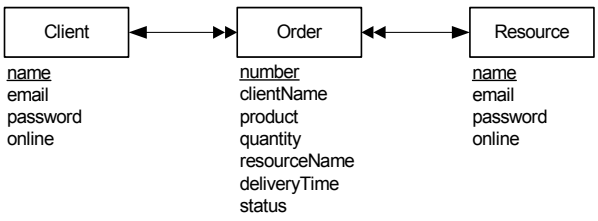

Figure 3: Entity-relationship diagram for platform agent's database.

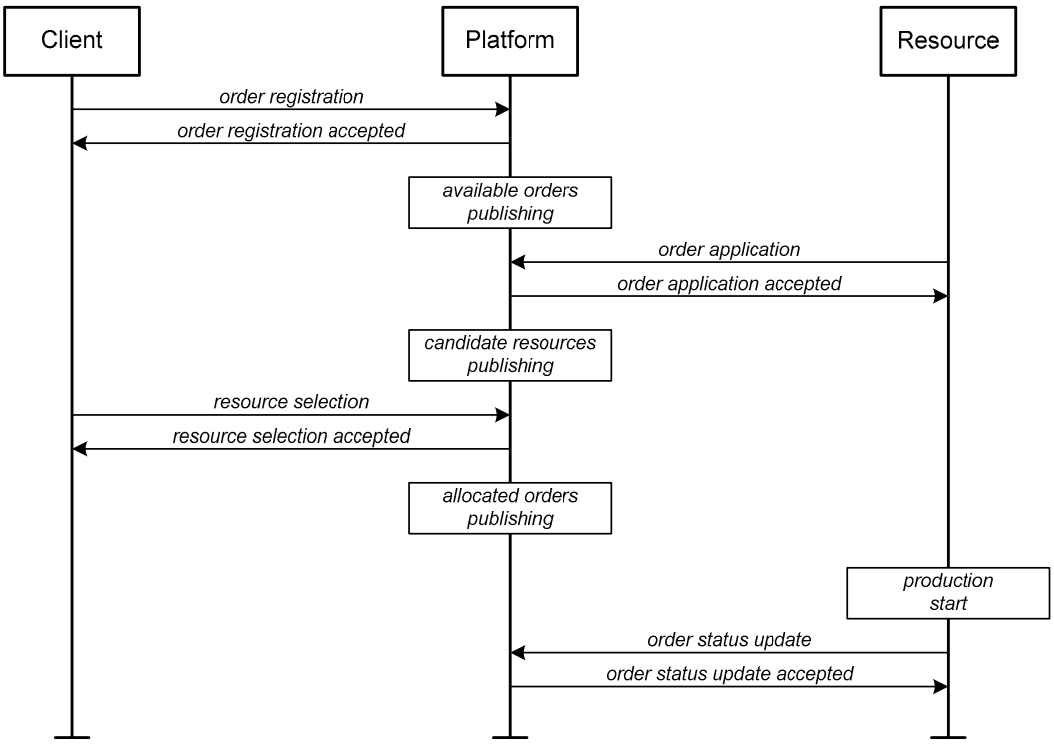

Figure 4: Message sequence chart for order dealing.

The Ontology of this multi-agent system is embedded in code and some of the fundamental concepts are registered on a database, namely: Client, Order and Resource. The database structure is very simple and it is represented on Figure 3. For each order there exists only one Client and one Resource selected for that order. That order represents operations of the transformation process of the product structure that can be completely executed by an autonomous production resource.

The structure of the messages exchanged within the system is fixed - a string array with five elements: origin/destination, operation code and three parameters. All the messages posted by clients and resources are sent to the platform agent. Thus, for those cases, the first string represents the message origin (clientName or resourceName). All the messages received by clients and resources are posted by the platform, and thus, for those cases, the first string identifies the message destination (clientName or resourceName).

The use, and semantics, of the parameters depends on the message operation code. The current prototype implements thirty one different operations (e.g. login request, order application registration, production status update, etc.). To avoid data loss, every sent message demands a response message, and, thus, the sender can always check if its message has reached the destination (the only exception is the platform shutdown indication message).

Among other techniques/tools, Message Sequence Charts (MSC) where used for prototype specification. As an example, Figure 4 represents a typical operation scenario, described on the next subsection.

The publishing mechanisms implemented on the platform are in fact information broadcasts. Thus, without specific request, any registered agent can access that information whenever he needs. When a given agent (client or resource) leaves the system (executing the logout procedure), the platform sends him an email with the communications log which includes all the relevant information (e.g. registered orders with respective products and quantities, order applications with respective candidate resources and delivery times, production status, etc.). Further processing is possible, as this information is represented in a plain text file which can be easily imported by any data processing application.

3.2. Operation

Using the client agent interface (Figure 5), a client builds up an order, selecting the desired product and quantity, and sends it to the system. The platform agent registers the incoming order under a sequential order number, sets the correspondent status to “not allocated” (Figure 2), and notifies the client about the successfulness of the procedure. The sequence of exchanged messages between the client and the platform during this procedure is represented on Figure 4, and the correspondent message contents are represented on Figure 6.

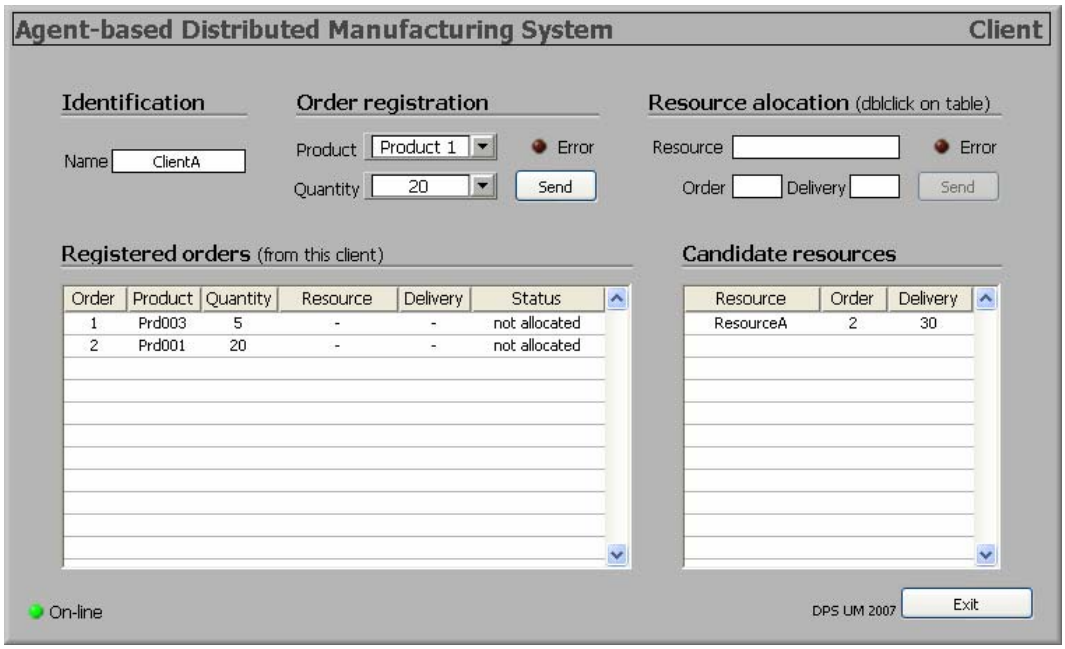

Figure 5: User interface for client agents (main panel).

All the registered orders are published by the platform, and those having the "not allocated" status become available not only for resources' application, but also for clients' order registration double check. In fact, the contents of the registered orders display on the client agent interface (Figure 5) results from the subscription of the information published by the platform, filtered by the client name.

Whenever a "not allocated" order is introduced into the system, one or more resources may apply to process that order, providing the correspondent delivery date. The platform displays and publishes the list of all candidate resources, for every "not allocated" order (Figure 2). The client automatically subscribes that information, filters the candidates for its own orders, displays them on the candidate resources display (Figure 5) , and finally decides, using a given decision criteria, which resource is going to process the order (if there is more than one candidate resource).

clientName	*operationCode*	*product*	*quantity*	*n.a.*

(a)

clientName	*operationCode*	*orderNumber*	*product*	*quanttiy*

(b)

Figure 6: Messages involved on the order registration procedure: (a) client request; (b) platform confirmation.

In the current prototype the resource allocation procedure is manual, but it can be easily automated using the shortest delivery time as decision criteria. When the client successfully selects a resource to process (Figure 5) an order, the platform clears the list of candidate resources for that order, updates the registered orders display (introducing resource name and delivery time and setting the status to "0% done"), and publishes all that information. Thus all the agents know that the resource allocation procedure has been accomplished for that particular order.

According to the developed PPC model, the application process should be open during a given time window. In this prototype, that period starts when the order is registered and ends when the client succeeds to manually select a candidate resource to process that order. To implement automatic resource allocation, the referred time window must be set (eventually by the client itself, but not necessarily) – when time expires, candidate resources are analysed and the decision criteria is applied.

A resource agent should continuously monitor the list of available orders, and decide which of them he is going to apply for. The resource agent maintains an updated available orders display (Figure 7) by subscribing the contents of the platform' registered orders display (Figure 2) and filtering it for orders with "not allocated" status. Once again, in the current prototype the selection procedure (this time of orders to apply for) is manual - automation can be achieved after definition of decision criteria (e.g. production skills, scheduling plan, client, etc.).

To apply for a given "not allocated" order, the resource should provide a delivery time. According to the developed PPC model, every resource should have an internal agenda allowing thus the automatic determination of the delivery time for a given order. Alternatively the client could receive the agenda of each candidate resource and, based on that information, decide which one is the most adequate. In the current prototype the agenda concept is not implemented, so the resource should manually provide a delivery time whenever he wants to apply for a given order.

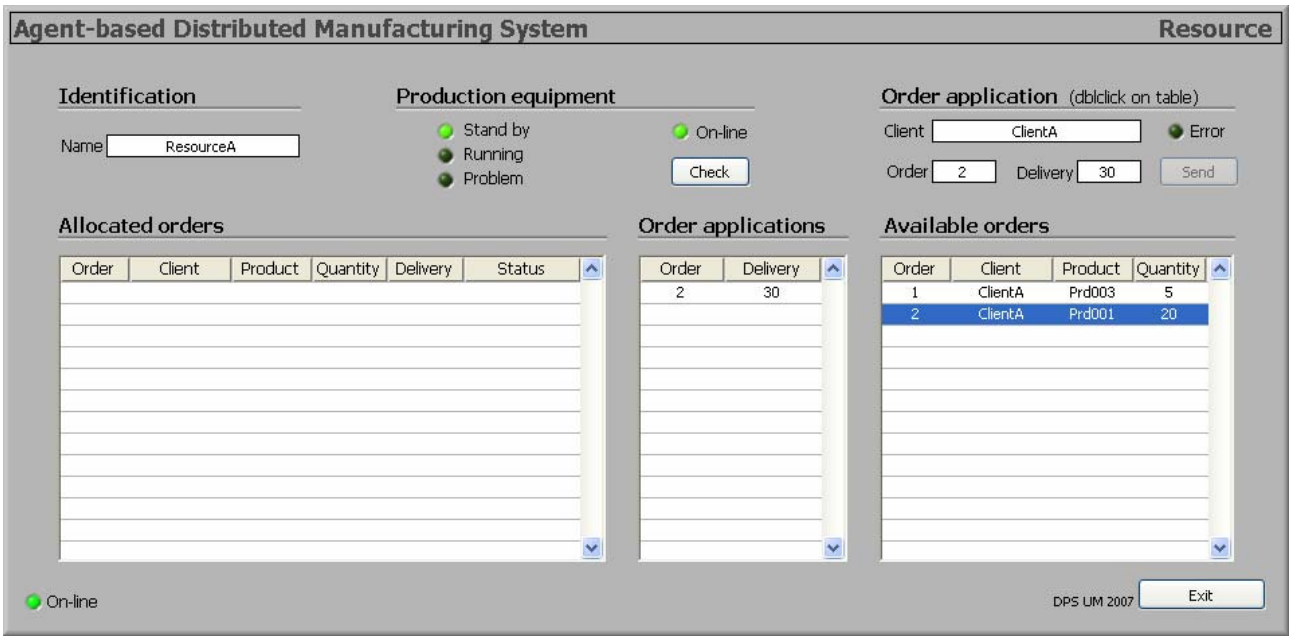

Figure 7: User interface for resource agents (main panel).

After receiving an order application (Figure 4), the platform updates the candidate resources display (Figure 2) and publishes those contents. Thus, the candidate resources display of the client agent (Figure 5) and the order applications display of the resource agent (Figure 7) are both updated.

3.3. Automation equipment

Typically a resource agent includes some kind of manufacturing equipment, being thus able to execute the physical operations necessary to fulfil clients' orders. In this prototype one of the resource agents includes an industrial PLC (Programmable Logic Controller) which controls a toy-model milling machine (Figure 8).

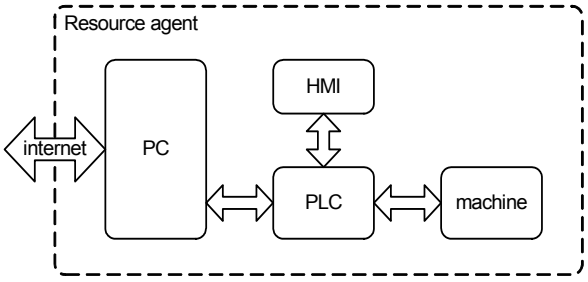

Figure 8: Resource agent's architecture

The PLC is a Simatic S7-200 CPU224XP, with fourteen digital inputs, ten digital outputs and two RS485 communication ports. One of the ports is connected to a TD200 HMI (Human Machine Interface) device which allows the monitoring of the production status (Figure 9). The toy-model milling machine was build up with Lego Technics / Mindstorms components and includes two DC motors (driving spindle and vertical axis) and two contact sensors (defining vertical axis upper and lower limits).

The objective of this prototype is the demonstration of the developed model, which includes the integration of shop-floor equipment.

Although both PLC and HMI device are industrial equipments, the described resource agent is not intended, obviously, to produce a physical product. When the resource receives an order allocation, the PC (Figure 8) instructs the PLC to start the production, providing the necessary information (order number, product and quantity). The PLC starts the production cycle driving the toy-model milling machine through a pre-defined sequence of operations. Simultaneously, the TD200 HMI device displays the production status (order number, product, quantity and execution percentage) including an eventual problem situation. A "failure switch" was implemented, allowing thus disturb introduction. The production status is propagated from resource to client, via the platform agent, until the order becomes 100% fulfilled.

The software developed for the PLC was implemented using ladder logic and has two main components: (1) communications and (2) toy-model milling machine control and monitoring.

Communication between PC and PLC (Figure 8) uses a RS232 port (PC side) and the second RS485 port on the PLC, requiring thus a RS232/485 converter. This kind of solution is widely used in industry as the RS485 standard allows communication distances up to 1200m at 100Kbit/s. Special attention has been paid to the implementation of the communications component, which is based on interrupts, allowing the CPU of the PLC to dedicate the most part of his time to production equipment control.

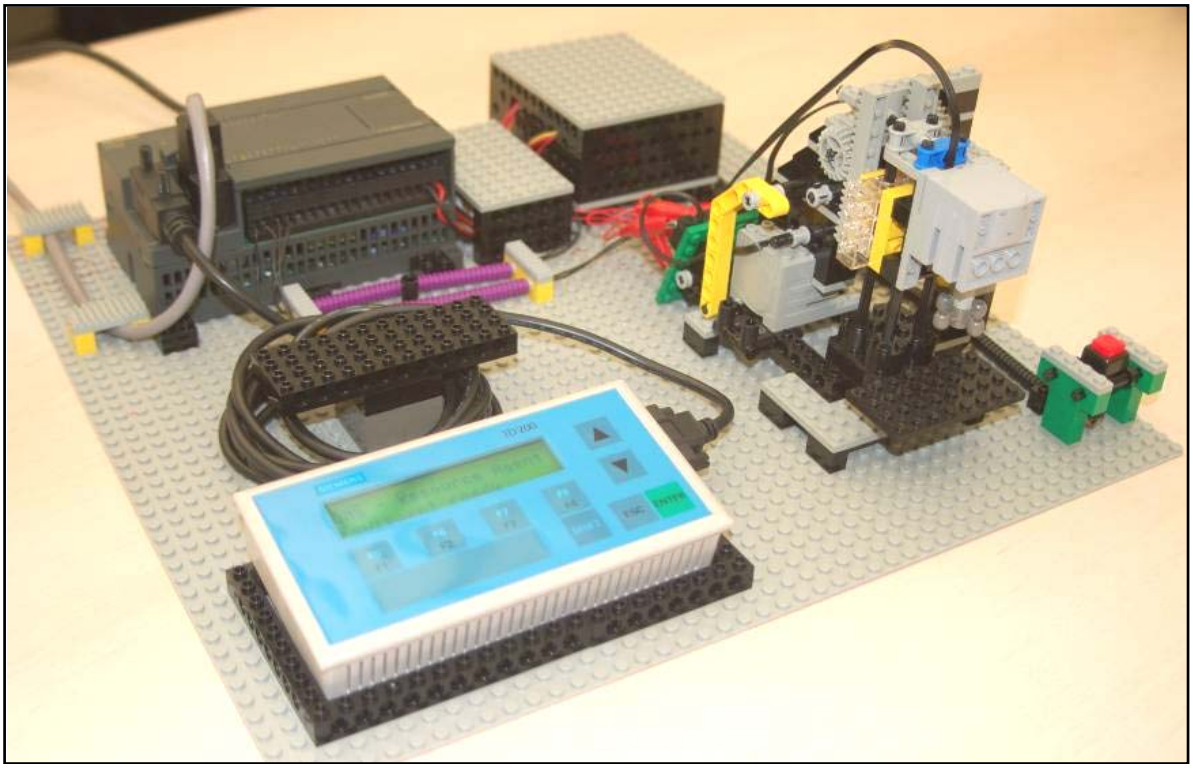

Figure 9: Industrial PLC, HMI device and toy-model milling machine.

The specification of the toy-model milling machine control and monitoring is based on a simple Finite State Machine (FSM) which basically drives the machine through three distinct states, constituting a production cycle of about 30s.

3.4. Testing prototype utilization

At this point a fundamental verification test related with functional acceptance of the prototype is required. The two main objectives of this test are: the prototype can have a full distributed utilization; the interoperation between dynamic PPC planning functions and physical execution of automation equipment is in accordance with design objectives.

One of the experiments has involved twenty two undergraduate students - client (Figure 5) and resource (Figure 7) applications were provided. Some students have decided to register themselves as clients, while others as resources, and they were asked to introduce orders (clients), apply for orders production (resources), introduce machine failure (resource with the production equipment), leave and re-enter into the system, etc. This experiment, performed within the university campus network, was an important contribution to the operational test of the DMS system with several agents logged in.

Other experiments were successfully worked out in order to test the distribution perspective, with the platform agent running on the university campus and some agents (clients and resources) installed outside campus (e.g. 50Km away). The results were very positive and the DMS system – which implements a fraction of the developed model - is fully operational.

Both experiments results showed that the prototype can be further developed in order to be used as an experimental platform for Dynamic PPC and interoperation with automation equipment.

4. Conclusion

The presented dynamic PPC model can be used as a framework for the development of autonomous, dynamic and reconfigurable PPC systems. For each client order of a product, a configuration of production resources is proposed. This is closely related with the product structure, with the allocation of an autonomous production resource for each transformation process delivering an item of the referred structure. So, product structure is used as the key element for production system configuration, in this case a system configuration for each order. Furthermore, a system configuration can be proposed for each item of the product structure. An adequate system configuration must rely on the selection and allocation of the right production resources. Dynamic selection and allocation can be, in some cases, an automatic process. This goal is being pursued in this project, and this paper demonstrates that the PPC model can contribute to the interoperation between planning processes and automation equipment.

The developed prototype has allowed the demonstration of two different perspectives: (1) the

adequacy of the selected programming approach (LabVIEW), including the correspondent communications technology (DataSocket), and, (2) the feasibility of the developed PPC model, which includes the integration with manufacturing automation equipment. Several experiments, involving different scenarios (e.g. geographically distributed agents, production equipment failure, etc.), were successfully carried out.

As immediate future work, involving undergraduate and graduate students, some refinements and improvements are already defined at both model level and prototype level. The experiments carried out have revealed some interesting aspects which can now be subject of further developments, namely the implementation of decision criteria for resource allocation, and of an agenda within each resource.

References

[1] H. Yanli, Y. Haicheng, H. Weiping, Z. Wei, and H. Xinping, "Flexible Workflow Driven Job Shop Manufacturing Execution and Automation Based on Multi Agent System," presented at Intelligent Agent Technology, 2006. IAT '06, 2006.

[2] D. Wang, S. V. Nagalingam, and G. C. I. Lin, "Development of an agent-based Virtual CIM architecture for small to medium manufacturers," *Robotics and Computer-Integrated Manufacturing*, vol. 23, pp. 1-16, 2007.

[3] M. Mahesh, S. K. Ong, A. Y. C. Nee, J. Y. H. Fuh, and Y. F. Zhang, "Towards a generic distributed and collaborative digital manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 23, pp. 267-275, 2007.

[4] M. Bruccoleri, P. Renna, and G. Perrone, "Reconfiguration: a key to handle exceptions and performance deteriorations in manufacturing operations," *International Journal of Production Research*, vol. 43, pp. 4125 - 4145, 2005.

[5] R. M. Lima, R. M. Sousa, and P. J. Martins, "Distributed production planning and control agent-based system," *International Journal of Production Research*, vol. 44, pp. 3693-3709, 2006.

[6] R. G. Smith, "The Contract Net Protocol - High-Level Communication and Control in a Distributed Problem Solver," *Ieee Transactions on Computers*, vol. 29, pp. 1104-1113, 1980.

[7] A. Tharumarajah, "Survey of resource allocation methods for distributed manufacturing systems," *Production Planning & Control*, vol. 12, pp. 58-68, 2001.

[8] N. R. Jennings and M. Wooldridge, "Applications of Intelligent Agents," in *Agent Technology Foundations, Applications and Markets*, N. R. Jennings and M. Wooldridge, Eds.: Springer Verlag, 1998, pp. 3-27.

[9] H. V. D. Parunak, J. Sauter, M. Fleischer, and A. Ward, "The RAPPID Project: Symbiosis between Industrial Requirements and MAS Research," *Autonomous Agents and Multi-Agent Systems*, vol. 2, pp. 111-140, 1999.

[10] J. Tweedale, N. Ichalkaranje, C. Sioutis, B. Jarvis, A. Consoli, and G. Phillips-Wren, "Innovations in multi-agent systems," *Journal of Network and Computer Applications*, vol. 30, pp. 1089-1115, 2007.