Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

An Implementation Environment for Automated Manufacturing Systems

Ricardo A. Diogo, Carlos A. Vicari, Eduardo de F. R. Loures, Marco A. Busetti, Eduardo A. P. Santos

Pontifical Catholic University of Paraná, Curitiba, CO 80215-901 Brazil
(Tel: +55(41)3271-1333; e-mail: r.diogo, carlos.vicari, eduardo.loures, marco.busetti, eduardo.portela@pucpr.br)

Abstract: This paper presents an implementation environment applied to reconfigurable processes in automated manufacturing systems. This environment is based on a methodology which consists of a cyclic three-stage development – modeling, synthesis and implementation – until the real system accomplishes the required specification, resulting in the project of the automated system. This kind of development allows a continuous result revision of each stage. The present paper describes the methodology and details of components of the proposed environment.

1. INTRODUCTION

In order to be competitive, manufacturing, more than any other activity area in economy, needs to continuously adapt to changes in the market. The increase in global competition is pushing enterprises to reduce the time response when launching new products and to offer competitive prices. Diversity, fluctuations in demand, the short life cycle of products due to the frequent introduction of new needs, in addition to the increase in the client's expectations in terms of quality and delivery time, are nowadays the main challenges with which companies have to deal in order to keep competitiveness and stay in the market.

Moore et al. (2003) add that the system flexibility is related to a control system's implementation time. Generally, the system's flexibility imposes that the implementation time for new applications, which can demand the reconfiguration of software and hardware, be as short as possible, since new products invariably have new requirements and specifications related to automation, layout and integration. In order to approach the flexibility issue in the context of reconfigurable processes, it is essential to establish systematic procedures that may characterize the development cycle of a control system. This systematization consists of using formal models for analysis, synthesis and implementation of control systems for Discrete Event Systems (DES). A DES can be defined as a dynamic system that evolves according to the occurrence of events. A classical example of DES is a manufacturing system (Cassandras & Lafortune, 1999).

The model proposed by Ramadge & Wonham (1989) allows an automatic control synthesis process instead of the usual manual and heuristic procedures. In addition to this advantage, the synthesis procedure demands that the obtained supervisor must always fulfill the control specifications. This way, new control systems may be rapidly and automatically designed when modifications, such as redefinition of specifications and physical changes, are necessary. For these reasons, the present work uses the Supervisory Control Theory (SCT) as a formal tool to obtain the supervisors for the manufacturing automated systems.

This work initially presents an approach for the project and for the development of the control system in a context of reconfigurable processes. The contribution of the proposed approach consists of treating the manufacturing automated systems project with higher efficiency, effectiveness and reliability when new applications are necessary. Such applications might come from the insertion of new products, from the reconfiguration of processes and existing products, from new demanding necessities or from technological modernization, among others.

Secondly, this paper presents its main proposal: an environment for the accomplishment of the third step in the development cycle-implementation. Such environment makes use of a software and hardware devices in order to progressively implement and validate the control structure based on the SCT, and obtained in the synthesis stage. In this environment, simulation is systematically used. Furthermore, the present work introduces the Dynamic Environment (DE), a SCADA-based software (Supervisory Control And Data Acquisition) which aims for the integration of both the virtual (simulation) and real models (plant controllers). In order to develop the DE and the simulation environment and integrate them with the physical system, some concepts from Hibino et al (2006), Holst et al (2000), Holst et al (2001) and Min et al (2002) were studied.

This paper is structured as follows: section 2 introduces SCT fundamentals; section 3 presents the development cycle proposed to the design of reconfigurable automated systems; section 4 presents an application example; section 5 details the implementation stage, presenting all the software and hardware devices used; finally, chapter 6 brings the paper's final conclusions, drawbacks and perspectives.

2. SUPERVISORY CONTROL THEORY (SCT)

The SCT (Ramadge & Wonham, 1989) was developed as a proposal for a formal methodology for the automatic synthesis of optimum controllers for DES. The main premise of a SCT is the plant and specification modeling by means of Formal Languages and Automata (Carrol & Long, 1989). The resulting supervisor generates the maximum controllable

language, i.e., the most permissible control action. In this theory, a system (called a plant) is modeled by an automaton, and it is assumed to have an uncontrolled behavior that may violate some required properties (e.g., safety). Hence, this behavior has to be modified by means of a feedback controller (a supervisor) in order to achieve a given set of requirements (Ramadge & Wonham, 1989). In order to do that, the supervisor acts on the plant by forbidding some discrete events and allowing others.

Queiroz and Cury (2000) extend the SCT with a modular approach named Local Modular Control (LMC). This approach takes advantage of the natural modularity of the physical system and control specifications for the synthesis of supervisors. Queiroz and Cury (2002) propose an architecture for structuring a modular supervisory Control System (CS). According to this architecture, the logical structure of the control program is composed of three levels: Modular Supervisors (MS), Product System (PS) and Operational Procedures (OP). Such levels carry out the control action of the supervisors and act as an interface between the theoretical model and the physical system to be controlled. This control structure is shown in Figure 1.

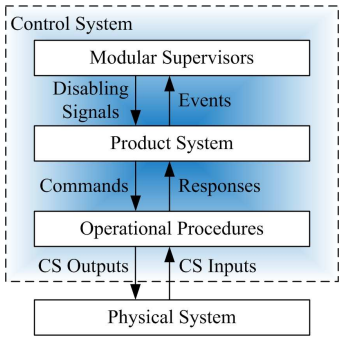

Fig. 1. Control architecture proposed by Queiroz and Cury

Based on architecture proposed by Queiroz and Cury (2002), Vieira et al. (2006) define a model for Programmable Logic Controller (PLC) implementation that preserves the natural modularity of the physical system and synthesized supervisors. This model leads the designer to systematically convert the set of supervisors and the set of automata describing the free behavior of the physical system into a set of Sequential Function Charts (SFC). The obtained PLC program is in accordance with the IEC 61131-3 (1993). The LMC architecture proposed by Queiroz and Cury (2002) and the implementation method proposed by Vieira et al. (2006) are used in our implementation environment as fundamental theoretical tools.

3. PROPOSED DEVELOPMENT APPROACH

The proposed implementation environment supports the development cycle of control system used by our research group. Figure 2 shows the development cycle, which is characterized by three stages: modeling, synthesis and implementation. In the modeling stage we select from the subsystem and specification libraries a set of models to represent the real system and the application, respectively. In the synthesis stage, those models are used to generate MS, according to the SCT (Ramadge & Wonham, 1989) and the LMC. In the implementation stage, the three levels of control structures are integrated according to Queiroz and Cury (2002) and gradually implemented in three steps: simulation, simulation and insertion of Control and Communication Technologies (CCT), and execution.

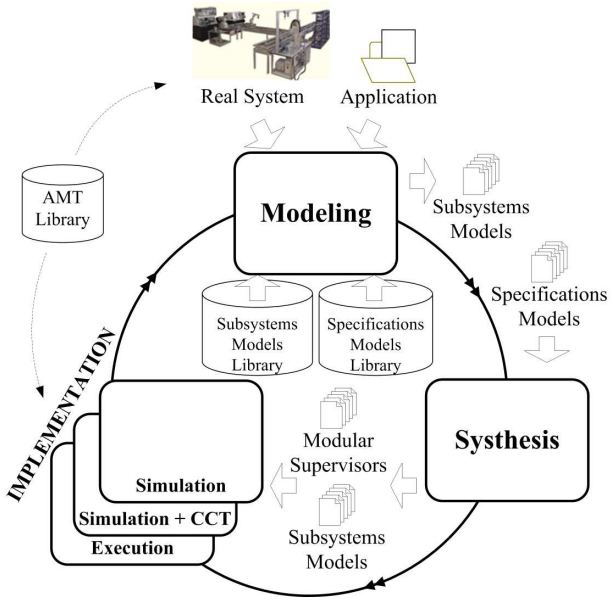

Fig. 2. Development cycle proposed

The control system development occurs cyclically in three stages – modeling, synthesis and implementation – up to the moment in which it complies with the real system's demanded application, thus resulting in an integrated and automated system. This development mode allows a continuous review of the results obtained in each step. By doing this, the designer can receive a new application (for example, a need for processes reconfiguration) and select new specification or subsystem models, which will appropriately comply with this new application.

4. AN APPLICATION EXAMPLE

This section presents an application example that will be employed to illustrate the implementation environment proposed in this paper. This is example is used in this paper as a basis for the experiment accomplishment. Figure 3 presents this system, which is made up of three machines and two buffers, one between the machines 1 and 3, and another one between machines 2 and 3. The automata associated to the machines are also shown in figure 3, where: M1 in state 0 means machine not working and in state 1, working; the same occurs to M2; M3 in state 0 means robot manipulator not working, in state 1, taking piece from B1 to B2, and in state 2, returning to the not-working position. The α events are controllable events (beginning of operation), β and μ events are uncontrollable events (end of operation). According to SCT, controllable events can be disabled by supervisors and uncontrollable events cannot.

The objective is to use, in its utmost, the available resources in a correct and safe way. This way, the processing of several pieces in the system cannot cause buffers underflow or overflow. These two specifications are represented by the automata shown in figure 4. This figure also presents the

reduced MS obtained by the synthesis procedure described in Vaz and Wonham (1986). The events into a square are disabled events. They cannot occur in the corresponding states of MS.

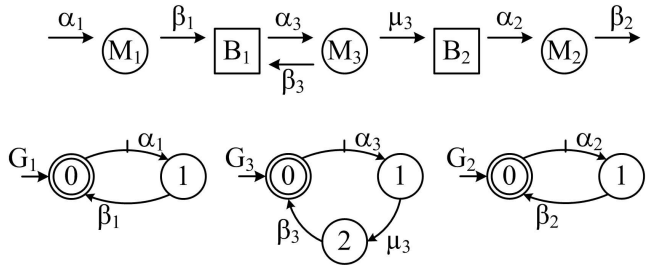

Fig. 3. Subsystems models

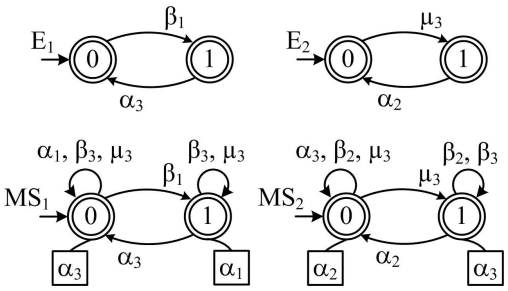

Fig. 4. Specifications and reduced modular supervisors

5. IMPLEMENTATION STAGE

This section describes the dynamic environment which is used to support the implementation stage of development cycle (see Figure 1). Firstly, the automata corresponding to obtained reduced supervisors and the subsystems models are implemented in the control structure according to Queiroz and Cury (2002). Also, this structure is coded into a PLC program according to Vieira et al. (2006). To do that, it is necessary to stablishes the variables necessary to code the control structure. See table 1 with *i = 1, 2, 3*.

Table 1. Control Structure Variables

Controllable event	Associated command	Uncontrollable event	Associated response
α_i	$Cmd\alpha_i$	β_i	$Rps\beta_i$
		μ_i	$Rps\mu_i$

According to Figure 2, the implementation stage includes three steps: Simulation, Simulation + CCT and Execution. In the implementation stage's first step, we simulate the CS' three levels (MS, PS and OP). To do that, the PLC is attached to a plant simulator (Arena®) (Rockwell Automation, 2007) so that both may exchange signals (commands and responses). In the simulator, the subsystems are implemented according to the real system's configuration, which was used for the PS generation. This way, the simulation occurs according to the restrictions imposed by the MS. The simulation result allows us to evaluate the completion and correctness in the subsystem or specification models. The control structure simulation is useful either to do the first validation of the built models (subsystems and specifications) or to detect modifications and the necessary inclusions. The designer may start to run the MS level and the PS, and follow the evolution of the states and associated control actions.

In the implementation stage's second step, the subsystems implemented in the plant simulator are progressively substituted by real components of the plant that runs the real OP (e.g. PLC, PC-based control). This way, the PLC communicates simultaneously with the plant simulator and the real subsystems by inserting CCT. In this step, it is possible to progressively validate the control structure (software), when it is connected to the real system, and to analyse questions related to the distribution of the physical control system (hardware).

In the third step, the devices (e.g. PLC, PC-based control) that implement the OP are completely attached to the respective sensors and actuators in the real subsystems.

5.1 Dynamic Environment

To make the implementation's three stages possible, a computer platform must be conceived in order to establish the communication between the PLC and the Simulator. This platform is the DE (illustrated on figure 5). This consists of a computer platform that accomplishes several operations: *i*) It sends to the Simulator the commands ($Cmd\alpha_i$) generated in the Central PLC; *ii)* It receives acknowledgements ($AckCmd\alpha_i$) from the Simulator. This is done to turn off the corresponding command in the Central PLC; *iii)* It receives responses ($Rps\beta_i$) from the Simulator and sends to the Central PLC to generate the corresponding uncontrollable events; *iv)* It sends to Simulator acknowledgements ($AckRps\beta_i$) from the Central PLC. This is done to turn off in the Simulator the corresponding reply; *v)* It has a interface to switch between the build-in simulation and the real environment. However, the simulation carries on running with the real subsystem. So, DE reverses the responses and corresponding acknowledgments when a real subsystem is connected. This is done to stop the generation of responses by the Simulator. In this case the Simulator receives the responses from the corresponding real OP and generated the acknowledgement to the Local PLC. By using this technique, it is possible to retrieve real information from the manufacturing system.

DE was made use of a SCADA software named Elipse E3 (Elipse Software, 2007) to accomplish the DE. This platform was chosen due to the fact that it comprises programs which enable the communication with peripheral devices (*drivers*), it enables system integration with other software via Data Base (DB), and mainly, due to the feasibility of generating program libraries. Particularly, this feature allows us to store subsystems' models (and their respective signals, according to the three-level implementation model).

The methodology presented in this paper makes use of a Dynamic Environment Communication Model Library (DECML) (see figure 5). This library stores models from the proposed development modeling cycle's stage (see figure 2). In so doing, implementation projects may make use of project models already accomplished.

5.2 Simulation and the Dynamic Environment

According to the development cycle shown in figure 2, the first implementation step corresponds to the simulation of the

plant under study. As described in section 3, simulation takes place by considering restrictions imposed by the MS. And to make this approach possible, the DE is connected to the Simulator, so that they switch signals between them. Each DECML model comprises a pair of communication channels. In that case, the DE's goal is enabling the simulation to occur from events sent by the central PLC. Also, it is necessary to conceive an internal codification in the Simulator, so that both the communication with the DE and the simulation geared to external events be possible.

As described before, Arena® was the simulation environment chosen. Based on an architecture divided into levels as stated by ISO TR10314-1, an adaptation of the model proposed by Hinino et al (2006) is intended. This way, the simulation is divided into three levels, according to figure 6.

The *interface level* is the first one, and it is responsible for the communication between the simulation environment and the DE. Its main purpose is to perform the reading of the commands sent by the DE, process them, and run the OP related to each command. The commands sent from the PLC to the DE are stored in a DB, and read by the Simulator afterwards. Also in the Simulator, a codified decision-taking structure defines which commands have a true logical value. In that case, the interface level sends a signal to the OP level, otherwise it waits for a new DB reading.

The second level is called OP level, and it is responsible for sending the commands from the interface level to the virtual models level. In order to minimize the data processing time, the event linked to a reply will be directly sent from the DE to the DB, without being processed by the interface level. At this level, log generation is intended. Such logs should contain quantitative data over the plant's productive capacity. Thus, not only the control structure's validation is possible, but also the decision-taking over the available resources' planning in the plant.

The third level is the Virtual Models level. It is responsible for the simulation of the subsystems, which make up the plant under study. In that case, this level has a direct communication with the OP's level, both sending and receiving commands and responses, respectively. At this level, virtual models are formed by taking into consideration the physical disposition of both the real plant and the PS model. This way, for every subsystem model which is part of the PS, there is a corresponding virtual subsystem.

According to the implementation stage proposal, OP may be simulated one by one, as shown in figure 7. Firstly, only one virtual model is simulated, after that the new models are gradually being simulated until achieve the full system simulation. By doing that, it is possible to gradually determine if the control action imposed by the LMC is correct, or if the designer has fully achieved specifications. OP's gradual insertion enables a systematization-enhancing of the obtained CS' verification, and a reliance-enhancing as real subsystems are inserted.

5.3 Simulation + CCT and the Dynamic Environment

According to the development cycle proposed, CCT are inserted in the implementation's second stage, so that the central PLC communicates with the simulator and enables data transfer between them. These technologies may cover from industrial communication networks to data base, computer networks (Ethernet), among others. The industrial network is used for data exchange among PLC, the DE and the Ethernet network used for exchanging information between the DE and the Simulation system. Some of these technologies are part of the implementation environment since the first stage, once in that stage the central PLC must be

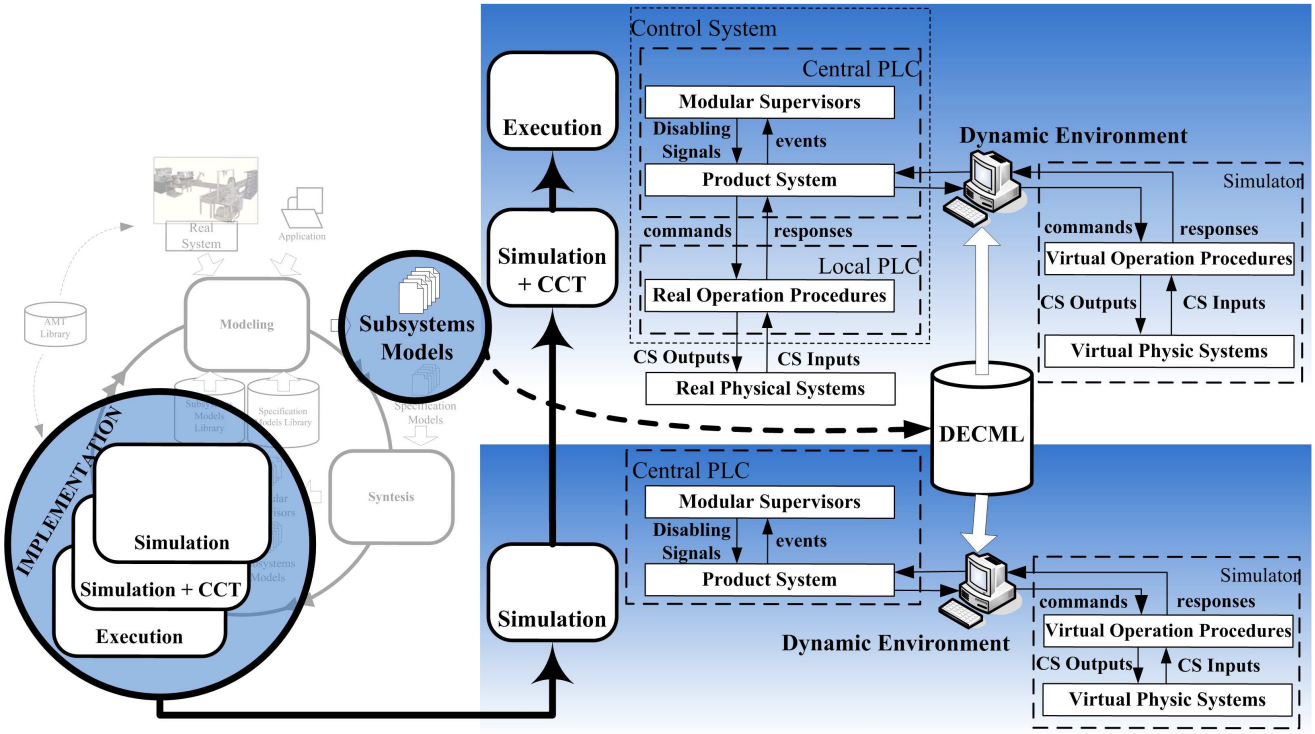

Fig. 5. Implementation details

connected to the DE. It is considered, however, that the main insertion refers to the connection of technologies associated with the real plant.

Gradually, Real Physical Subsystems (RPS) are inserted in the communication structure, as illustrated in figure 8. Each RPS controller runs a related OP. The signal exchange between the central PLC, responsible for the supervisory control structure, and the local PLC, which run the OP, is done by means of commands and responses, according to Vieira et al. (2006). This way, RPS are progressively performed, parallel to the virtual subsystems simulation. During this stage the DE thus acts as the PLC (central and local) and the Simulator's integrator.

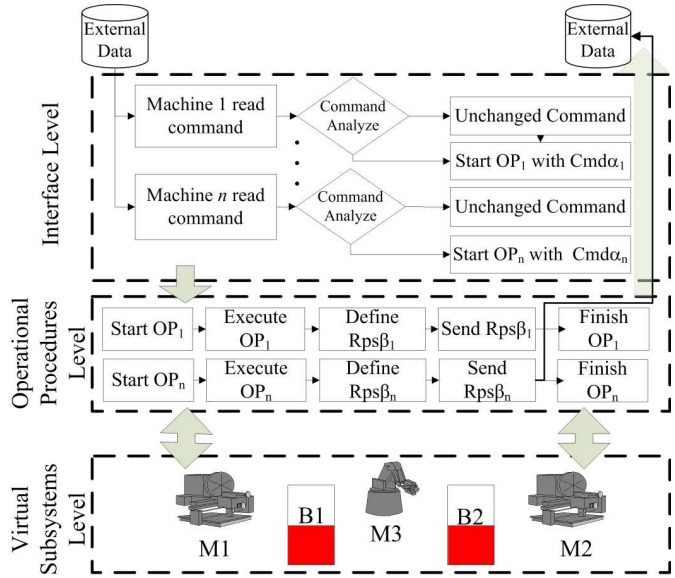

Fig. 6. Levels of simulation environment

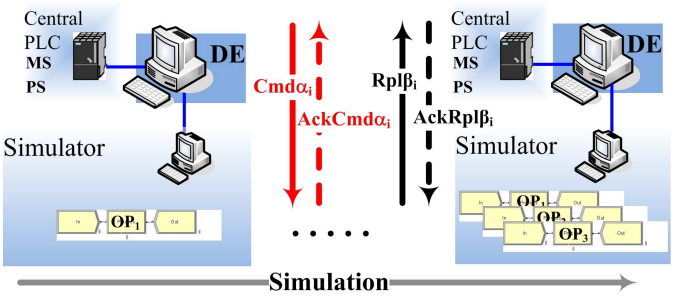

Fig. 7. Evolution of the Simulation

The reply originated from simulation, as a result of the Virtual Physical Subsystem (VPS) operation, must be sent to the CS only under *off-line* simulation, as shown in figure 7. In that case, the simulator executes only the VPS. In case of an *on-line* simulation, a RPS generates the reply after the related operation is finished, as demonstrated in figure 8. Therefore, the reply-related channel must process such responses to the CS only if the implementation environment is under the *off-line* simulation mode. RPS responses must, then, be processed under *on-line* simulation. In this case, the reply must be sent to the virtual system, since simulation synchronization is intended to obtain the time between the beginning and ending of a subsystem cycle. By using synchronization, it is possible to obtain the approximate time in which manufacturing system's operations are executed. The store data can be used in the future to determinate the manufacturing needs. However, this synchronization does not occur in real-time, because the DE is applied in a non-real-time operational system.

According to Bullock et al (2004), in off-line simulation system, the operations are executed as promptly as possible. Depending on the operational system and computational hardware used, it is possible to simulate hours, days, weeks and so on in few seconds. That is done to decrease the project lead-time of manufacturing systems. If a physical subsystem is inserted, it will run according to the time allowed by its physical configuration. However, the part of the system which is still simulated can be executed as soon as possible.

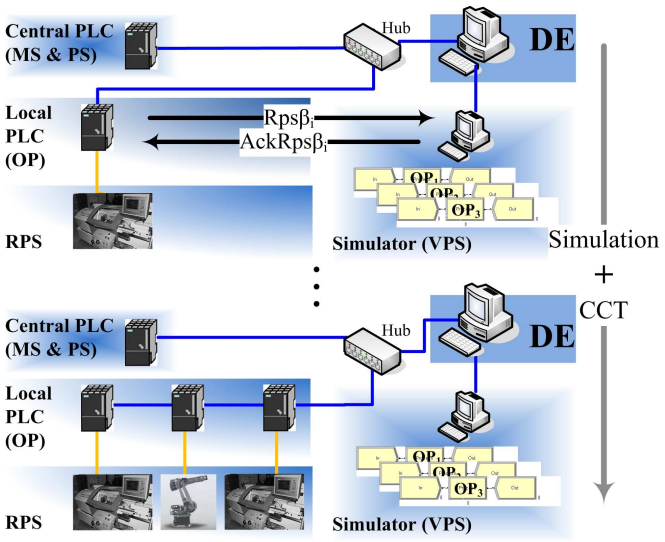

Fig. 8. Evolution of the Simulation + CCT

By using the manufacturing real subsystems' gradual insertion, the CCT expansion occurs as a result of local PLC insertion and, consequently, of the industrial network used.

The implementation of all local PLC (and the associated RPS) enables the global behavior of the manufacturing plant to be evaluated. In this step, it is possible to progressively validate the control structure (software), when it is connected to the real system, and to analyse questions related to the distribution of the physical control system (hardware). It is also possible to get data from the physical system through the simulation environment. This way, it is possible to determine productive needs and capacity, and consequently verify whether the manufacturing system is adapted to supply the production demand.

5.4 Execution and the Dynamic Environment

At this point, all RPS are connected to the implementation environment and the plant's global behavior can be fully analyzed. It means that the devices (e.g. PLC, PC-based control) that implement the OP are completely attached to the respective sensors and actuators in the real subsystems. The responses generated by the local PLC are sent to the central PLC and to the Simulator (non-controllable events synchronization). In so doing, responses geared to the CS do not pass through the DE, only the ones geared to simulation, as shown in figure 8.

Still at this stage, productive processes can be continuously analyzed in the simulation environment by getting the production lead-times obtained through the DE. According to the logs returned, new capacities and productive needs can be determined. If that happens, the whole Development Cycle is started, but only to determine new models of subsystems and new operation specifications. After the synthesis of these models and specifications, the implementation stage is executed again.

6. CONCLUSIONS

This work presented an implementation environment applied to reconfigurable processes in automated manufacturing systems. The insertion of the SCT, computational tools for synthesis and simulation, subsystems and specifications models libraries, brought a reduction of the manufacturing system's development time. The SCT is a formal approach that allows automatic synthesis of supervisors. Model libraries make the modeling step easier and allow models to be reused in subsequent projects. The integration of the simulation and implementation steps allowed a greater liability for validating, optimizing and accomplishing the CS.

Control structure simulation and the progressive implementation of control devices are also characterized as important tools for the methodology consolidation. These tools allow a time reduction in the global carrying out of the manufacturing automated and integrated system. In addition, by making use of a progressive implementation, the maintenance staff may scan and repair unexpected problems before and after the complete software and hardware's execution. This feature surely enhances the global project's reliance, and the maintenance staff's prompt response.

The proposed methodology still presents some limitations and this allows us to presume the continuity of the work that was developed. A drawback of the methodology application is associated with the progressive implementation of the control structure. Since this technique makes use of CCT, several problems related to integration and automation occur during implementation. The main cause of such problems is the adoption of proprietary systems (e.g. communication protocols, industrial networks, software, among others) by the existing manufacturers in the market. Traditionally, each equipment, device or software maker adopts a specific model or system. The drawback lies exactly in dealing with technologically heterogeneous systems, once the progressive establishment of communication with the various devices that control the plant is necessary.

REFERENCES

Bullock, D., B. Johnson, R. B. Wells, M. Kyte and Z. Li (2004). Hardware-in-the-loop simulation. *Transportation Research Part C*, 73-89, Elsevier, West Lafayette, IN, USA.

Carrol, J. and D. Long (1989). *Theory of finite automata*, Prentice-Hall, Inc, Upper Saddle River, NJ, USA.

Cassandras, C. G. and S. Lafortune (1999). *Introduction to discrete event systems*, Kluwer Academic Publishers, USA.

Elipse Software (2007), available in <http://www.elipse.com.br>, access in September 19th, 2007.

Erbe, H. (2002). Low Cost Intelligent Automation in Manufacturing. In: *Proceedings of the 15th Triennial IFAC World Congress*, Barcelona, Spain.

Hibino, H., T. Inukai and Y. Fukuda (2006). Efficient manufacturing system implementation based on combination between real and virtual factory. *International Journal of Production Research,* **44**, 18–19.

Holst, L., L. Randell and G. Bolmsjö (2000). Integrated Development of Manufacturing Systems Using Simulation – Proposing the Fundamentals for a Joint Research Project. In: *Proceedings of the 33rd CIRP International Seminar on Manufacturing Systems: The Manufacturing Systems in its Human Context – A Tool to Extend the Global Welfare,* 5-7, Stockholm, Sweden.

Holst, L., G. Bolmsjö, L. Randell and J. Norgren (2001). Integrating Simulation into Manufacturing System Development: A Methodological Framework. In: *Proceedings of the Twelth Annual Conference of the Production and Operations Management Society, POM-2001*, Orlando, USA.

IEC 61131-3 (1998). *International Electrotechnical Commission. Programmable Controllers. Programming Languages*.

ISO/TR10314-1 (1990). *Industrial Automation: Shop Floor Production: Part 1: Reference Model for Standardization and Methodology for Identification of Requirements*.

Lauzon, S. C., A. K. L. Ma, J.K Mills and B. Benhabib (1995). Application of Discrete-Event-System Theory to Flexible Manufacturing. In: *1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan.

Min, B., Z. Huang, Z. J. Pasek, D. Yip-Hoi, F. Husted and S. Marker (2002). Integration of Real-time Control Simulation to a Virtual Manufacturing Environment. *Journal od Advanced Manufacturing Systems,* **1,** 67-87.

Moore, P. R., J. Pu and H. C. Ng (2003). Virtual engineering: an integrated approach to agile manufacturing machinery design and control. *Mechatronics*, **13**, 1105–1121.

Queiroz, M. H. and J. E. R. Cury (2000). Modular Supervisory Control of Large Scale Discrete-Event Systems. In: *Proceedings of WODES 2000*.

Queiroz, M. H. and J. E. R. Cury (2002). Synthesis and implementation of local modular supervisory control for a manufacturing cell. *Discrete Event Systems: Analysis and Control*, 103-110, Kluwer Academic Publishers.

Ramadge, P.J., and W. M. Wonham (1989). The control of discrete event systems. In: *Proceedings of IEEE, Special Issue on Discret Event Dynamic Systems*, **77**, 81-98.

Rockwell Automation (2007), available in <http://www.arenasimulation.com>, access in September 19th, 2007.

Vieira, A. D., J. E. R. Cury and M. H. Queiroz (2006). A Model for PLC Implementation of Supervisory Control of Discrete Event Systems. In: *11th IEEE International Conference on Emerging Technologies and Factory Automation*, 225-232.