A method for design and valuation of manufacturing system control architecture

Bruno Denis, Jean-Jacques Lesage, Jean-Marc Roussel

A Method for Design and Valuation of Manufacturing System Control Architecture

Bruno DENIS, Jean-Jacques LESAGE and Jean-Marc ROUSSEL

e-mail: {denis, lesage, roussel}@lurpa.ens-cachan.fr
LURPA, ENS de Cachan, 61 av. du Pt Wilson, 94235 Cachan Cedex, France

In proceedings of 1995 IEEE International conference on systems, man and cybernetics
pp. 4486-4491, Vancouver, British Columbia, Canada, 22-25 October 1995
DOI: 10.1109/ICSMC.1995.538501

ABSTRACT

In the control design of complex manufacturing systems, the design of a control architecture is required. This paper deals with an integrated framework to make the design of control architecture as formal as possible. Models used in this framework are presented, and a valuation method of control architecture is constructed.

1. INTRODUCTION

The performance of manufacturing systems and that of their computerized control systems are closely connected. Furthermore, a higher automatization, a better reactivity, and a higher flexibility level are required for manufacturing systems. This positions the control system development at a strategic place both from technical and economical points of view. The focus of this paper is the development of the control architecture of manufacturing systems.

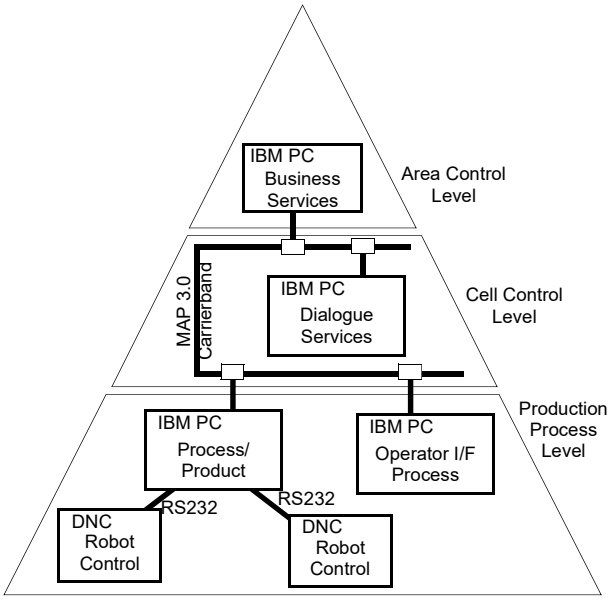

Fig.1 : example of the hardware point of view of control architecture (from [6])

This kind of systems includes component such as general purpose computers, industrial boards, Digital Controller System (DCS), and Programmable Logic Controller (PLC). A control architecture is often presented as in figure 1, but only the hardware point of view is described.
The design of a control architecture needs to be done early in the development of the whole system, precisely at the beginning of the control system life cycle. The designer therefore must:

- study and formalize requirements in details enough to propose a good control architecture so as to decide how appropriate the project is (to do that, the designer must sense a large part of the whole control system).
- study the control architecture briefly enough to prevent study costs before being sure that the project will be carried on.

Only engineer experience can bypass this paradox. With former similar study cases (i.e. expert knowledge), he is able to propose a solution accurate enough, after a short study. Nevertheless a computer aided architecture design and architecture evaluation could allow to:

- rationalize and homogenize the expert knowledge of each company
- perpetuate and share the built-up knowledge of previous studies
- increase quality of designed control architectures and succeed in managing higher complexity of manufacturing systems
- increase engineer reactivity against fast evolution of the component market as previous projects cannot help the design of control architecture which use new components
- increase productivity of the design activity of control architecture at the beginning of the project development cycle.

So the context of the design of control architecture is complex and submitted to continuous evolution. Models and method have to be proposed to handle complexity and to increase the quality of designed architectures. The next session presents our framework for the control architecture designer.

2. METHODOLOGY TO DESIGN CONTROL ARCHITECTURES

The proposed framework handles the design of control architecture from the general specifications of the system to the hardware architecture (fig. 2).

During the first stage of the method, components of system requirement are extracted to build the Implementation model. All the components which need to be implemented in a hardware equipment are concerned. This way, our framework is independent from requirement specification models (for example SA, SADT, OOA...)

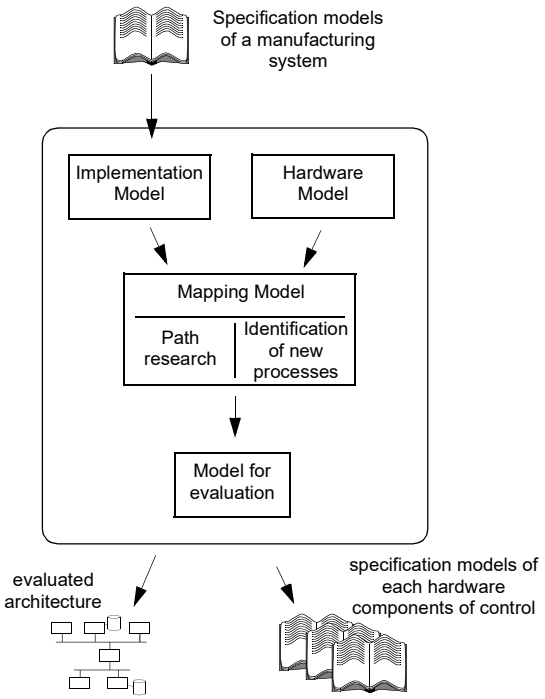

Fig.2 : framework to design control architecture

In the second stage, the designer has to propose an acceptable control architecture. This view of architecture is called the hardware model. A third stage consists in mapping implementation model components onto hardware model components. The result is an architecture solution called the mapping model. With the rigorous design of control architecture we propose, it is now possible to evaluate solutions and choice against checked and evaluated control architectures. Session 4 of this paper deals with models and methods to evaluate computerized control architecture of manufacturing systems.

3. MODELIZATION OF THE ARCHITECTURE OF CONTROL SYSTEMS

Implementation model

The aims of the implementation model are:

- to group and integrate all interesting data related to the control architecture design and that into a single model. For example, the function which appears in functional requirement of the control system has to be integrated to the functional model.
- to specify events which activate a control function. For example, functions appear in specification requirements as components receiving and producing data. The implementation model specifies which conjunctions of events can start the function i.e. "when is the function supposed to be computed?"
- to specify how long the data life is. Indeed, the life duration of a data could be limited to the duration necessary to carry it from its source to its target function. But the designer could also decide to keep data stored.

To build implementation models, the designer must reason as if the control architecture is a one single computer. The implementation model is a stage between control system specifications and the design of control architecture. In control system specifications there is no reference to any architecture, and the implementation model is built as if the final control architecture was a one single computer.

The implementation model must specify all processes which are required to compute control functions, stores, data flows and synchronization rules. The chosen language is based on the data flow diagram for two reasons: it is a well-known language to model control systems [3], and numerous works have proposed extensions [2] [7] [10].
The implementation model is a network of processes, linked with data flows. The model is built upon four components: processes, stores, flows, and externals.

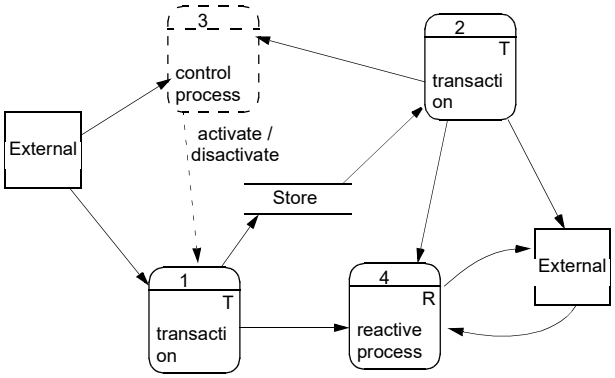

Fig.3 : example of implementation model

In the implementation model, three kinds of processes appear as regards the way they can be computed.

- transformation processes (T): a finite sequence of instructions to transform input data into output data,
- reactive processes (R): a finite state machine to transform input events into output events according to its own current state,
- control processes (dashed lines), which activate and disactivate control transformation processes and reactive processes.

Data stores can take all kinds and sizes of data into account. From the point of view of processes, stores are servers offering read and write data services.

Externals are processes which are not inside the designed control system, but which are source and/or destination of data flows.
Data flows allow processes to exchange data. Three kinds of flows are used in the implementation model according to the nature of flowing data.

- data flows are oriented links between processes and flows. They show how processes read and write data.
- message flows transport both events and data from process to process. By sending a message, processes are able to trigger off other processes (with event component of messages) and send data.
- control flows are produced by control processes to activate regular processes such as transformation and reactive processes.

On this data flow model, an extended semantic specifies how processes are activated. The aims of these extensions are to show conditions required for processes to be computed (i.e. conditions to send a CPU time request), and data and message flows sent after current activation conditions.

The behavior of transformation processes such as in figure 4 is described as follows. The process runs when an event conjunction appears from messages. When the process is finished, a set of messages (called disjunction) is sent out. Stores do not participate to process synchronization.

Nevertheless processes can read data in stores just after being activated by a message conjunction, and write in stores just before sending a message disjunction. The couple $(C(p), D(p))$ completes the description of transformation processes with:

- $C(p)$ set of conjunctions associated with the p process
- $D(p)$ set of disjunctions associated with the p process

Where $\forall c \in C(p)$, c is the couple $(M(c), S(c))$ such as:

- $M(c)$ set of messages of conjunction d required to start the process p,
- $S(p)$ set of read stores after the disjunction d at the beginning of process p,

And where $\forall d \in D(p)$, d is the couple $(S(d), M(d))$ such as:

- $S(d)$ set of written stores before the disjunction d at the end of process p,
- $M(d)$ set of sent messages of disjunction d at the end of process p,

A transformation process verifies the following properties:

- the p process is activated when the boolean equation (1) is true, i.e. when there exists a conjunction such as all its messages are present

$$\sum_{c_i \in C(p)} \left(\prod_{m_j \in M(c_i)} m_j \right) = 1 \qquad (1)$$

- $\forall d \in D(p) |\ d=(S(d), M(d))$, when disjunction d appears, one datum is sent in each store in $S(d)$, and then all messages $M(d)$ in the disjunction are sent.

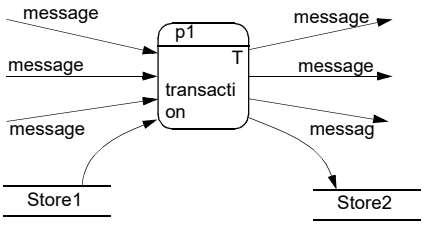

Fig.4 : example of graphic view of transformation process

figure 4 shows an example for transformation processes. It is activated when

$$(message_1 \cdot message_2) + (message_2 \cdot message_3) + (message_1 \cdot message_3) = 1 \qquad (2)$$

Once the computation of the transformation process is done, two disjunctions can occur:

- either writing data in $store_2$ and sending $message_4$ and $message_5$,
- or sending $message_5$ and $message_6$.

In other words, the graphic view shown in figure 4 must be completed with the couple

$$(C(p1), D(p1)) = (\{c1, c2, c3\}, \{d1, d2\}) \qquad (3)$$

where c_i and c_j could be as follows:

- $c1 = (\{message1, message2\}, \{store1\})$
- $c2 = (\{message2, message3\}, \varnothing)$
- $c3 = (\{message3, message1\}, \{store1\})$
- $d1 = (\{store2\}, \{message4, message5\})$
- $d2 = (\varnothing, \{message5, message6\})$

To summarize, to build the implementation model of a manufacturing system is to translate and enrich its specifications. If the system specifications exist, this task can be formalized and automated [4]

Hardware model

The aim of the hardware model is to show the physical point of view of the control architecture design [6]. It is often the only point of view used to design control architectures.

The hardware model is a network of processor, store system, man-machine interfaces, and communication links.

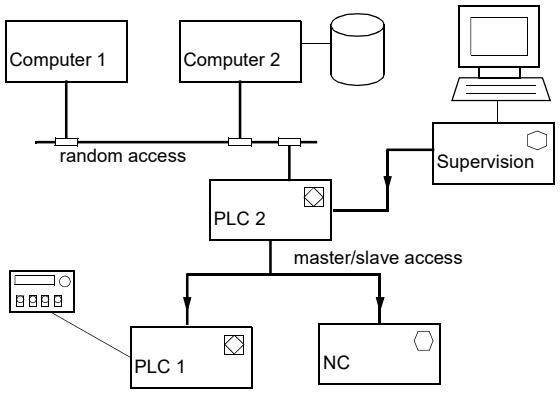

Fig.5 : example of hardware model

We want our proposed framework to be able to tolerate technology evolutions of hardware systems. Hence, entities of hardware models are not market components but classes of components.

The set of classes which is proposed here, is not comprehensive. there are just classes implemented in our experimental CASE tool. There can easily accept new classes for new purposes.

Among processor classes there are:

- Programmable logic controllers (PLC),
- Numeric controllers (NC),
- computers such as embedded boards, supervision computers and general purpose computers.

Among man-machine interfaces classes there are:

- video control panel,
- keypad and message displays,
- push-button panels.

Among communication links classes there are:

- point-to-point links which are divided into processor links (master/slave or random access), and multipole link such as push-button panel to PLC or between two PLC,
- networks which are divided into field busses, master/ slave access networks (for instance PLC network) and random access network.

Mapping model

The aims of the mapping model are to allocate implementation model components to the hardware model. To be computed, processes will be allocated to the processor, stores will be allocated to memories or to data bases, and data flows will allocated to communications links.

We propose a two stage method to build the mapping model. The first stage consists in mapping each process into a processor. The second stage consists in finding a path in the hardware model for data flows. An aid is proposed to designers to find data flow paths inside of the hardware. The hardware model is transformed into a graph, and this new problem is actually to find a path in the associated graph

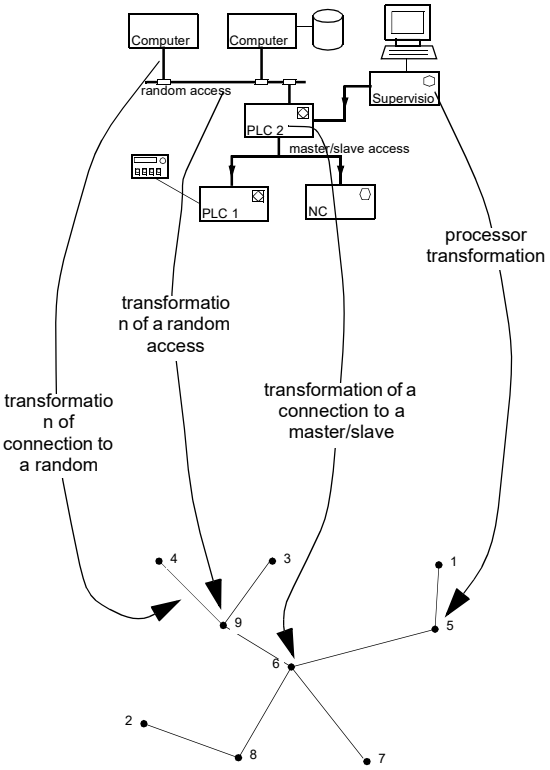

Fig.6 : example of transformation of the hardware model into graph

The hardware model transformation is as follows:

- each processor and man-machine interface are associated to a graph node,
- each random access network is associated to a graph node,
- for all point to point connections between two processors (or between a processor and man-machine interfaces) an arc between related nodes is drawn.
- for all master/slave access networks, links are drawn between nodes relating to the master processor and nodes relating to slaves,
- for all random access networks, arcs are drawn between the network node and each equipment node.

An example of graph construction is presented on figure 6. To find a path for data flows requires two steps. The first one is to identify the two nodes relating to the source and destination processors of data flows and the second is to search a path between two nodes of a graph. If several paths exist, the designer can weight arcs (for example transfer speed weighting); then the aid gives the shortest path in the graph, i.e. the best data flow path from the designer's point of view.

To transport data across several processors of control architecture a new kind of process is required: technical processes. They do not come from the implementation model as they are not functional, their existence only comes from a particular hardware structure. Though they are often ignored by designers during the specification stage of control architecture development, technical processes have a heavy influence on the control system performance. We have proposed a systematic approach to identify technical processes, based on graph paths and reference models [5]. A path is a succession of triplets such as source of flow (processes or stores), communication links (random, master/ slave, or slave/master) and destination (processes or stores).

Nine different reference models have been associated to each triplet such as a reference model using pooling processes (technical processes) to transmit data from the slave side to the master side of a communication link.
The study of technical processes is important as it is the stage where the designer discovers the consequences of his choice of hardware architecture.

When the mapping model is done, the designer has a complete view of the control architecture. A mapping model is a solution of control architecture of the manufacturing system.
In the next session we propose a performance analysis of control architectures.

4. EVALUATION OF CONTROL ARCHITECTURES

To get information about performance criteria such as dynamic response time, a dynamic model must be used. Timed and Colored Petri Nets (TCPN) have been chosen as they can model manufacturing systems in a compact way (color aspect) and allow us to simulate and validate time performances (time aspect).
Several works have already proposed extensions to model the dynamic behavior of system specifications [10] [7] [11], and others proposed to model the dynamic behavior of the hardware [1] [9].
Our assessable model is original for it is built according to the mapping model. So it takes the functional point of view (processes, stores…) the hardware point of view (computer, PLC…) and also the technical processes into account.
The proposed assessable model describes the dynamic behavior of the implementation model mapped into the hardware model.
The proposed method is as follows:

- each type of components of the implementation model (processes and stores) is described by a generic TCPN (fig. 7). Each type of generic TCPN proposed admits two different types of Input/Output. There are I/O relating to the implementation model (message conjunctions, message disjunction, data read and data write), and there are I/O relating to the mapping model (requests of CPU resources).
- each hardware equipment such as processors and communication links is modeled by a generic TCPN too. This model describes resources ready to answer process and store requests
- then the whole assessable model is built by assembling all previous generic TCPN.

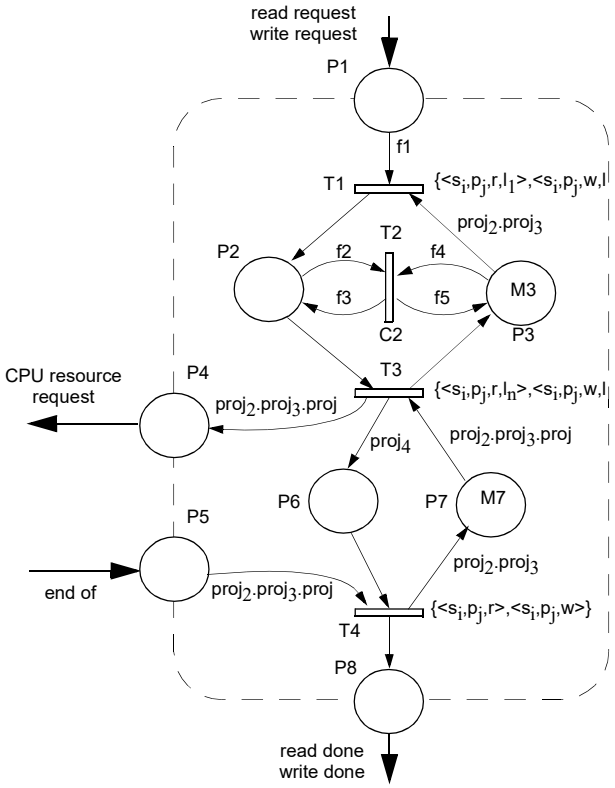

Fig.7 : generic TCPN for the behavior of store components

Generic TCPN models such as in figure 7 are not supposed to be detailed views of hardware equipment or components of the implementation model. The aim of these TCPN models is to show when processes and stores obtain the hardware resources they require. However, more detailed models are provided for example in [9] for networks and in [8] for computer.

The proposed dynamic model of stores is presented figure 7 and table 1. It is model of all $s_i \in S$, where S is the set of store from the implementation model. The model is organized around places P6 and P7. Place P7 represents idle stores, and P6 busy ones. Places P1, P2, P3 and transition T1,T2, T3 model a FIFO stack including $n(s_i)$ location. As soon as a request get out from stack (firing T3 transition), a CPU resource request is sent. Color $\langle s_i, p_j, r, l_k \rangle$ means read request (r) of si store from p_j process, and this request is in the l_k location in the stack.

Tab. 1 : description of behavior model of stores

	Description
f1	$f_1(\langle s_i, p_j, r, l_1 \rangle) = \langle p_j, s_i, r \rangle$ $f_1(\langle s_i, p_j, w, l_1 \rangle) = \langle p_j, s_i, w \rangle$
f2	identity
f3	$f_3(\langle s_i, p_j, w, l_k \rangle) = \langle p_j, s_i, w, l_{k+1} \rangle$ $f_3(\langle s_i, p_j, r, l_k \rangle) = \langle p_j, s_i, r, l_{k+1} \rangle$
f4	$f_4(\langle s_i, p_j, r, l_k \rangle) = \langle s_i, l_{k+1} \rangle$ $f_4(\langle s_i, p_j, w, l_k \rangle) = \langle s_i, l_{k+1} \rangle$
f5	$f_5(\langle s_i, p_j, r, l_k \rangle) = \langle s_i, l_k \rangle$ $f_5(\langle s_i, p_j, w, l_k \rangle) = \langle s_i, l_k \rangle$
C2	$\{\langle s_i, p_j, r, l_k \rangle, \langle s_i, p_j, w, l_k \rangle, k \in [1, n(s_i)[\}$

Tab. 1 : description of behavior model of stores

	Description
M3	$\sum_{s_i \in S} \left(\sum_{j=1}^{n(s_i)} \langle s_i, l_k \rangle \right)$
M7	$\sum_{s_i \in S} \langle s_i \rangle$

The proposed TCPN models have been design as generic as possible. They set up a dynamic models library which can be used and enriched to build a whole valuable model of control system.

A model of a particular architecture is obtained with a token mailing from generic TCPN models to others. Figure 8 shows arcs and transitions which model the generic aspect of the mailing.

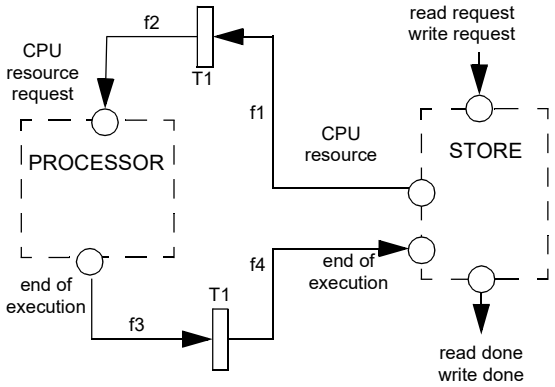

Fig.8 : extract of global mailing model for CPU resource requests

5. CONCLUSIONS

In the paper we propose a formalization of the design of control architectures. We show the designer's main activities and we propose a framework to design control architectures.

The interest of the formalization of control architecture is to allow to build a behavior model systematically, and to get a performance analysis. A dynamic model is presented in this paper. Our current work is on the stochastic model to study the reliability point of view of control architectures.

6. REFERENCES

[1] M. Ajmone and G. Chiola, "Construction of generalized stochastic petri net models of bus oriented multiprocessor systems by stepwise refinements: a case study" in *Proceedings of the International conference modelling techniques and tools for performance analysis*, (Sophia Antipolis, France), pp. 265-278, June 1985.

[2] W. Bruyn, R. Jensen, D. Keskar, and P. Ward, "Esml: an extended systems modeling language based on the data flow diagram"*ACM software engineering notes*, vol. 13, no. 1, pp. 1-19, 1988.

[3] T. Demarco, *Structured analysis and system specification*. YOURDON PRESS, englewood cliffs ed., 1979.

[4] B. Denis, J.-J. Lesage, and G. Timon, "Toward a theory of integrated modelling" *Journal of Design Science and Technology*, vol. 2, no. 1, pp. 87-96, 1993.

[5] B. Denis, *Assistance à la conception et à l'évaluation de l'architecture de conduite des systèmes de production complexes*. PhD thesis, University of Nancy I, 1994.

[6] M. Didic, "Cimosa model creation and execution for a casting process and a manufacturing cell" *Computers in Industry*, vol. 24, pp. 237-247, September 1994.

[7] R. France, "Semantically extended data flow diagrams: a formal specification tool" *IEEE transaction on software engineering*, vol. 18, no. 4, pp. 329-346, 1992.

[8] R. Lepold, *Performability evaluation of degradable computer systems based on stochastic Petri nets*. PhD thesis, University of Haute Alsace, France, 1992.

[9] A. D. Stephano, O. Mirabella, and C. Zappalà, "Featuring fddi in a process control environment" *Computer in industry*, vol. 21, no. 1, pp. 35-49, 1993.

[10] P. Ward, "The transformation schema: an extension of the data flow diagram to represent control and timing" *IEEE transaction on software engineering*, vol. 12, no. 2, pp. 198-210, 1986.

[11] J. Zaytoon, E. Neil, A. Mille, and A. Jutard, "A temporal sadt for automated manufacturing systems" in *International conference on industrial engineering and production management*, vol. 1, (Mons, Belgium), pp. 154-164, June 1993.