A Coordination Structure for a class of Intelligent Machines

M.N.VARVATSOULAKIS
Electrical & Computer Engineering Dept.
National Technical University of Athens
Zographou 15773-Athens
GREECE

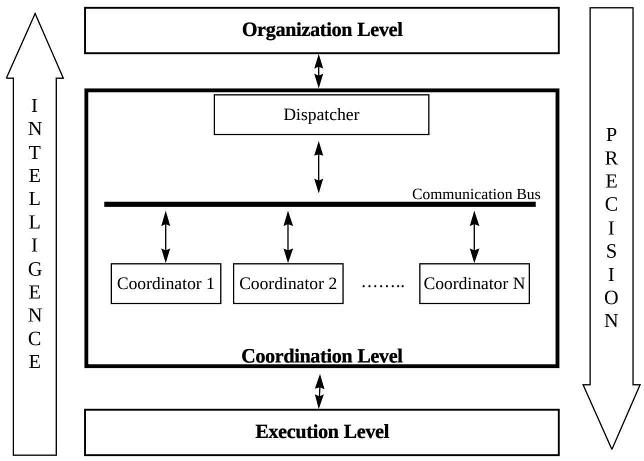
G.N.SARIDIS
ECSE Department
Rensselaer Polytechnic Institute
Troy, NY 12180-3590
USA

P.N.PARASKEVOPOULOS
Electrical & Computer Engineering Dept.
National Technical University of Athens
Zographou 15773-Athens
GREECE
CSCC'99 Proc.pp.2181-2186

***Abstract* - In this paper we present a model for the Coordination level of a class of Intelligent Machines suitable for industrial applications. The Coordination level is intended to translate a high-level command into a schedule of low-level primitive activities. The purpose of this model, based on the theory of Hierarchically Intelligent Control Systems developed by Saridis, is to specify the integration of the individual efforts on task translation, task coordination and task signaling of cooperating systems that combine to form an intelligent machine.**

1. Introduction

The theory of intelligent machines has been recently reformulated to incorporate new architectures that are using state machines to represent processes completed in a fixed number of sequential steps. Intelligent machines are based on the Principle of Decreasing Precision with Increasing Intelligence to form an analytic methodology, using Entropy as a measure of performance. The original architecture proposed by Saridis represents a three level system, according to the principle, including Organization level, Coordination level and Execution level [Fig. 1].



[Fig. 1 Intelligent Machine Representation]

The Organization level is intended to perform such operations as planning and high level decision-making and may require large quantities of information processing but little or no precision.

The Coordination level is an intermediate structure between the organization level and the execution level with functions dominated by discrete event control that dispatch commands to devices and coordinate their activities.

The Execution level consists of devices with high requirements in precision having functions dominated by classical control theory.

This hierarchical approach implies that the Organization level represents abstract activities and it is evident that the structure of the Coordination level dispatcher designed to interpret the Organizer strings and allocate commands among the coordinators is highly dependent on the natural language representing the sequence of the planned tasks.

In the present work [18] a new structure for the Coordination level is proposed, mainly oriented to manufacturing applications. The model requires the following capabilities:

- Task representation and processing ability which classifies the tasks in an hierarchical manner by defining initial, intermediate and final ones and in the sequel selects the appropriate control procedures to send as commands to the execution level.
- Learning ability which taking into account the performance indices from the execution level improves the task sequence selection by reducing uncertainties in decision-making as more experience is obtained.

Previous research for the Coordination level has been presented in [13], [14], [15], [16]. In [13] Saridis and Graham propose the use of linguistic decision schemata for the translation of a given input string to a set of strings representing operating commands for hardware devices. In [14], [15], [16] Wang extends this idea to Petri-Net Transducers. These formalisms are intended rather to produce commands, combining an input language with output languages.

In the present work the finite state machine has been selected as the basic construction module for the proposed model, mainly oriented to produce appropriate signals that are shared among different subsystems and coordinate their operation.

The discrete event formalism is stated in Section 2. In Section 3 the analytic model is established. Section 4 presents the learning methods. Section 5 applies the theory presenting a model suitable for industrial applications. Finally, Section 6 summarizes the work and presents its major conclusions.

2. Preliminaries and Terminology

The Coordination level of intelligent machines serves as an interface between the Organization level and the Execution level and dispatches organizational tasks to execution devices. An analytical model for this level should comprise :

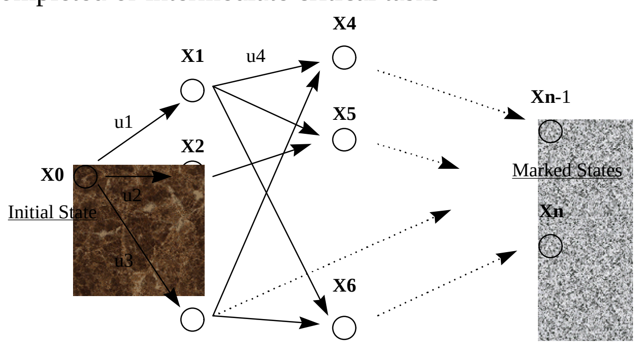
-Formal definition of the subsystems and representation of the processes within each one.
-Formal translation of the tasks issued by the Organization level.
-Communication mechanisms between individual subsystems and coordination of their activities.

The Finite State Machine Generator is the basic structure for the proposed model, being appropriate for the representation of tasks completed in a fixed number of sequential steps. It can also serve all the previous requirements and provide us a hierarchical, modular and stepwise design approach.

The class of systems we consider is an extension of non deterministic finite-state machine generators following the framework of Ramadge-Wonham [8], [9]. A finite-state machine generator represents a discrete event dynamic system that responds to generated spontaneous events producing internal state transitions and output symbols.

Definition 2.1. FSMG = $(X, U, Y, f, g, X_0, X_f)$ is a finite-state machine generator [Fig. 2] where

X is the finite state set
U is the events alphabet
Y is the output alphabet
$f : X \times U \rightarrow 2^X$ is the transition function
$X_0 \in X$ is the initial state
$X_f \subseteq X$ is the set of marked states representing completed or intermediate critical tasks



[Fig. 2 Finite State Machine Generator]

The internal state transition can be achieved in either a deterministic or a non deterministic fashion. In the deterministic approach only one next state is defined after the occurrence of one event. In the non deterministic approach a fixed number of states are prespecified and the selection is based on the current status of the process.

Letting f be extended to a function $f^* : U^* \times X \rightarrow X$ the internal behavior of FSMG is described in terms of the formal regular languages:

$L_X(FSMG) \subseteq U^* := \{u^* \in U^* \mid f^*(u^*, X_0) \text{ is defined}\}$

i.e. the set of all finite traces of events accepted by FSMG.

$L_{Xf}(FSMG) \subseteq L_X(FSMG):=\{u^* \in L_X(FSMG) \mid f^*(u^*,X_0) \in X_f\}$ i.e. the set of all finite traces representing completed tasks by FSMG.

In the linguistic decision approach, FSMG accepts commands that consist of strings belonging to $L_X(FSMG)$ or $L_{Xf}(FSMG)$.

Let U* denote the set of all finite strings over U including the empty string e. In this way let

$h : U^* \rightarrow 2^Y$ be the output function

The behavior of the FSMG is stated as:

$$X(k+1) \in f(X(k), U(k))$$
$$Y(k+1) = h(U^*(k)) \text{ if } X(k+1) \in X_f$$

Here $X(k+1) \in 2^X$ is the state after the k event, $U(k) \in U$ is the k event, $U^*(k)$ the sequence $U(k)U(k\text{-}1)\ldots U(0)$ of events at the instance k and $Y(k+1) \in 2^Y$ is the produced output symbols set when the string $U^*(k)$ processed by FSMG represents a complete task.

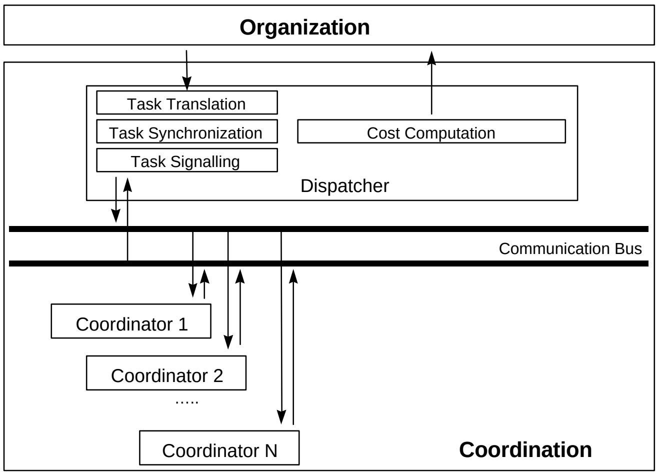
3. The analytic model

The Coordination level of the proposed model is composed of one dispatcher, a fixed number of coordinators and a communication bus for the exchange of signals among them. In this coordination structure the dispatcher occupies a dominant position in the connection configuration. Each coordinator transmits and receive signals via the communication bus and there is no direct connection between individual coordinators. In this way the dispatcher serves as both a task control center and an information communication center.

The dispatcher receives the task commands from the Organization level in the form of strings of finite length and in the sequel signals the appropriate coordinators. The functions of the dispatcher are defined to be task translation, task coordination and task signaling.

The coordinators correspond to the subsystems dedicated for the accomplishment of each process and represent deterministic or non-deterministic task sequences. In many instances of prespecified processes it is necessary to enforce particular sequences that need two or more coordinators synchronize their activities.

The dispatcher, modeled by a FSMG, accepts as input the sequences of events from the Organizer and when this sequence is complete, generates the output symbols for the corresponding coordinators and sends them to the communication bus [Fig.3]. Every output symbol in the communication bus addresses only one coordinator i.e. accessed by all coordinators but processed by the corresponding coordinator.



[Fig. 3 Coordination Structure]

The coordinators, modeled also by a FSMG, accept as input the appropriate symbols in the communication bus and start the task execution. In the sequel they follow internal modeled transitions and, when they accomplish predefined intermediate or final tasks, generate output addressed symbols to the communication bus. The output symbols may i)inform the dispatcher for the completion of a task ii)signal another coordinator to start or continue a task execution.

Every intermediate task in the coordinator is translated to appropriate operating instruction required by the appropriate execution devices in the Execution level. The process of task translation is continued until the job issued by the Organization level is completed.

Note that the coordinators have to cooperate under the supervision of the dispatcher in the sense that none of them has sufficient ability and information to accomplish the entire task. Mutual sharing of information is necessary to allow the dispatcher and the coordinators, as a whole, to attack the requested job.

4. Learning

The task evolution when the transitions are deterministic is achieved based on a combination of event symbol and previous state. When the transitions are non-deterministic a fixed number of options for a task are prespecified and the objective is to select the optimal one according to a measured performance index. We define the following learning schemata based on special problem complexity:

Single-level Schema : Let m the number of options u_i, i=1,…, m and p_i the subjective probabilities. The decision making in the probabilistic model proceeds as follows :

-A random performance index is associated with each option u_i. After the execution of the action update the performance estimate using the formula :

$$J_i(k+1) = J_i(k) + \frac{1}{k+1}\left(J_{MRi}(k) - J_i(k) \right)$$

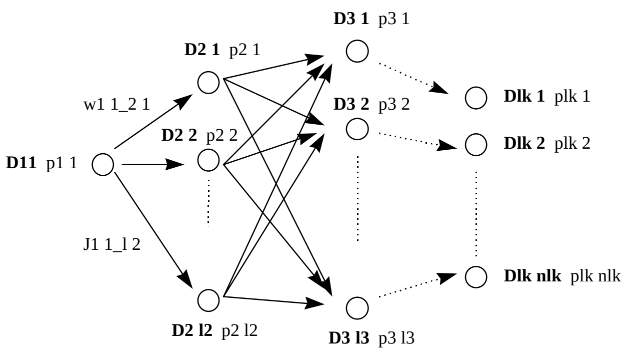
where $J_{MRi}(k)$ is the k measured performance index and $J_i(k)$ the k performance index estimation.

-Next update the subjective probabilities by the formula :

$$p_i(k+1) = p_i(k) + \frac{1}{k+1}\left(p - p_i(k) \right)$$

where p=1 if $J_i(k) = \min J_i(k)$,i=1,…,m and p=0 otherwise.

Multi-level Schema : If the non deterministic options are represented as a Boltzman Machine of [Fig. 4] the learning can proceed as follows :



[Fig. 4 Boltzman Machine Representation]

A probability of activation is assigned to each node, a weight for the transfer between nodes and the entropy associated with it is calculated. In order to calculate the total entropy of connections the following elements are defined:

-The ordered set of levels L = (l_i), i=1,..,k is the set of abstract primitive tasks of the machine and each one contains a number nl_i, i=1,..,k of independent primitive nodes.

-The set of nodes D = {d_{11}, d_{12}, …, $d_{l_i nl_i}$ }, i=1,..,k is the task domain of the machine and each node represents an individual task unit.

-The set Q = {q_{11}, q_{12}, …, $q_{l_i nl_i}$ }, i=1,..,k represent the state of events associated with each node D. The random variable q is binary (0,1) and indicates whether an event is inactive or active in a particular task.

-The set of probabilities P = { p_{11}, p_{12}, …, $p_{l_i nl_i}$ }, i=1,..,k associated with the random variables q is defined as follows:

P = { p_{ij} = Pr (q_{ij} = 1), i = 1, .., l_k , j = 1, .., nl_i }

-The set of weights W = { $w_{i\,j\,-\,i+1\,k}$ }, i=1,..,ln-1, j=1, .., nl_i , k = 1, .., nl_{i+1} associated with the interconnections between nodes $d_{i\,j}$, $d_{i+1\,k}$.

The probabilities and the weights are defined at the beginning of the learning stage according to previous experience.

The negative entropy in Shannon's sense of transfer d_{im} to d_{i+1j} is calculated in [17] as :

$$H_{d_{im}d_{i+1j}} = -E[\ln p_{d_{im}d_{i+1j}}] = \bar{a}_{d_{im}} + \frac{1}{2} w_{d_{im}d_{i+1j}}\, p_{d_{im}}\, p_{d_{i+1j}}$$

with $\bar{a}_{d_{im}} = E[a_{d_{im}}] = \ln \sum_{j=1}^{nl_{i+1}} \exp(-\frac{1}{2} w_{d_{im}d_{i+1j}} p_{d_{im}} p_{d_{i+1j}})$

The decision making is obtained by selecting the total maximum negative entropy at every transition which gives the optimum sequence of nodes to be selected. If we define as S(f), f = 1,..,k the array containing the selected nodes from each level the total maximum entropy of knowledge flow after n ≤ k nodes is

$$H^*(n) = \max_{S(f),f=1,..,n} \sum_{i=1}^{n} (\bar{a}_{d_{iS(f)}} + \frac{1}{2} w_{d_{iS(f)}d_{i+1S(f+1)}}\, p_{d_{iS(f)}}\, p_{d_{i+1S(f+1)}})$$

The learning is obtained by feedback devices that upgrade the probabilities and the weights by evaluating the performance of the lower levels after each iteration. The stochastic approximation reinforcement learning scheme used in this work is an extension of the algorithm proposed by Nicolic and Fu [5].

For every transition between nodes the performance index

$J_{d_{iS(f)}d_{i+1S(f+1)}}(t)$ is estimated according to:

$$J_{d_{iS(f)}d_{i+1S(f+1)}}(t+1) = J_{d_{iS(f)}d_{i+1S(f+1)}}(t) + \frac{1}{t+1}\left(J_{MRd_{iS(f)}d_{i+1S(f+1)}}(t+1) - J_{d_{iS(f)}d_{i+1S(f+1)}}(t) \right)$$

where $J_{MRd_{iS(f)}d_{i+1S(f+1)}}(t)$ is the t measured value of $J_{d_{iS(f)}d_{i+1S(f+1)}}$ and $\lim_{t \to \infty} E[J_{MR\,d_{iS(f)i+1S(f+1)}}(t)] = J_{S\,d_{iS(f)i+1S(f+1)}}$.

To update the probabilities and the weights Fu's stochastic approximation reinforcement learning scheme is also used:

$$p_{diS(f)}(t+1) = p_{diS(f)}(t) + \frac{1}{t+1}\left(p - p_{diS(f)}(t) \right)$$

$p_{d_{i+1S(f+1)}}(t+1) = p_{d_{i+1S(f+1)}}(t)$
$+ \frac{1}{t+1}\,(p - p_{d_{i+1S(f+1)}}(t))$

$w_{d_{iS(f)}d_{i+1S(f+1)}}(t+1) = w_{d_{iS(f)}d_{i+1S(f+1)}}(t) +$
$\frac{1}{t+1}\,(w - w_{d_{iS(f)}d_{i+1S(f+1)}}(t))$

where

$$p, w = \begin{Bmatrix} 1 \text{ if } \sum_{i=1,f=1}^{n} J_{d_{iS(f)}d_{i+1S(f+1)}}(t) = \min \sum J \\ 0 \text{ if } \sum_{i=1,f=1}^{n} J_{d_{iS(f)}d_{i+1S(f+1)}}(t) \neq \min \sum J \end{Bmatrix}$$

A Learning Index is defined in order to track convergence of the algorithm during execution. The maximum entropy of the selected sequence is obtained when all probabilities and weights of its nodes are equal to 1 and all the others are equal to 0. In this way

$H_{max}(n) = \frac{n-1}{2} + \ln[(e^{-1} + nl_2 - 1)(e^{-1} + nl_3 - 1)...(e^{-1} + nl_n - 1)]$

The Learning Index (LI) is defined as $LI = \frac{H(n)}{H_{max}(n)}$ and according to Learning $\lim_{t \to \infty} LI = 1$ when the noise has been eliminated by the cost estimates and the optimum sequence is selected for a long number of continuous iterations.

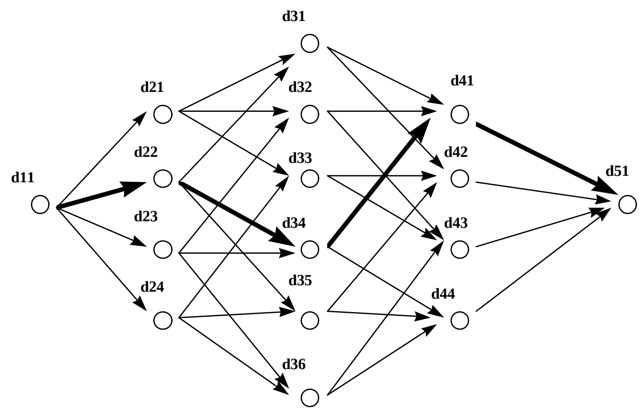
5. Application Study

The operation and control of large scale flexible manufacturing systems is usually a difficult task because it involves several control alternatives in order to succeed an optimal policy. Such systems are very complex and the efficiency of the dynamic models which describe them is limited by the enormity of their dimensions.

This application study involves a typical automated scheduling process represented by a multi-level network [Fig.5]. We have L = (l_1, l_2, l_3, l_4, l_5) and nl_1= 1, nl_2 =4, nl_3=6, nl_4=4, nl_5=1.

A. The Organization level

The Organization level produces strings of symbols from the alphabet S={d_{ij} , i=1,...,5, j=1,…,nl_i} corresponding to alternative operating plans. They are generated by the grammar G=(N,S,P) where N={N_i , i=1,...,5} the set of non terminal symbols, S the set of terminal symbols and P the set of production rules generated by the recursive form $P_1 \to d_{11}$, $P_{i+1} \to P_i d_{i+1\,j}$, i=1,...,4, j=1,..., ,$nl_{i+1}$.



[Fig. 5 Multi-Level Network]

B. The Dispatcher

The Dispatcher in the Coordination level is modeled by a FSMG=(X_D,U_D,Y_D,f_D,g_D,X_{0D},X_{fD}) where $U_D \supseteq S$ i.e. inputs strings from the Organization level. When the system reaches a state $X_D \in X_{fD}$ the corresponding output symbols for the coordinators are produced. After the completion of a task the coordinator outputs symbols $y \in U_D$ to inform the dispatcher. In this way the model incorporates the logic of the Organization level as with as additional low-level operating constraints between subsystems. All the state transitions are deterministic and no learning mechanism is defined.

C. The Coordinators

The Coordinators correspond to the levels of the Organizer and each one represents alternative lower level plans of the same task. They are modeled by $FSMG_{Ci}$=(X_{Ci},U_{Ci},Y_{Ci},f_{Ci},g_{Ci},X_{0Ci},X_{fCi}) where $Y_D = \bigcup_{i=1,...5} U_{Ci}$ i.e. input strings from the Dispatcher. When the system reaches a state $X_{Ci} \in X_{fCi}$ the corresponding output symbols for the Dispatcher are produced i.e. $Y_{Ci} \in U_D$. In this way the model incorporates the logic for the cooperation of different coordinators as with as additional internal operating tasks in each subsystem.

Learning mechanisms are provided if the transitions are non deterministic i.e. an optimal plan must be selected according to observed uncertain data. If the number of alternating plans is comparatively low the selection is based on a Single-level Schema. If the complexity of selection is represented as a multi-level network where dynamic reconfiguration may be required [17] the Multi-level Schema is more appropriate.

6. Conclusions

The architecture described in this paper is based on the architecture proposed by Saridis [2], [3], [14], [15], [16], [17]. The details have been more specified and more efficient internal structures have been used. This coordination structure provides an analytical mechanism of control and communication for autonomous intelligent control systems in various fields of modern industry. The task representation provides the base for designing the task scheduling procedure and the learning algorithm gives an adaptive approach for finding the optimal operating schedules when the environment is uncertain. This approach is more suitable to industrial applications with fixed number of operating plans. The main contribution is that this structure is well suited to many existing industrial applications and is extremely effective as compared to other architectures.

On-going research is trying to evaluate more complicated structures and establish a unified approach in dealing with discrete event control problems. Based upon modern automated processes, like flexible manufacturing systems, robotics and other advanced automation systems simulation studies should test the validity of the obtained results.

Acknowledgement :
This work is financially supported by the Greek General Secretariat for Research and Technology (Ministry of Development) and the Greek State Scholarships Foundation.

References :

[1] M.C.Moed and G.N.Saridis, ”A Boltzman machine for the organization of intelligent machines”, *IEEE Trans. SystemsManCybernetics,* vol.20, no.5, 1990, pp.1094-1102.

[2]G.N.Saridis and K.P.Valavanis, ”Analytical Design of Intelligent Machines”, *Automatica,* vol.24, no 2, 1988, pp.123-133.

[3]G.N.Saridis ”Analytic Formulation of the Principle of Increasing Precision with Decreasing Intelligence for Intelligent Machines”, *Automatica,* vol.25, no 3, 1989, pp.461-467.

[4]H.Lewis and C.H.Papadimitriou, ”Elements of the theory of computation”, Prentice-Hall Inc., 1981.

[5] Z.J.Nicolic and K.S.Fu, ”An Algorithm for Learning without External Supervision and its application to Learning Control Systems”, *IEEE Trans.Automatic Control,* vol.11, no.3, 1966, pp.414-422.

[6]K.S.Fu and Z.J.Nicolic, ”On some reinforcement techniques and their relation to the stochastic approximation”, *IEEE Trans.Automatic Control,* vol.11, no.2, 1966, pp.756-758.

[7]Pedro U.Lima and G.N.Saridis, ”Design of Intelligent Control Systems based on Hierarchical Stochastic Automata”, World Scientific, 1996.

[8]P.J.Ramadge and W.M.Wonham, ”Supervisory control of a class of discrete event processes”, *SIAM J.Contr.Optimization,* vol.25, no.1, 1987, pp.206-230.

[9]W.M.Wonham and P.J.Ramadge, ”Modular supervisory control of discrete event systems”, *Mathematics of Control, Signals and Systems,* vol.1, no.1, 1988, pp.13-30.

[10]F.Lin and W.M.Wonham, ”Decentralized supervisory control of discrete event systems”, *Information Sciences,* vol.44, 1988, pp.199-224.

[11]R.David and H.Alla, ”Petri Nets for Modeling of Dynamic Systems-A Survey”, *Automatica,* vol.30, no.2, 1994, pp.175-202.

[12]Jaynes E.T. ”Information Theory and Statistical Mechanics”, *Physical Review,* 4, 1957, pp.106.

[13]G.N.Saridis and J.H.Graham ”Linguistic Decision Schemata for Intelligent Robots”, *Automatica,* vol.20, no 1, 1984, pp.121-126.

[14]F.Y.Wang and G.N.Saridis ”A Coordination Theory for Intelligent Machines”, *Automatica,* vol.26, no 5, 1990, pp.833-844.

[15]F.Y.Wang and G.N.Saridis ”Task Translation and Integration Specification in Intelligent Machines”, *IEEE Trans. Robotics and Automation,* vol.9, no.3, 1993, pp.257-271.

[16]F.Y.Wang, K.Kyriakopoulos, A.Tsolkas and G.N.Saridis ”A Petri-Net Coordination Model for an Intelligent Mobile Robot”, *IEEE Trans. Systems Man Cybernetics,* vol.21, no.4, 1991, pp.777-789.

[17]M.Varvatsoulakis, G.Saridis, P.Paraskevopoulos , ”A Model for the Organization Level of Intelligent Machines’’, *IEEE International Conference on Robotics and Automation, Leuven:Belgium,* no 187, 1998.

[18]M.N.Varvatsoulakis ‘’Learning Control of Discrete Event Processes’’, Ph.D. thesis, National Technical University of Athens, *i*n preparation.