USE OF ADAPTIVE COLOURED PETRI NETWORK IN SUPPORT OF DECISION-MAKING

Haroldo Issao Guibu[1] and João José Neto[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Brazil
[2]Escola Politécnica da Universidade de São Paulo, Brazil

ABSTRACT

This work presents the use of Adaptive Coloured Petri Net (ACPN) in support of decision making. ACPN is an extension of the Coloured Petri Net (CPN) that allows you to change the network topology. Usually, experts in a particular field can establish a set of rules for the proper functioning of a business or even a manufacturing process. On the other hand, it is possible that the same specialist has difficulty in incorporating this set of rules into a CPN that describes and follows the operation of the enterprise and, at the same time, adheres to the rules of good performance. To incorporate the rules of the expert into a CPN, the set of rules from the IF - THEN format to the extended adaptive decision table format is transformed into a set of rules that are dynamically incorporated to APN. The contribution of this paper is the use of ACPN to establish a method that allows the use of proven procedures in one area of knowledge (decision tables) in another area of knowledge (Petri nets and Workflows), making possible the adaptation of techniques and paving the way for new kind of analysis.

KEYWORDS

Adaptive Petri Nets, Coloured Petri Nets, Adaptive Decision Tables

1. INTRODUCTION

Coloured Petri Nets are an improvement of the original Petri Nets introduced by Carl Petri in the 1960s. Because of their ability to describe complex problems, their use has spread both in the engineering area and in the administrative area. Adaptive Coloured Petri Nets introduces an adaptive layer composed of several functions capable of changing the network topology, including or excluding places, transitions and arcs. In the area of decision support systems, the tools most used by specialists are the decision tables, which gave rise to several methods to help managers in their choices.

Among the methods developed for decision support there are the so-called multicriteria methods, which involve the adoption of multiple, hierarchically chained decision tables. In the process of improving the decision tables, new features are observed, which, although more complex, give the specialists the ability to describe their work model in a more realistic way. This paper describes the operation mode of the decision tables and the way of transcribing the rules of the tables for extended functions of Petri nets. By embedding a decision table in a Petri net, the simulation and analysis tools available in the Petri net development environments can be used, which leads to an increase in confidence in the decision criteria adopted.

2. DECISION TABLES

The Decision Table is an auxiliary tool in describing procedures for solving complex problems [9]. A Conventional Decision Table, presented in Table 1, can be considered as a problem composed of conditions, actions and rules where conditions are variables that must be evaluated for decision making, actions are the set of operations to be performed depending on the conditions at this moment, and the rules are the set of situations that are verified in response to the conditions .

. Table 1. Conventional Decision Tables.

	Rules column
Conditions rows	Condition values
Actions rows	Actions to be taken

A rule is constituted by the association of conditions and actions in a given column. The set of rule columns should cover all possibilities that may occur depending on the observed conditions and the actions to be taken. Depending on the current conditions of a problem, we look for which table rules satisfy these conditions:

• If no rule satisfies the conditions imposed, no action is taken;

• If only one rule applies, then the actions corresponding to the rule are executed;

• If more than one rule satisfies the conditions, then the actions corresponding to the rules are applied in parallel.

• Once the rules are applied, the table can be used again.

• The rules of a decision table are pre-defined and new rules can only be added or deleted by reviewing the table.

2.1. Adaptive Decision Tables

In 2001 Neto introduces the Adaptive Decision Table (ADT) [7] from a rule-driven adaptive device. In addition to rule lookup, an ADT allows you to include or exclude a rule from the rule set during device operation. As an example of its potential, Neto simulates an adaptive automaton to recognize sentences from context-dependent languages. In the ADT a conventional decision table is the underlying device to which a set of lines will be added to define the adaptive functions.

Adaptive functions constitute the adaptive layer of the adaptive device governed by rules. Modifying the rule set implies increasing the number of columns in the case of rule insertion, or decreasing the number of columns in the case of rule deletion. In both cases the amount of lines remains fixed. The Adaptive Decision Table (ADT) is capable to change its set of rules as a response to an external stimulus through the action of adaptive functions [7]. However, the implementation of more complex actions is not a simple task due to the limitation of the three elementary operations supported by ADT [9].

When a typical adaptive device is in operation and does not find applicable rules, it stops executing, indicating that this situation was not foreseen. For continuous operation devices, which do not have accepting or rejecting terminal states, stopping their execution would recognize an unforeseen situation and constitutes an error.

2.2. Extended Adaptive Decision Tables

To overcome this problem faced by continuous operation devices, Tchemra [9] created a variant of ADT and called it Extended Adaptive Decision Table (EADT), shown in Table 2

. Table 2. Extended Adaptive Decision Table.

		Adaptive Actions	**Rules**
Conventional Decision Table	Criteria	Set of elementary Adaptive actions	Criteria values
	Alternative		Actions to be applied
Set of auxiliary functions	Auxiliary functions		Auxiliary Functions to be called
Adaptive layer	Adaptive functions		Adaptive actions to be performed

In EADT, adaptability does not apply only during the application of a rule, but also in the absence of applicable rules. A modifier helper device is queried and the solution produced by the modifier device is incorporated into the table in the form of a new rule, that is, in the repetition of the conditions that called the modifier device, the new rule will be executed and the modifier device will not need to be called.

3. PETRI NETS

3.1. Ordinary Petri Nets

Petri nets (PN) were created by Carl Petri in the 1960s to model communication between automata, which at that time also encompassed Discrete Event Systems (DES). Formally, a Petri net is a quadruple PN= [P, T, I, O] where

P is a finite set of places;

T is a finite set of transitions;

I: (P x T) $\rightarrow$ N is the input application, where N is the set of natural numbers;

O: (T x P) $\rightarrow$ N is the output application, where N is the set of natural numbers

A marked network is a double MPN = [PN, M], where PN is a Petri net and M is a set with the same dimension of P such that M (p) contains the number of marks or tokens of place p.

At the initial moment, M represents the initial marking of the MPN and it varies over time as the transitions succeed. In addition to the matrix form indicated in the formal definition of the Petri nets, it is possible to interpret the Petri nets as a graph with two types of nodes interconnected by arcs that presents a dynamic behaviour, and also as a system of rules of the type "condition $\rightarrow$ action" which represent a knowledge base.

Figure 1 shows a Petri net in the form of a graph, in which the circles are the "Places", the rectangles are the "Transitions". The "Places" and the "Transitions" constitute the nodes of the graph and they are interconnected through the oriented arcs

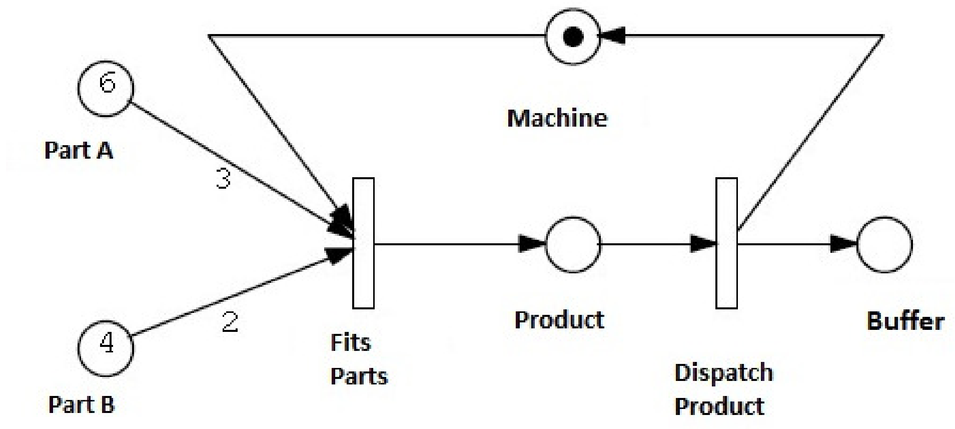

Figure 1. Example of a Petri Net

3.2. Reconfigurable Petri Nets

Several extensions of the Petri Nets were developed with the objective of simplifying the modelling of Discrete Events Systems, even for distributed environments. However, m Several extensions of the Petri Nets were developed with the objective of simplifying the modelling of Systems to Discrete Events, even for distributed environments. However, most extensions are not designed to model systems that change during their operation.

One group of Petri Nets extensions that attempts to tackle the problem of modelling systems that change during their operation is composed by the Self-Modifying Petri Nets [10], by the Reconfigurable Petri Nets via graph rewriting [6], by the Adaptive Petri Nets [1] and the Adaptive Fuzzy Petri Nets [5]. Each of these extensions has its own characteristics, but they share the fact that they can modify, during execution, the firing rules of the transitions or the topology of the network.

With the same name, the same acronym, but of different origins, we find in the literature Reconfigurable Petri Nets (RPN) introduced in [3] and in [6]. The work of Llorens and Oliver is an evolution of the work of Badouel and Oliver [1] and combines the techniques of graph grammars with the idea of Valk's Self-Modifying Petri Net, creating a system of rewriting of the network. In their work, Llorens and Oliver demonstrated the equivalence between the RPN and the PN in terms of properties and also that the RPN are equivalent to the Turing machines regarding the power of expression.

In Figure 2 we have schematized a Reconfigurable Petri Net according to Guan [3]. There are two interdependent layers, the control layer and the presentation layer. The places of the control layer are different in their nature from the places of the presentation layer.

Each place of the control layer has associated a set of functions that is capable to change the topology of the presentation layer, that is, they reconfigure the presentation layer. The tokens of the places are actually functions designed to modify the topology of the presentation layer.

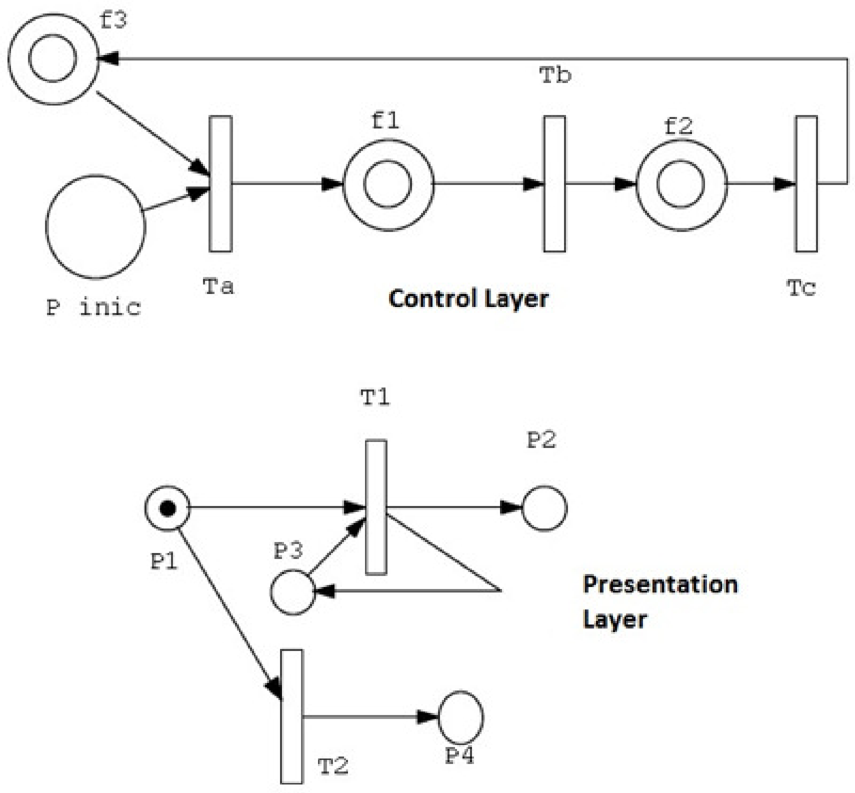

Figure 2. Reconfigurable Petri Net, version of Guan

3.3. Adaptive Petri Nets

An adaptive Petri net was defined by Camolesi [2] from the underlying device scheme plus adaptive layer as follows:

$APN = (C_0, AR_0, \Sigma, c_0, A, NA, BA, AA)$ in initial configuration c_0.

Input stimuli move the APN to the next configuration if, and only if, a non-empty adaptive action is performed. In the k-th step we have:

$APN_k = (C_k, AR_k, \Sigma, c_k, A, NA, BA, AA)$, where

$APN = (PN_0, AM)$ is formed by an underlying initial device (PN_0) and an adaptive mechanism AM;

PN is the Petri Net device in step k. PN_0 is the initial underlying device and the set CR_0 represents the initial non-adaptive behaviour;

C_k is the set of all possible behaviours of PN in step k and $c_k \epsilon C_k$ is its initial behavior in step k;

ε ("empty string") denotes absence of valid element;

Σ is the set of all possible events that make up the input chain;

A⊆C is the subset of PN acceptance configurations;

F = C-A is the set of PN rejection configurations;

BA and AA are sets of adaptive actions, which include empty action;

$w = w_1 w_2 \dots w_n$ is the input string;

NA is a finite set of all symbols that can be generated as output by APN in response to the application of adaptive rules;

AR_k is the set of adaptive rules that define the adaptive behaviour of APN in step k and is given by a relation $AR_k \subseteq BA \times \Sigma \times C \times RP \times AA$.

AR_0 defines the initial behaviour of APN and the adaptive actions of insertion or elimination of places and transitions are transforming the set of rules.

The rules $reg \in AR_k$ are of the form

(<ba>, (P, T, I, O), <aa>) and operate as follows:

A symbol $\sigma\epsilon\Sigma$ causes reg to execute the action $ba \ \epsilon \ BA$. If the action of $ba \ \epsilon \ BA$ deletes reg from AR_k, the execution of reg is aborted, otherwise, the underlying rule of reg = (P, T, I, O) applies. Finally, the adaptive action $aa \ \epsilon AA$ is performed .

3.4. Adaptive Coloured Petri Nets

The Adaptive Coloured Petri Net uses the same scheme of wrapping the underlying device (CPN) with an adaptive layer (AL).

ACPN = (CPN, AL) where

CPN is the conventional coloured Petri net,

LA = (AF, AR) is the adaptive layer.

In turn, the adaptive layer is composed by the set of adaptive functions (AF) and by the set of rules type IF - THEN (AR).

AF is the set of adaptive functions and is embedded in the Adaptive Coloured Petri net.

AR is the set of rules that must be inserted in the Adaptive Coloured Petri net through the execution of the adaptive functions.

The basic adaptive functions are inspection, insertion or incorporation and exclusion of a rule.

The ACPN uses the same strategy of RPN devised by Guan, but the control layer is a kind of interpreter of Decision Tables previously defined in order to produce the decision layer.

4. METHODOLOGY

The operation of the ACPN is based on an algorithm composed of four main phases:

Phase I: definition of the underlying decision table, with the inclusion of the criteria, alternatives and rules of the decision problem;

Phase II: generation of the decision matrix, whose values represent the relative weights and preferences of criteria and alternatives of the decision maker;

Phase III: transformation of the decision matrix in a XML format, in order to incorporate as a set of rules.

Phase IV: appending the XML file to the basic ACPN, resulting in the second layer, the decision layer.

The following example was adapted from [9] to illustrate the sequence of phases mentioned above, based in a decision procedure proposed in [8]. This is a decision problem in which the decision maker needs to certify and select suppliers of a particular product for his company. Two suppliers A and B are analyzed, according to the judgments of the decision-maker in accordance with selected for comparison:

C1 - Quality of services;

C2 - Flexibility for product delivery;

C3 - Distance from supplier and company location.

In phase I, criteria and alternatives to the problem are introduced in a conventional decision table, and the decision maker can create an initial set of rule, as showed in Table 3.

Table 3. Initial Decision Table.

		Rule1	**Rule2**	**Rule3**	**Rule4**
Criteria	C1 - Quality	Y	N	N	Y
	C2 - Flexibility	Y	Y	Y	Y
	C3 - Distance	Y	N	Y	N
Alternatives	A1 – Supplier A	X		X	X
	A2 – Supplier B		X		

In this example, the comparisons between the criteria are shown in Table 4, in which the decision-maker judges the criteria pairs, obtaining the matrix of reciprocal judgments[7].

Table 4. Judgement Matrix.

	C1	**C2**	**C3**
C1 - Quality	1	4	3
C2 - Flexibility	1/4	1	2
C3 - Distance	1/3	1/2	1

After checking the consistency of the values of judgment and normalization of values, the weights of each criterion are calculated, generating the vector of weights:

W = (0.62, 0.22, 0.16).

According to the judgment of the decision maker, the vector with the weight of each criterion represents the relative importance of each one. In this example, the resulting weights indicate that the criterion is more important in relative to others:

Quality: 0.62 - Flexibility: 0.22 -Distance: 0.16.

At this point, it is necessary to check the consistency of the criteria judgments. The matrix of comparison between the criteria of Table 4 is evaluated for the verification of the degree of consistency of the judgments, which is given by the consistency ratio (CR), as a function of the order of the matrix:

a) vector of weights pe = $(1.98, 0.69, 0.47)^T$

b) consistency vector cs = $(3.20, 3.08, 3.04)^T$

c) eigenvalue λmax = 3.11

d) consistency index CI = 0.05

e) consistency ratio CR = 0.09

According to [8], the value of CR = 0.09 indicates that the judgments are consistent and acceptable since CR <10%, otherwise it would be necessary to review Table 4. The next operation is to obtain the performance matrix. For this, the alternatives are compared to the pairs, with each of the criteria. The comparisons made by the decision maker in the example are shown in Table 5.

Table 5. Pairwise comparison matrix.

	C1		**C2**		**C3**	
	A1	A2	A1	A2	A1	A2
A1 – Supplier A	1	8	1	6	1	4
A2 – Supplier B	1/8	1	1/6	1	1/4	1

Normalizing the matrices, we obtain the following values:

$$z_{1,1} = 0.89 \quad z_{1,2} = 0.86 \quad z_{1,3} = 0.80$$
$$z_{1,2} = 0.11 \quad z_{2,2} = 0.14 \quad z_{2,3} = 0.20$$

which are performance matrix cells.

Table 6. Performance matrix Z.

	C1	C2	C3
A1 – Supplier A	0.89	0.86	0.80
A2 – Supplier B	0.11	0.14	0.20

From the performance matrix Z we obtain vector ax containing the figures indicating the relative importance of the alternatives.

$ax_1 = 0.85$ and $ax_2 = 0.15$ indicating that in this example the alternative A1 is much better than the alternative A2 and supplier A must be chosen. Figure 3 shows the Petri Net version of the initial decision table.

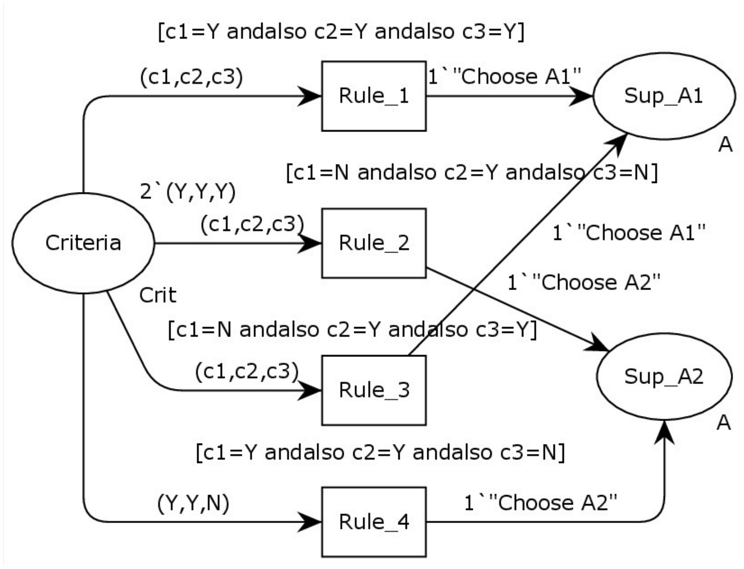

Figure 3. Petri Net equivalent of initial Decision Table

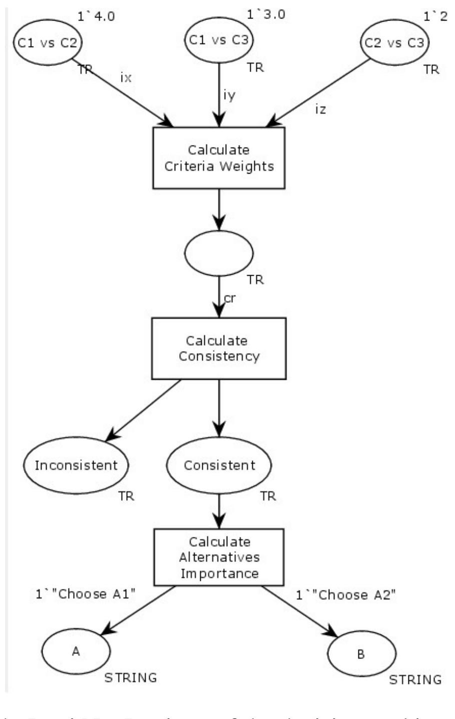

Figure 4. Petri Net Petri net of the decision-making process

Figure 4 summarizes the decision process in a graphical way.

5. CONCLUSIONS

In this paper we show how to incorporate decision tables in Petri nets. Well-established procedures in decision aid using decision tables can be used in a new tool that has additional analysis features to detect inconsistencies in procedures [4][13]. In daily activities, the business expert does not share his knowledge with the factory manager, losing synergy. Sharing two knowledges that are usually isolated provides a synergy. Good management practices are often disregarded in the day-to-day running of an enterprise because of the unawareness of the consequences of certain decisions, which are not always apparent.

In other areas where the use of Petri nets is widespread, gain is also possible, for example in flexible manufacturing systems [11][12]. Reconfigurable networks can be understood as a particular case of adaptive networks, where adaptation is achieved through reconfiguration of the network. The adaptive network is more general than the reconfigurable network because it can change its behavior while maintaining the same configuration by modifying the firing rules of the transitions.

In an adaptive network, the rules for operation in case of failures can be incorporated into the standard network, allowing greater agility in the operation without the need to stop the process until the experts in the failures take control. The recommendations of the experts would already be incorporated into the standard Petri net used in monitoring the operation. Reconfigurable systems during operation are a trend in the design of control systems and the ability to incorporate procedures from related areas is a feature that cannot be underestimated.

REFERENCES

[1] Badouel E. ; Oliver, J. Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes in Workflow Systems. (S.l.),1998.

[2] Camolesi, A. R. Proposta de um Gerador de Ambientes para a Modelagem de Aplicações usando Tecnologia Adaptativa. Tese (Doutorado) —Escola Politécnica da Universidade de São Paulo, 2007.(in portuguese).

[3] Guan S. U. ; Lim, S. S. Modeling adaptable multimedia and self-modifying protocol execution. Future Generation Computing Systems, vol.20, no 1, pp. 123-143, 2004.

[4] Kheldoun, A., Barkaoui, K., Zhang, J., & Ioualalen, M. (2015, May). A High Level Net for Modeling and Analysis Reconfigurable Discrete Event Control Systems. In IFIP International Conference on Computer Science and its Applications_x000D_ (pp. 551-562). Springer International Publishing.

[5] Little T. D. C.; Ghafoor, A. Synchronization and storage model for multimedia objects. IEEE J. Selected Areas Comm., pp. 413-427, Apr.1990., 1990.

[6] Llorens, M. ; Oliver, J. Structural and dynamic changes in concurrent systems: Reconfigurable petri nets. IEEE Trans. Comput., vol. 53, no.9, pp. 11471158, 2004.

[7] Neto, J. J. Adaptive rule-driven devices - general formulation and case study. Proc. 2001 Lecture Notes in Computer Science. Watson, B.W.and Wood, D. (Eds.): Implementation and Proc. 2001 Lecture Notes in Computer Science. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conf., Springer-Verlag,Vol.2494, pp. 234-250., 2001.

[8] Saaty, T.L., “How to make a decision: the Analytic Hierarchy Process”, Interfaces, Vol. 24, No. 6, pp19–43, 1994

[9] Tchemra, A. H. Tabela de decisão adaptativa na tomada de decisões multicritério. Tese (Doutorado), Escola Politécnica da USP, São Paulo, 2009(in portuguese).

[10] Valk, R. Self-modifying nets, a natural extension of petri nets. Lecture NotesLecture Notes in Com, vol. 62, pp. 464-476, 1978.

[11] Zhang, J., Khalgui, M., Li, Z., Mosbahi, O., & Al-Ahmari, A. M. (2013). R-TNCES: A novel formalism for reconfigurable discrete event control systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 43(4), 757-772.

[12] Zhang, J., Khalgui, M., Li, Z., Frey, G., Mosbahi, O., & Salah, H. B. (2015). Reconfigurable coordination of distributed discrete event control systems. IEEE Transactions on Control Systems Technology, 23(1), 323-330.

[13] Zhang, J. (2015). Modeling and verification of reconfigurable discrete event control systems (Doctoral dissertation, XIDIAN UNIVERSITY).

AUTHORS

Haroldo Issao Guibu
graduated in Electrical Engineering and MSc in Electrical Engineering in Polytechnic School of São Paulo (EPUSP) University, Brazil. Lecturer at Instituto Federal de Educação, Ciência e Tecnologia de São Paulo(IFSP) with main interest in Automation, PLC programming and Automation related adaptive technologies.

João José Neto
graduated in Electrical Engineering (1971), MSc in Electrical Engineering (1975) and doctor in Electrical Engineering (1980), and "livre docente" associate professor (1993) in the Polytechnic School of São Paulo (EPUSP) University,Brazil. Nowadays he is the head of LTA - Adaptive Technology Laboratory at the Department of Computer Engineering and Digital Systems at the EPUSP. His main experience is in the Computer Science area, with emphasis on the foundation of computer engineering and adaptivity. His main activities include adaptive devices, adaptive technology, adaptive automata and their applications to computer engineering and other areas, especially in adaptive decision making systems, natural language processing, compiler construction, robotics, computer education, intelligent system modeling, computer learning, pattern matching, inference and other applications founded on adaptivity and adaptive devices.