DEVELOPING A PARALLEL MODEL FOR OIL PRODUCING PROCESS

Hamdi A. Awad

***Abstract*—This paper introduces a novel Petri net model for an important process in oil industry named oil producing process (OPP) for two reasons. First, the OPP has huge modules. Second it has continuous and discrete modules. In this parallel model, the continuous modules are modeled using intelligent systems while the discrete modules designed using Petri nets. Finally, these continuous and discrete modules are merged in a unified parallel frame work. Two main issues are discussed for developing this unified frame work: First, which graphical tool is appropriate for oil industry modeling and supervision? and second, what methodology is appropriate for modeling and analysis of such industries?. Testing, verifying, and validating the developed parallel model is performed using the Petri net tool software version 2.1 using real data that are collected from the field of the OPP and its catalogues. Simulation results show that the developed OPP Petri net model has small computational demand due to its compact structure that drastically reduces the scan time limited by the available technologies.**

***Index Terms*—Automation systems, Hybrid systems, Discrete event systems, intelligent systems, Petri nets.**

I. INTRODUCTION

Prospective to modeling oil industry is still unexploited. This leads to some difficulties to enhance and increase the mass production of this important industry due to the fact that most of industrial processes supervisory control is Ad hoc approaches. These schemes are difficult to follow when the process to be supervised is complex, uncertain and have continuous and discrete dynamics. Frequent shutdown due to such Ad hoc techniques leads to the loss of billions of dollars in oil industry every year.

An oil producing process (OPP) has discrete and continuous modules interacted with each other. In this paper, the OPP will be completely modeled using the graphical tool highlighted in this paper. In the OPP there are three primary pump stations which receive stabilized crude oil from different stabilizers and outputs to the main two shuttle lines which divert oil to different production lines. A flow control valve (FCV) usually controls the flow amount in each shuttle line. Unfortunately, the pressure and flow of these shuttle lines can be disturbed due to operating problems, such as pump vibrations and high temperature which cause shaft and bearing friction. In this case the Emergency shutdown task (EST) would isolate the effected pump. Of course, this is not the ideal solution; a new technique for fault detection and isolation (FDI) will be discussed in our future work.

H. A. Awad is with the Dept. of Industrial Electronics and Control Eng., University of Menoufia, Faculty of Electronic Eng., Menouf, 32952, Egypt. Awadhaa@yahoo.co.uk.

Grafcets and finite automata schemes are usually employed for modeling discrete event systems (DES) [1], [2] however these schemes are impractical for systems with large numbers of states due to rapid increases in complexity and consequently computation demanding. One way of dealing with these problems is to model DES using Petri nets that avoids the state explosion problem associative with using automata and ad hoc feature associative with using grafcets. Petri net (PN) models are more compact than similar automata based models and are better suited for the representation of DES [3], [4]. So, they are used to model many successful applications such as flexible manufacturing [5] industrial automation and robotics [6], and batch chemical processes [7]. This fact will be assured in section III.

Unlike, the gradient descent algorithms, genetic algorithms are stochastic search methods [8]. They find the global solution of a given problem and use only a simple scalar performance measure that avoids the derivative problems of back propagation algorithms. Genetic algorithms were used in many applications include off-line [9] and on-line identification and control systems [10]. Recently, new evolutionary computing methods named particle swarm algorithms are proposed [11]. Similar to a genetic algorithm (GA), a particle swarm algorithm (PSA) is initialized with a population of random solutions, however, it was developed based on the analogy of the swarm theory of bird flocking and fish schooling. Each individual in the PSA is called particle and it is assigned with a randomized velocity according to its own and buddy flying experiences. Compared with the GA, the PSA has good memories that retain good solutions and has constructive cooperation between particles (solutions). Successful applications of the PSA to several optimization problems assured its potential. So, on one hand, it will be employed to model continuous modules inside the OPP. Modeling such modules using PSA is due to the fact that many industrial continuous modules e.g. industrial valves have not models in any mathematical or graphical description. Of course, there are mathematical models for valves, but no model for industrial valves. In other words, control valve sizing refers to the engineering procedure of determining the correct size of a control valve for a specific installation. A capacity index called the valve flow coefficient, C_v, has greatly reduced the difficulty in "sizing" a control valve. Accordingly, a particular valve can be ordered from companies based on its size with no model. This leads to some difficulties at modeling industrial oil hybrid systems such as the OPP as mentioned above.

Hybrid systems have a set of control levels [12]. The local control level deals with processes that are usually continuous in nature. It is therefore superimposed by higher levels of the control hierarchy, ranging from supervisory control, production planning to business process management, which are discrete in nature. The study of hybrid processes is the corner stone for structuring their reliable models and consequently designing supervisory controllers. In this paper, developing a unified frame work that includes both discrete and continuous modules of the OPP to be modeled is the main target.

Two main themes will be discussed in this paper, highlighting Petri nets for modelling discrete event systems and employing intelligent schemes for modelling continuous modules included in hybrid industrial systems. Developing a unified systematic frame work for modelling the industrial OPP is the main contribution of this paper. This contribution can be achieved as follows.

1- Highlighting Petri nets as a graphical tool for modelling the OPP
2- Modelling continuous modules of the OPP using the PSA
3- Merging the discrete and continuous modules of the OPP in a unified frame work
4- Analysis the developed Petri net model of the OPP using an efficient Petri net tool.

This paper is organized as follows: Section II gives the essential background to Petri nets used in this paper. Section III highlights Petri nets as graphical tools for modelling the OPP. Section IV describes the oil producing process (OPP), details a modelling methodology used, and models this industrial oil producing process. Simulation results and conclusions are given in section V.

II. Back ground to Petri Nets

Petri nets are classed based on their structural and behavioral properties [3]. This section summaries these properties and employs them to analysis the developed Petri net model of the OPP. Basically, a Petri net is a particular kind of bipartite directed graphs characterized by three types of objects, places, transitions, and directed arcs. The latter connects places to transitions or transitions to places. This elementary Petri net may be employed to represent various aspects of the system to be modeled. In order to study the dynamic behavior of a system structured with a Petri net in terms of its states and state changes, each place may potentially hold with positive number of tokens. A set of definitions related to Petri nets used in this paper can be briefly discussed below [3], [4], [13].

Definition 1: A Petri net structure is a weighted bipartite graph; $G = (P, T, F, W)$, where $P = \{p_1, p_2, ..., p_n\}$ is a finite set of places, $T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions, $F \subseteq (PxT) \cup (TxP)$ is a set of arcs from places to transitions and from transitions to places in the structure, and $W : I \cup O \to \{1,2,3,,...\}$ is the weight function on the arcs, the set of input /output places of transition $t_i : {}^{\bullet}t = I(t_j)$ $t_i : t^{\bullet} = O(t_j)$.

In general, a transition without any input place is called a source transition, and a transition without any output place is named a sink transition. Source transition is unconditionally enabled (sensors), while a sink transition consumes tokens. A pair of a place, *p*, and a transition, *t*, is called a self-loop if the place, *p*, is both an input and output to the transition, *t*. A Petri net is said to be pure if it has no self-loops while it is said to be ordinary if all of its arc weights are ones. A token in a place indicates the fact that the condition described by that place is satisfied. The way in which tokens are assigned to a Petri net graph defines a marking vector, *M*, which can be formally defined $M = [m(p_1), ... m(p_i), ..., m(p_n)]$ where *n*, is the number of places in the Petri net. The state transition mechanism is provided by moving tokens through the Petri net and hence changing its state. When a transition is enabled, it can be fired or not depends on its condition that is free or timed.

Definition 2: A transition $t_i \in T$ in a Petri net is said to be enabled if the number of tokens equal or exceed the weights on its output arcs, $M(p_i) \geq W(p_i, t_j), \forall p_i \in I(t_j)$. Since places are associated with conditions, a transition is enabled when all the conditions required for its occurrence are satisfied using a set of tokens.

Definition 3: Given a Petri net with the initial marking M_o; $N = (G, M_0)$, the set of reachable states of this Petri net is called the reachability set and is signified by $R = (N, M_0)$.

Definition 4: A Petri net is said to be *k*-bounded if the number of tokens in each place does not exceed a constant number *k* for any reachable marking vector from its initial state. It is also said to be safe if it is one-bounded.

Definition 5: A Petri net is said to be live if it guarantees deadlock-free operation, regardless what firing sequence is chosen.

In contrast to the behavioral properties defined above that depend on the initial marking vector of a Petri net, structure properties do not depend on the initial marking vector of the net. Based on the structural properties, Petri nets are classified also into many classes. These important classes are described below.

Definition 6: A marked graph is a Petri net in which all places have a single input and a single output transition.

Definition 7: A state machine (*finite state automata*) is a Petri net in which all transitions have one input and one output place.

Definition 8: A free choice net is a net such that for every arc from a place *p* to a transition *t;* $(p \to t)$ such that

- *t* is the only output transition of *p* (conflict free), and
- *p* is the only input place of *t* (synchronization free)

Definition 9: An interpreted Petri net (IPN) that has (Q, M_0) structure is described as follows.

$$Q = (G, \Sigma, \lambda, \phi) \qquad (1)$$

Where G is a Petri net (PN) graph (see definition 1), $\Sigma = \alpha_1, \alpha_2, ..., \alpha_r$ is the alphabet of input symbols. λ is a labeling function of transitions. $\phi : R(Q, M_0) \rightarrow (Z^+)^q$ is an output function, q is the number of outputs and M_0 is the initial marking of the net.

In short, the above definitions are employed to model and analysis the OPP in this paper. The IPN is the most suitable Petri net for this objective due to its simplicity, and small computational demanding. For more definition related to Petri nets the reader can be directed to [3], [4], [13], [14].

III.

IV. Investigating the Best Graphical Scheme for Modeling

This section investigates the best graphical and mathematical tool for modeling industrial oil processes. It shows the superiority of Petri nets to grafcets and automata. Recently, many various tools to design and analysis discrete event systems (DES) were developed. In general, these tools can be classed into three main groups: Graphical, algebraic, and formal language-based tools [15]. The first group includes automata, reactive flow diagrams, ladder diagrams, Grafcet, and Petri nets. The second group includes Boolean algebra, temporal logic, and max-plus algebra. The third group includes formal language approaches, and real-time programming languages. These three groups are not equivalent with respect to the application field. For example, max-plus algebra is an effective method to analysis job scheduling, while Grafcet is useful for the specification of sequential control schemes [16].

Ladder diagram (LD) and grafcets are the main graphical programming languages of industrial programmable logic controllers (PLC) that are usually employed for supervising oil industry processes [17], [18]. For simple systems, it is easy to write down PLC programs using an ad hoc method, however, as the complexity of a system increases this programming method becomes very difficult to tackle problems effectively. Also, verifications of PLC programs are made through meticulous method that takes long time to be executed and some errors may pass without being seen in complex systems. On one hand, LD programs are difficult to debug because they are written through trial and error in many cases. On the other hand, grafcets represent a process model in its safe mode; one token in a place, that is a special case of Petri nets [2]. These shortcomings direct us to looking for a powerful modeling tool to be used for modeling the OPP.

Further, emergency shutdown task (EST)'s scan time is very important and critical criterion in oil industry [19]. Some oil companies do not accept any EST if the scan time execution exceeds 100 msec. So low scan time is a valuable achievement in oil industry. A set of control parameters or blocks such as conventional controllers and timer will increase the scan time which will exceed the required time. Some of the PLC companies tried to solve this issue by dividing the programs in different hardware controllers that will reduce the scan time, however, this solution costs more money. Reducing the scan time using a powerful tool is a big and another challenge in this paper.

A. Petri nets and grafcets

The international electro technical committee standard on programming languages of PLC promotes the use of sequential function chart or (grafcet) of the control logic. Grafcet inherits many of its features from the theory of Petri nets (PN) and represents the processes with safeness property [20]-[21]. However, hybrid systems are discrete, continuous, and usually are not safe. Unlike grafcets, PNs are mathematical-based and matrix-manipulated approaches that combine a well defined mathematical theory with a graphical representation of the dynamic system behavior. The theoretic aspect of PN allows precise modeling and analysis of a system behavior, while the graphical representation of PNs enables visualization of the changes of the system states [22]. This combination is the main reason for the great success of Petri nets used to model various kinds of dynamic event-driven systems unlike grafcets. Also, many PN-based software modules are published recently: Mitsubishi, Schneider Automation group, Siemens, ABB Automation, Fatek, Omron, and Allen Bradley [23]. Out of these modules, Siemens modules are commonly used in oil industry and are recommended in this paper.

There are however two basic differences between PNs and grafcets. First, the marking of a grafcet scheme or simply grafcet is Boolean (0 or 1) while the marking of a place in a Petri net is any positive integer. Accordingly, the conversion of a Petri net to a grafcet is possible only when the net is safe. Second, any pair of transitions in a conflict result some differences in firing rule of a Petri net and a grafcet. This leads to a non-deterministic behavior of a Petri net since there is no rule to choose which of the enabled transitions will be fired. When such a situation emerges in grafcet the transitions are fired according to their priority to ensure the deterministic behavior. Also, out of the graphical tools, ladder diagram (LD) is the major programming languages of industrial programmable logic controllers (PLC) and can be compared with Petri nets (see appendix-B).

As a final conclusion, the mathematical power, generality, and graphical features of PN-tool make it superior to grafcets and LD [16]. Points to be highlighted from the above brief discussion can be summarized as follows.

- It is possible to represent every grafcet by an equivalent Petri net, but is not vice versa.
- Grafcet are Ad hoc approach while Petri nets are systematic approach
- Petri nets have good power of mathematics compared with grafcets

B. Petri nets and automata

While automata have one type of nodes namely “states”, the Petri nets have two types of nodes, called “transitions” and “places”. This includes the possibility to branch the net parts not only at the places, but also at the transitions. Accordingly the advantage of system modeling with Petri nets is the opportunity to model parallel sequences and their synchronization in effective manner compared with automata. Petri nets can be understood as a generalization of automata. To clarify this point, let us consider two states employed for representing motor's states using automata, while a marking vector employed for representing the same motor using Petri nets. In other words, Petri nets can represent this motor using marking vector concept as, $M_0=[0\ 1]^T$, while automata have not this concept. Now, if the complexity of a system increases as will be discussed below, the number of states needed by automata for modeling this system will be exponentially increased, while the mathematical power of Petri nets overcome this problem easily by using two simple equations; state and output equations defined in (2) and (3) respectively.

$$M(k)=M(k-1)+C\,q \qquad (2)$$

$$Y(k)=\phi_k M(k) \qquad (3)$$

where ϕ_k is the output matrix, $M(.)$ is the marking vector, $Y(k)$ is the output of the PN net and k is the time step.

Grafcets, Petri nets, timed Petri nets, automata, and timed automat are graphical tools; however, an effective modeling tool able to provide a representation of the system to adapt this paper goal is required. Consequently, this paper adopts the time Petri net model [24] to be compared with timed automata to detect that tool. Although timed graphical tools do not use in this paper, they are used to just clarify the superiority of Petri nets to automata. The difference between conventional graphical tools and timed graphical tools is time intervals associative with their transitions. Moreover, selecting Petri nets and timed automata is due to the close two features between them; graphical tools and mathematical power. To clarify this point, this paper briefly reviews the example published in [25] as follows.

The aim of this example is to model a system that glues together two parts with timed Petri net (TPN) and timed automata (TA) to show the power of them. The part one is transported by a robot and putted down on the part two for gluing purpose. The robot needs 4–5 time units (t.u.) to transport the first part and must begin its task within 2 t.u. since the beginning of the process. The part two is prepared for gluing by an operator who puts the glue in one t.u. A good quality gluing requests the glue to be neither too humid, nor to dry that is achieved within 3 t.u. since the operator has started to add the glue. The controllable events are the beginning of each task however the accomplishing of these tasks is uncontrollable events. Modeling this process using timed Petri nets is described as follows.

Initially, neither the robot nor the operator has begun their tasks as depicted in Fig. 1. The robot task is modeled by places and transitions with the following time intervals, T_1 is [0, 2], T_2 is [4, 5]. Two events may occur when the robot puts down the first part on the second part, *a proper gluing* and a *failure* (firing T_3).

The operator task is also modeled another set of places and transitions according to the same principle as the robot task shown in Fig. 1. When the glue is ready, two behaviors are also defined, proper gluing (firing T_4) or failure (firing T_8). The goal is to obtain a proper gluing, so firing of the transitions T_3 and T_8 represents the forbidden behavior of this system.

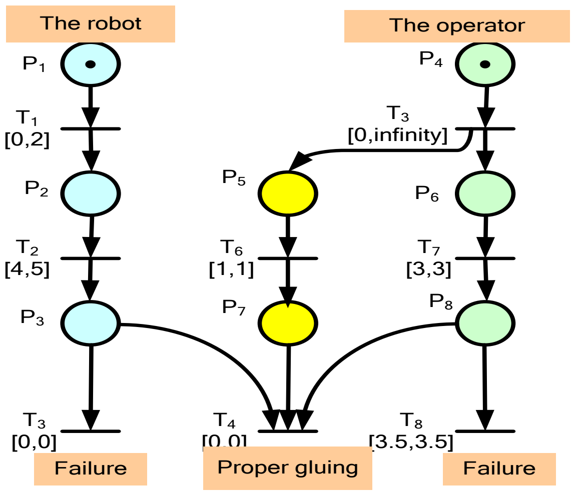

Fig.1. TPN model of the gluing process

The actions of the TPN depicted in Fig. 1 can be converted to TA-based model shown in Fig. 2 [25]. Each TPN transition Ti associates with a clock o$_i$. The initial TA location Γ_0 corresponds to the state where neither the robot, nor the operator has started their task. Starting from this location the automata model can be constructed with a set of locations and a set of transitions. The former corresponds to the marking vectors of the TPN shown in Fig. 1 and the latter corresponds to the timed transition of the TPN. Some of locations in the figure represent forbidden locations that must be avoided.

Compared with the TPN model of the glue process shown in Fig. 1, modeling of this process using timed automata depicted in Fig. 2 has a complex structure. A set of forbidden locations (yellow color) that may be not happen in real process adds more complexity to analyze the structured model using suitable software; Cygwin software that is a tool that makes the user able to use the Microsoft Windows as if anyone is working on UNIX environment (http://www.cygwin.com/). Consequently, it is possible to easily port many UNIX programs without the need for extensive changes to the source code. For more information, the reader can be directed to the website (http://cygwin.com/faq/faq-nochunks.html#faq.what). This means that modeling with timed automata details the process to be modeled with redundant locations. This leads to some difficulty to model and follow the timed automata for complex industrial processes due to state explosion problem of TA. For these reasons, this paper assures and recommends the use of Petri nets as a transparent graphical and powerful tool for modeling industrial oil processes.

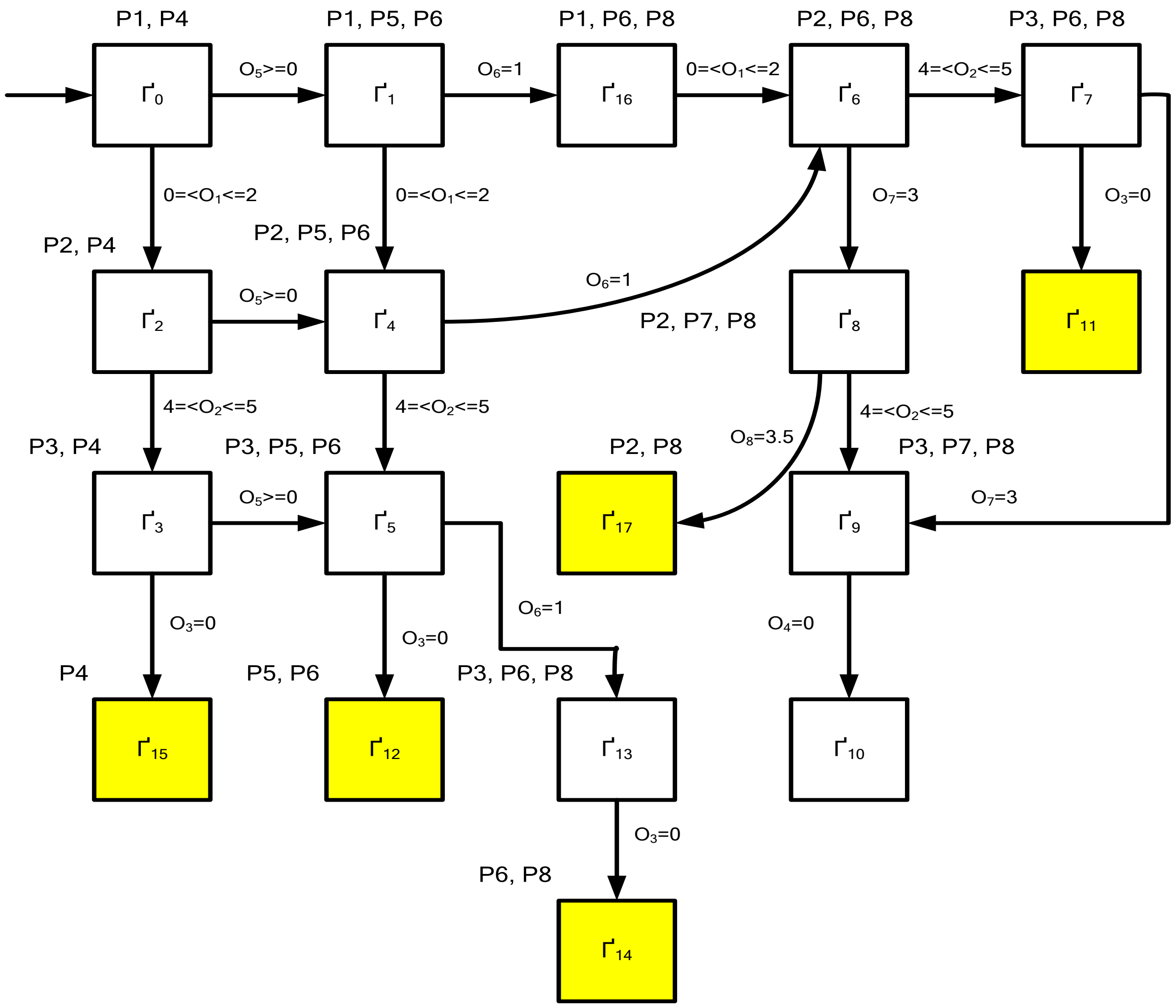

Fig. 2. The TA associated with the TPN shown in Fig. 1

C. Automata and Petri nets for supervisory control

Finite automata and Petri nets have been discussed as powerful methodologies for modeling hybrid systems. It has been shown that the states and transitions concept are key features of both models. Process modeling is an important phase in the supervisory synthesis procedure, thus there is a set of methods for designing supervisors based on automata system models [13], [26], however, the disadvantage of these methods is that they need a huge search over the system states. This search makes them impractical for systems with large number of states and transitions causing events. As the number of states increases, the *state space explosion* (complexity increases exponentially as the number of states increases) problem arises.

One way of dealing with these problems is to model discrete event systems with Petri nets. Petri net models are normally more compact than similar automata based models and are better suited to represent systems with ordered structures and large reachable state space. The Petri net-based solutions have several advantages over automata that can be listed as follows.

- In contrast to TA, the states of a Petri net are represented by the possible markings vector and not by the locations. Thus Petri nets give a more compact description compared with TA.
- A plant's specifications can be represented graphically in an easily understood format using Petri nets.
- Petri net models can be used for the analysis of their properties, performance evaluation and the systematic construction of discrete event supervisors using many available softwares.
- A Petri net model allows concurrency without suffering from increased model complexity, unlike finite state automata.

Compared with the supervisory control synthesis of a model may develop by automata, the control synthesis using Petri nets is easier, more systematic and transparent. Accordingly, this paper assured the superiority of using Petri nets as graphical and mathematical tools to automata that have complex structure. The complexity of the structure stems from the fact that automata interest with details that may not happen in real world while Petri nets do not.

D. Automata and Petri nets for supervisory control

In this section, the characteristics of a set of graphical tools are evaluated in the sense of seven criteria, transparency, simplicity, portability, generalization, testability, and computations demand. These criteria can be summarized as follows:
Transparency: This index is employed to assess the ease to follow.
Simplicity: This index is used to assess how easy a method designs a graphical model to a system to be modeled.
Portability: This index is employed to assess how applicable a method is to a range of commercially available PLCs.
Generalization: The range of applicability in real time supervisory control is assessed using this index.
Testability: This index is used to assess the ease with which the design can be tested before its implementation.
Computation Cost: This index is used to test the computations demand, small or heavy, of the graphical tool to be used.
Design and analysis: This index is employed to evaluate the method in the sense of Ad hoc or systematic. The former is not related to engineering while the latter means that the method can design, analysis, and control synthesis the whole Petri net in systematic manner.

These criteria can be used for comparison reasons as depicted in table I that assures the superiority of Petri nets to other graphical tools.

TABLE I
A COMPARISON BETWEEN GRAFCETS, AUTOMATA, AND PETRI NETS

Index	Grafcets	Automata	Petri nets
Generalization	No	Yes	Yes
Simplicity	Yes	No	Yes
Portability	Yes	No	Yes
Transparency	Yes	No	Yes
Testability	No	Yes	Yes
Design	Ad hoc	Systematically	Systematically
Computations	Medium	Heavy	Small

According to the above discussion this paper highlights Petri nets for modeling oil industry instead of other graphical tools. This recommendation is based on the following reasons. First, Petri nets have good power of mathematics. Second, they are transparent, and can capture all the dynamic features of the process to be modeled such as concurrency and conflict that are difficult to achieve by other graphical tools. Third, compared to TA, the TPN is easier to design and to follow. This leads to the fact that PN are more applicable graphical tool to abstract the process compared with other graphical tools. Unlike TA, many PN software modules are published recently: Mitsubishi, Schneider Automation Group, Siemens, Fatek, Omron, ABB Automation, Allen Bradley.

IV. THE INDUSTRIAL PRODUCING PROCESS

This section collects the essential information for an oil producing process (OPP) that comprises five units, the tank farm unit #2 (TFU-2), the booster pump (BP); G-0960 and G-0961, the shipper pump (SP); G-0955-G0959, the flow control valves (FCV), and the pressure reducing station (PRS). The BP consists of a motor driven booster pump with 1500 hp, while the SP comprises a motor driven booster pump with 10000 hp. The former is piped to take the suction from TFU-2 common header and discharge to the latter at pressure of 100 psi. The SP takes suction from the BP at TFU-2. It discharges into two shuttle lines headers at 650 psi, 48" and 56". The information collected from the expertise and the OPP is listed in this section that forms the recipe of this process and develops its parallel PN-model.

A. The TFU-2 module

The tank farm #2, TFU-2, contains two tanks which provide storage capacity for stabilized crude from stabilizers. Stabilized crude oil from the stabilizer is routed to the tanks. The oil level in these tanks is normally maintained within the range of (10~12) feet, however during emergency conditions the oil level increases at the range (35~40) feet. All tanks are utilized for oil storage. The tank farm number 2 performs the following in sequence:

- Open the outlet-block-valves on the tanks in TFU-2,
  Tank-2050: Valve 54
  Tank-2051: Valve 58
- Open the air operating valve (AOV-0152 & AOV-0163).
- All tanks in the TFU-2 receive sweet oil from the stabilizers via the rundown lines. The BP takes suction from the lines filling the TFU-2 tanks, and feed the shipper pups, SP.
- The BP will pump the oil from tanks rundown line to suction of SP.
- Normal operating level in the TFU-2 tanks is about 10-12 feet. The maximum operating level of the tanks in TFU-2 is 40 feet. Individual tanks will be closed using the main tank block valve when these limits are reached.
- The critical level for TFU-2 operation is reached when one or more tanks are filled to 35 feet level or higher. When operating TFU-2 beyond the critical level, the oil dispatcher and shift coordinators will be frequently informed of changes in tank levels.
- When the tank-level reaches 35 feet, the level should be gauged by hand tape every hour if the tank is not

isolated. This will provide a close check to verify the level indication in the distributed control system (DCS) panel.
- When any tank reaches a hand tape gauged level of 40 feet, the main tank block valve will be closed, and it should be reported to higher management.
- When two tanks are blocked in because of high levels, notify higher management and the oil shift coordinator for immediate action.

B. *The booster pump module*

For starting up the booster pump (BP), the following steps should be performed.

- Close the discharge on/off valve; V-1783.
- Open the suction on/off valve; V-1782.
- Wait for the EST to show ready to start permissive.
- Clear all active alarms in the DCS.
- Reset the pump from the DCS.
- If the reading of the pressure transmitter (PI-1411) is not within high and low limits, it is considered as a fault detection problem that will be tackled in our future work otherwise the following steps should be carried out.
- Put the minimum flow-recycle control-valve (FCV-1401) on the BP in Remote-Set-Point (RSP) mode.
- Adjust the Set-Pint (SP) of the FCV-1401 to 280 MBD (55%) (by the operator); For industrial reasons we cannot give the description of the abbreviation MBD.
- Start the pump by pressing the respective ‘Start’ push-button (PB-1724A) in the field.
- Put the FCV1401 valve in auto mode to keep the discharge pressure between (100–115) psig that is indicated by the PI-1408 after running the pump.

C. *The shipper pump*

For starting up the shipper pump (SP) normally, perform the following steps.

- Wait for ready to reset signal.
- Press reset switch (HS-1738) from the distributer control system, DCS.
- The pump ready to start (indicator, XL-1738A, shall be light).
- Fully close the on/off discharge valve; V-1792.
- Fully open the booster discharge valve; V-1783.
- Fully open the suction valve; V-1790.
- If the reading of the PI-1428 is within high and low limits, the pump can be started, (100 – 115) psig.
- Put the minimum flow-recycle control-valve (FCV-1409) on the shipper pump in remote setpoint, RSP mode.
- Adjust the set point of the FCV-1409 to (10%).
- Start the shipper pump and open the discharge valve; V-1792 concurrently with the operation of this pump.
- Set the recycle valve FCV-1409 in auto mode.

D. *The flow control valve module, FCV*

The operation of the FCV starts from the normal process conditions that consist of the shuttle line pressure PC-1025 device. If the reading of this device is greater than the set point (550 psig), the pressure override loop PC1358OV is in auto mode while the master flow controller FCV1358 is in automatic control. The controlled valves HC1358/1359/1360, are in auto mode with control output provided by the master flow controller. This process depends on the flow reading transmitters that can be clarified as follows.

If the 56" shuttle line master flow controller FC1358 deviation (process variable (PV)-setpoint) is greater than + 25% , the rupture detection logic that places the FCV in direct digital control (DDC) mode to close the HC1358/1359/1360 controlled valves is activated. To restore from rupture detection, the operator has to change the HC1358/1359/1360 mode to manual or auto mode for normal operation of the shuttle line.

Also, the master controller and pressure override action of the 48" shuttle line is similar to the 56" shuttle line controller described above. In this case, the pressure override loop PC1358OV activates the pressure valves if the difference between the set-point and its process variable (PV) is very low. In this case, the problem is switched from flow to pressure problem (low pressure). As long as steady state operation exists, the shipper pump operates reliably. However, if the stabilizer trip affects the rundown line flow and pressure, the booster pumps would trip. The booster pump trip on rundown line upset would be minimized if pressure drop is compensated by the higher level in the TFU-2.

E. *The pressure reducing station*

The FCV module discussed in subsection-D is designed for flow control problem; the pressure reducing station (PRS) is designed for high pressure problem. The latter comprises four pressure control valves (PCV-952-1, PCV-952-2, PCV-952-3 & PCV-452). These control valves are sensed by the four-shuttle line transmitters. These transmitters’ are:
Inputs: The shuttle line-A (56") pressure transmitters are PT-1024 and PT-1406 respectively. The shuttle line-B (48") pressure transmitters are PT-1025 and PT-1405 respectively.
Output: The control valves are PCV-452-1, PCV-952-2, PCV-952-3, and PCV-452.
The high selected reading (pressure) from these four transmitters will be used as process variables (PVs) for the PRS control process described below.

- Get the four transmitters readings
- Select the highest reading out of the four transmitters
- Compare the highest reading (PV) with the set points
- If the PV > 610 psi then use the control valve PCV-452 only to reduce pressure.

- If the PV > 620 psi then use the control valves PCV-452, and PCV-952-1 respectively.
- If the PV > 630 psi then use the control valve PCV-452, PCV-952-1, and PCV-952-2 respectively.
- If the PV > 640 psi then use the control valve PCV-452, PCV -952-1, PCV-952-2, and PCV-952-3 respectively.
- Vice versa when pressure goes down
- Alarm will be set when the pressure goes higher than 700 psi and emergency shut down EST procedure maybe takes place.

Note that the PV is always the high selected value from the four transmitters only when the pressure reading is within the operating range. If any transmitter is faulty, below or high its range, or put on maintenance by the DCS operators, then it will be de-selected from the control and the next highest reading will be selected as an input (PV) to the PRS controllers. These staggered set points will adjust the pressure release to TFU-2 and will provide smoother control response for shuttle line's pressure. For example, when the shuttle line pressure rises to 610 psi, only PCV-452 will act to release the pressure to TFU-2. If the shuttle line pressure continues to increase then PCV-952-1&2 & 3 will open to release the pressure as described above.

F. Developing a Parallel Petri Net Model of the OPP

In recent years, a large number of hybrid system modeling schemes have been proposed [21]. These schemes can be grouped into three classes. The first class consists of the extensions of continuous schemes by the introduction of discrete variables; ordinary differential equations with Boolean variables. The second class comprehends discrete schemes where new elements are introduced for representing the continuous dynamics. It incorporates continuous places and transitions of a Petri net used for modeling such dynamics. The third class is stemmed from Petri nets as graphical tools that are particularly considered because of their well-known power for representing process features such as concurrency, conflict, and synchronization features [27].

This paper borrows the modeling scheme proposed in [14] to model the OPP. This modeling scheme follows a modular bottom-up modeling strategy. It builds binary interpreted Petri net (IPN) (see definition-9) model to represent the behavior of each component shown in table II that is formed based on the collected information in the previous subsections. This modeling method can be described as follows.

Let us use a simple component in the OPP named valve # 54 (V54) that has normal state (opened/closed) or faulty state ("Error On Open" or "Error On Closed). The methodology can be summarized as follows.

For each system there exist a set of component that can be defined in (4).

$$X = \{x_i \mid x_i \in X\} \tag{4}$$

where *i*=1,2, …*n*, and *n* is the number of system components. x_i is the valve # 54.

For each system component, x_i, the different variables needed to represent its behavior must be chosen. The finite set of state variables (*Y*) can be defined as:

$$Y = \{y_j^i \mid y_j^i \in Y\} \tag{5}$$

where *i*=1,2, …*m*, and *m* is the number of state variables. This set of variables is associated to the system component $x_i \in X$ must be built with the existing of at least one state variable for each system component; $Y \neq \{0\}$ i.e. y_j^i represents the state of the valve # 54.

For each state variable $y_j^i \in Y$, there exists a set of possible values that can be defined as:

$$Z = \{z_k^{ij} \mid z_k^{ij} \in Z\} \tag{6}$$

where *k=1, 2, ...* is the number of positions e.g. the set of values of the variable y_j^i is $z_k^{ij} = \{val_1^{ij}, val_2^{ij}, \ldots, val_P^{ij}\}$; in this case, the valve position may take four values: "Open", "Closed", "Error On Open" and "Error On Closed".

The values in each set z_k^{ij} must be represented in terms of IPN markings. For each state variable $y_j^i \in Y$ there exist a set of places $P_{y_j^i} = \{P_1^{ij}, P_2^{ij}, \ldots, P_P^{ij}\}$□. The marking of these places is binary and mutually exclusive; two processes are mutually exclusive if they cannot be performed at the same time due to constraints on the usage of shared resources. In this case, $P_{y_j^i} = \{P_1, P_2, P_3\}$ where *P₁* and *P₂* represent the state of the normal state of the valve and *P₃* reprints the faulty state of the valve.

For each pair of values val_n^{ij} and val_m^{ij} such that the state variable y_j^i could change from val_n^{ij} to val_m^{ij} a transition T_{nm}^{ij} must be created. Then one arc going from place p_n^{ij} to transition T_{nm}^{ij} and one arc going from T_{nm}^{ij} to place p_m^{ij} must be created as shown in Fig. 3. The initial marking of the net shown in the figure is defined as: $M_0(P_m^{ij}) = 1$ if the initial variable y_j^i is val_n^{ij} and $M_0(P_m^{ij}) = 0$. In our case, the selected valve # 54 (V54) can be represented by the marking vector, $M_0(P_m^{ij}) = [1\ 0\ 0]^T$. This valve is depicted in Fig. 3 as a part of Fig. 4. Notice that the faulty place, P_{F1}, is eliminated from Fig. 4 and also for each module and will be added in our future work (Fault detection problem).

The output of this algorithm is a set of isolated IPN modules similar to the valve #54 built above. This modeling algorithm models each component of the OPP. Table-II lists a part of these components. This paper performs synchronous composition and permissive among the IPN modules developed to build the parallel IPN model of the OPP shown in Fig. 4. Merging the developed modules is not strait forward. In other words, a set of transitions can be merged and the other sets may be deleted. Merging or deleting is based on the analysis methodology discussed in subsection (V-A) that uses the structural and behavioral properties of Petri nets as will be detailed. In the developed model depicted in Fig. 4, the places, P_{34}-P_{48} , are used for priority condition. For example, the valve V58 should be opened after the valve V54. This case is controlled by the place, P_{34}.

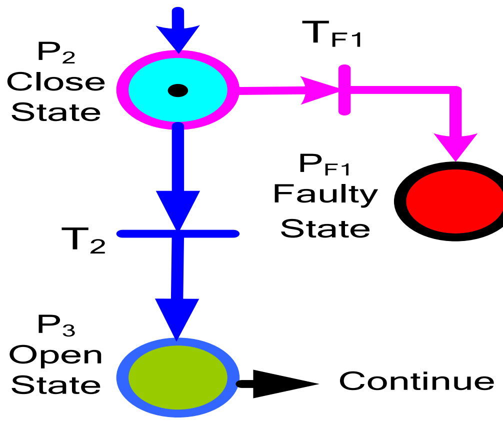

Fig. 3. Petri net module of the valve # 54 (V54)

TABLE II
THE RECIPE OF THE OPP

Place	Associated action	Tr.	Associated event
-----	-----	-----	-----
P_2	The valve V54 is ready	t_2	Start opening the valve V54
P_3	The valve V54 is opened	t_3	Start opening the valve V58
-----	-----	-----	-----
P_{11}	The valve V-1783 is closed	t_{11}	Start the pump by pressing PB-1724A
P_{12}	The valve V-1783 is opened	t_{12}	Put (FCV-1401) in remote-set-point (RSP) mode
P_{13}	The valve V-1782 is ready	t_{13}	Adjust the set-point of the FCV-1401 to 280 MBD (Auto)
-----	-----	-----	-----
P_{29}	The shipper ump is started		
P_{30}	The valve V-1790 is ready		
P_{31}	The valve V-1790 is opened		
P_{32}	The FCV-1409is ready		
-----	-----	-----	-----

In the developed model depicted in Fig. 4 there also are several continuous modules e.g. the flow control and pressure reducing units that have many throttling valves. Unfortunately, these valves have no models in any mathematical or graphical description. You may requisite a valve based on its sizing but not its model. This leads to some difficulties to complete the OPP Petri net model. Accordingly, this paper develops models of these valves using intelligent systems as described in subsection (IV-G). In the IPN-based parallel model of the OPP depicted in Fig. 4, when the transition T_{18} is fired one of its output is to activate the FCV and the PRS. The former controls the flow while the latter control the pressure of the shuttle lines. A part of the FCV Petri net module is depicted in Fig. 5 and its corresponding states and transitions description is shown in table III. These units are continuous in nature so modeling and controlling their valves should take place in section (IV-G). Keep in mind, the places P_{FV15}, P_{FV17}, P_{FV16},, and P_{FV18} represent the state of a throttling valves, while P_{FV13} and P_{FV14} act for controlling these valves.

TABLE III
THE RECIPE OF THE FCV DEPICTED IN FIG. 5

Place	Associated action	Tr.	Associated event
-----	-----	-----	-----
P_{FV7}	The reading of the value of the valve PC1025 (PV1)	T_{FV5}	Read the value of PC1025 (PV1)
P_{FV8}	The reading of the value of the flow transmitter (FT) (PV2)	T_{FV6}	Read the value of FT (PV2)
-----	-----	-----	-----
P_{FV11}	The value x1 is	T_{FV9}	Calculate the difference (PV1- SP1 = x1)
P_{FV12}	The value x2 is	T_{FV10}	Calculate the difference (PV2- SP2= X2)
P_{FV13}	Activating the PC-1358 OV controller (Pressure problem indicator)	T_{FV11}	If x2 high and x1 low
P_{FV14}	Activating the FC-1358 controller (Flow problem - indicator)	T_{FV12}	If x2 low and x1 high
-----	-----	T_{FV13}	Start to close the valves HC1358/1359/1360
P_{FV19}	Go back to read the value of process variable #1; PV1	-----	-----
P_{FV20}	Go back to read the value of process variable #2; PV2		
-----	-----		

G. Modeling the continuous valves of the FCV

In the OPP, manipulated control variables are usually the flow rate of the fluid and its pressure. These variables are controlled using the flow and pressure valves (HC1358/1359/1360) that are related to the FCV and PRS. Among of those continuous valves the HC1360 valve is selected to be modeled and controlled using intelligent systems in this paper. The input to this valve is a 3-15-psi air-pressure signa1 (0%-100%) which is applied to the top of the diaphragm.

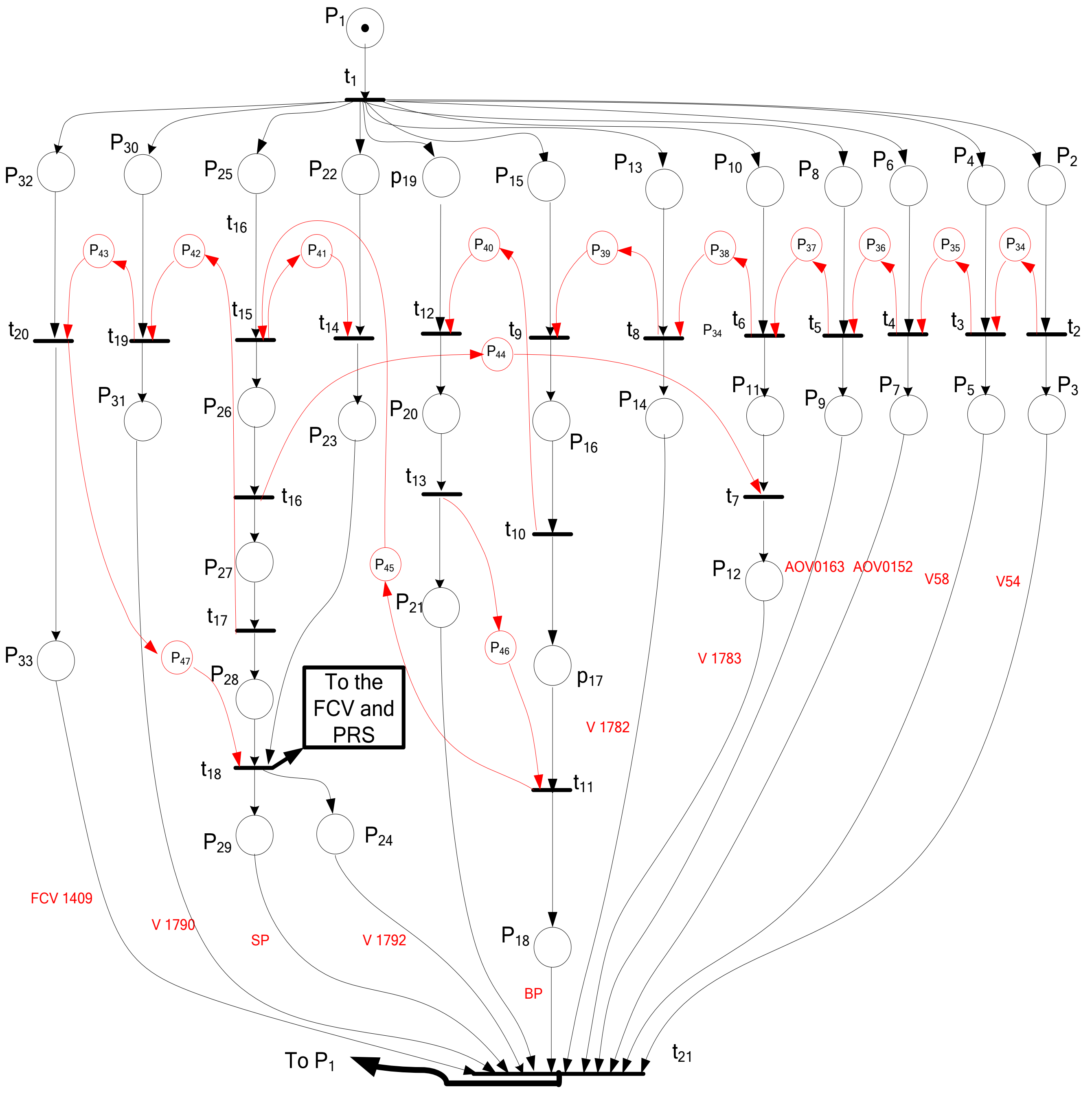

Fig. 4. The parallel Petri net model of the OPP

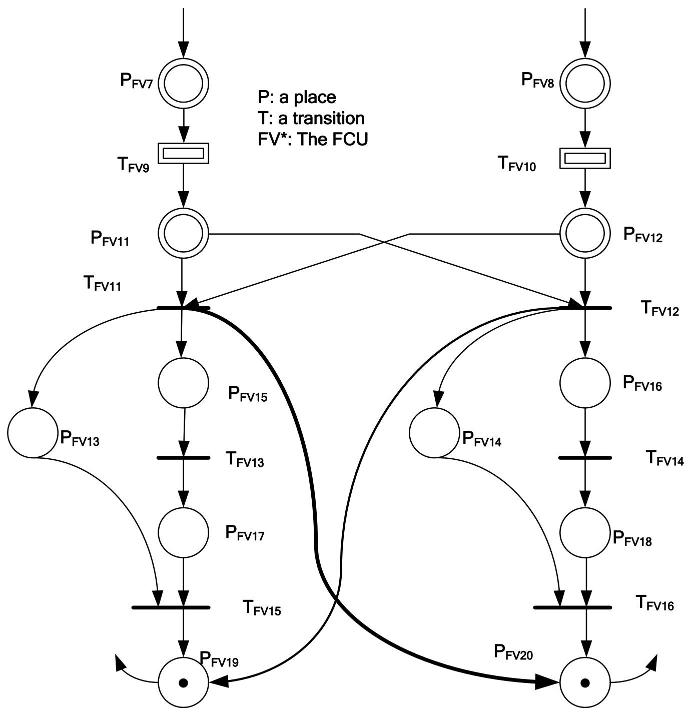

Fig. 5. The developed FCV PN-module of Fig. 4

The diaphragm actuator converts the air pressure into a displacement of the valve stem. The valve body and trim varies the area through which the following fluid must pass. In our problem, the valve represents a continuous module that should be represented with few places shown in Fig. 5 inside the hybrid Petri net model shown in Fig. 4. Controlling of this module, subjects to the availability of its model. Actually, there is no model of such valves in the OPP. This is the main reason for modeling such continuous modules. Once a reliable model is obtained of such continuous valves, it can be controlled at firing the transition T_{FV11} shown in Fig. 5 and consequently the flow and pressure of the shipper pump (P_{29}) depicted in Fig. 5 is regulated. The valve can be modeled using a particle swarm algorithm (PSA) (See the appendix-A).

For simplicity, let us develop an Auto Regressive with eXogenous input ARX model of the HC1360 valve. The parameters of this valve can be coded into a particle (solution) listed in the appendix-A as follows:

$$P = [a_1 \quad b_1]^T \qquad (7)$$

where i=1, 2, 3, n, and n is the order of the ARX model. The PSA with the inertia weight defined in (A2) and (A3) is used to obtain the best ARX model of this valve. Its parameters $\pi_{\max}$, $\pi_{\min}$, c_1 and c_2, are set to 0.98, 0.15, 1.45, and 1.95 respectively. Initially, 99 individuals (swarm=99) are randomly generated in a population and the evolution phase is processed for 10000 generation. Two experiments (n=1, and 2) were performed to get the best ARX model of the valve using the PSA. Real data are collected and used for this valve. The input-output training set consists of 99 input-output pair in the percentage range [1, 100] as listed in table IV. The RMS errors defined in (8) were computed for these experiments and were 0.03 (n=1), and 0.157 (n=2) respectively.

$$RMS = \sqrt{\frac{1}{T}\sum_{k=1}^{T}(y_d(k) - y(k))^2} \qquad (8)$$

$$f_i(P_i) = \frac{1}{RMS} \qquad (9)$$

where i=1,2,. . . N, $y_d(k)$, $y(k)$ are the desired output and the actual output of the ARX model at sample k , and T is the number of samples. The practical results show that the best harmony between the real valve ant its ARX model optimized using the PSA is obtained at n=1 $P = [a_1 \quad b_1 \quad b_2]^T$ and it can be defined below.

$$y(k+1) = 0.65\, y(k) + 0.74\, u(k) - 0.38\, u(k-1) \qquad (10)$$

Using another input-output training set that consists also of 99 input-output pair in the percentage range [1, 100], Fig. 6 depicts the actual and desired outputs of the HC1360 valve for 100 testing samples.

TABLE IV
REAL VALUES OF THE FLOW VALVE

Sample #	Input %	Output %
---------	---------	---------
3	6.50	3.00
4	10.00	11.40
5	10.50	12.50
---------	---------	---------
43	95.00	95.00
44	95.40	95.40
45	96.00	95.00
46	99.60	99.00
---------	---------	---------
92	10.50	11.00
93	5.50	10.50
94	2.60	10.00
95	0.00	5.40

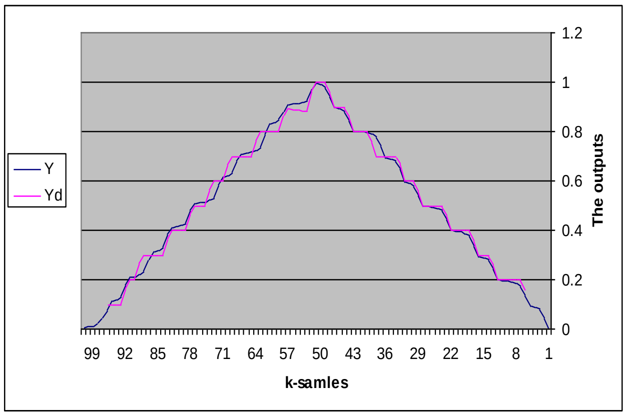

Fig. 6. The outputs of the real valve and its model

V. Simulation Results

This section analysis the parallel Petri net model of the OPP developed in section IV. It also simulates the developed model using Petri net software tool version 2.1.

A. *The OPP Petri net model analysis*

Given a system Petri net model and its specification there are two classes of problems to be considered; the *analysis* and *synthesis* problems. The former asks whether or not the model satisfies the specification. Solving this problem involves identifying *sufficient* or *necessary and sufficient* tests for satisfiability of the specification with respect to the assumed model (a hybrid system). Necessary and sufficient tests are often referred to as *verification* tests whereas only sufficient conditions are often referred to as *validation* tests. The latter problem of interest concerns the synthesis of controllers that enforce a specification on the plant behavior. Synthesis is closely related to analysis in that by parameterzing the verification or validation problems, it may be possible to effectively search for system parameters ensuring the satisfiability of the specification. Synthesis is out of this paper scope while the analysis problem, verification and validation is detailed below.

The verification methodology is performed in the sense of topology analysis (net class, traps, and siphons), behavioural analysis (coverability, and liveness), structural analysis (boundness, safeness, conservativeness, repetitiveness, and consistency), and invariant analysis (P-invariant). Among these properties, this analysis assures the most important ones and tries to enhance the PN-models to develop reliable PN-model of the OPP. Behavioral properties (reachablity, liveness), and structural properties (boundness, safeness, invariants) are the most important feature for a PN-model (some of these properties are briefly described in section II). The necessary and sufficient conditions of this verification method are performed using the MATLAB Petri net tool. They can be confirmed using the invariant properties; place-invariant (P-invariant), of the model developed. The P-invariant test satisfies the given recipe or specification by reconfiguring the PN-model developed for the process to be modeled. It is a set of places whose weighted token count remains constant for all possible marking vectors. The set of places are represented by *n*-dimensional integer vectors *X*, where *n* is the number of places of the Petri net; non-zero entries correspond to the places that belong to the particular invariant.

A place invariant is defined as every integer vector, *X* that satisfies:

$$X^T M_0 = X^T M \tag{11}$$

Equation (11) means that the weighted sum of the tokens in the places of the invariant remains constant for all reachable markings, and this sum is not depending on the initial marking of the Petri net; it is a structural dependent. The place invariants of a net can be computed by finding integer solutions to:

$$X^T C^n = 0 \tag{12}$$

where C^n is the *n x m* incidence matrix of the Petri net. Given any firing vector *q*,

$$\begin{aligned} X^T M(k+1) &= X^T\left(M(k) + C^n q(k)\right) \\ &= X^T M(k) \ \textit{iff} \ \ X^T C^n = 0 \end{aligned} \tag{13}$$

This is a very important feature that is the weighted sum of the tokens in the places of the invariant remains constant for all reachable markings as mentioned above. In general the weight associated with a place may be any integer, but usually one is mainly interested by P-invariants whose weights are positives. The set of places having a weight not null in such an invariant is a conservative component. Using this method, the developed parallel model shown in Fig. 4 can be verified and modified using the MATLAB-based PN-tool software ver. 2.1. The developed model should be P-invariant model to assure that the number of tokens in the model do not exceed certain value in a set of places. The value in each place in our application should be one to assure that the developed model is 1-bounded and consequently safe.

This paper verifies the developed parallel model shown in Fig. 5 using the Petri net tool software that enables us to redesign the developed module to satisfy the sufficient and necessary conditions described above. After reconfiguring the developed OPP PN-model, the results obtained are: Its net topology is marked graph (definition 6), reachable (definition 3), and bounded (definition 4). The latter can be confirmed using the P-invariant property as follows. Based on the simulation results obtained using the Petri net software tool version 2.1, the minimal-support P-invariants m-rank (A)=27 => at most 27 P-invariants are linearly independent. One of these P-invariants is: $X_1=[0\ 1\ 1 \dots\ 0\ 1]^T$ corresponding to the places *(P_2, P_3, P_{47})*. The initial marking vector M_0 of the parallel model depicted in Fig. 4 is: $M_0=[1\ 0\ \dots\ 0]^T$. Firing the transition T_1 gives the marking vector $M_1=[0\ 1\ 0\ 1\ 0\ 1\ \dots\ 0\ 0]^T$ that satisfies the $X_1^T M_1$ equals one. This operation is continued for all firing sequences. In each case the $I_i^T M_i$ equals one. This satisfies the relation (11) that means that the developed model is bounded. Although the developed module is 1-bounded, there is no guarantee for safeness property for many industrial applications. Accordingly, sufficient test is performed to assure the safeness property (see definition 4) described below.

The model can be validated using the PN-tool and the real information collected and formed in table II . The marking vector is $M=[P_1, P_2, \dots, P_{47}]^T$, and the input or firing vector is $q=[T_1, T_2, \dots, T_{21}]^T$. Let us start from firing t_1; the firing input is $q[t_1]=1$, and the initial marking vector is $M_0 = [1, 0, \dots, 0, 0]^T$, the new state after firing the transition t_1 is: $M_1=[P_2, P_4, P_6, P_8, P_{10}, P_{13}, P_{15}, P_{19}, P_{22}, P_{25}, P_{30}, P_{32}]^T$. Further, no transition can be fired except the transition t_2 as depicted in Fig. 4. The resulted state vector at firing input $q[t_2]=1$is: $M_1=[P_3, P_4, P_6, P_8, P_{10}, P_{13}, P_{15}, P_{19}, P_{22}, P_{25}, P_{30}, P_{32}]^T$. This means that the valve V54 is fully opened and the valve V58 is ready to open after V54. This assures the boundness and safeness of the developed parallel model. Also, simulation results show that the developed model identical to the given recipe shown in table II that shows the real information obtained from the field of the OPP.

B. *Firing the OPP Petri net model*

This paper employs the OPP Petri net model developed in Fig. 4 to show how the unified frame work that includes continuous and discrete modules is carried out. The marking vector of the Petri net model of the OPP can be represented with $M_{OPP}==[P_1, P_2, \ldots, P_{47}]^T$ and the input or firing vector is $q=[t_1,t_2, \ldots, t_{21}]^T$. Let us starting from firing t_1; the firing input is $q[t_1]=1$, and the initial marking vector $M_0=[1,0,\ldots,0]^T$, the new state after firing the transition t_1 is: $M_1=[P_2, P_4, P_6, P_8, P_{10}, P_{13}, P_{15}, P_{19}, P_{22}, P_{25}, P_{30}, P_{32}]^T$. According to the recipe shown in table II, the new state is start opening the outlet-block-valve-54 *(P_2)*, start opening the outlet-block-valve-58, *(P_4)*, and so on. The reader can fire the subsequent transitions and locate the new state description from table II. The process of firing is carried on to the transition t_{18}; the firing input in this case is $q[t_{18}]=1$, and the new state after firing the transition t_{18} is: $M_1=[P_3, P_5, P_7, P_9, P_{12}, P_{14}, P_{18}, P_{21}, P_{24}, P_{29}, P_{33}]^T$. This state can be described as follows:

- All valves corresponding to the places indicated in this firing vector M_1 are fully opened
- The booster pump (BP) is in its operation mode *(p_{18})*.
- The shipper pump (SP) is also in its operation mode *(p_{29})*.
- The necessary valves e.g. V1792 is fully opened and the FCV-1409 is in auto mode

The most important point in this firing sequence at firing the transition t_{18} is the activation of the FCV or the PRS. The former works when the flow is too low, while the latter works when the pressure is too high. Both of these two modules have a set of continuous valves that are working in a continuous mode although they are represented with few places (discrete representation) inside the hybrid model developed in a unified frame work. This is virtue of using Petri net as a graphical modeling tool for industrial processes.

V. CONCLUSIONS

Numerous shutdowns due to Ad hoc approaches used leads to the loss of billions of dollars in oil industry. This paper introduced a parallel Petri net model of the OPP. This model includes continuous and discrete modules that are modeled using intelligent systems and Petri nets respectively. Regarding the former, this paper investigated the idea of representing a continuous module with few places inside the parallel Petri net model developed. This representation enables us to model hybrid industrial systems e.g. the OPP easily and completely. It also reduces the scan time required by companies due to the matrix manipulation feature of Petri nets that are highlighted for modeling industrial oil processes instead of other graphical tools. This recommendation is based on the following reasons. First, Petri nets have good power of mathematics. Second, they are transparent, and can capture all the dynamic features of the process to be modeled such as concurrency and conflict that are difficult to achieve by most available graphical tools. Third, unlike automata, many PN based software modules are published recently: Mitsubishi, Schneider Automation Group, Siemens, Fatek, Omron, ABB Automation, Allen Bradley. Out of these modules, Siemens modules are commonly used in oil industry and are recommended by this paper.

There are a variety of degrees of functional and structural similarities between earlier control schemes, Petri nets, and automata. These similarities are important for better understanding and explanation of the process to be modelled and supervised using these tools. Although the Petri nets have some superiority to automata, they do not give details for the system to be modelled. This leads to some difficulties to find optimal solutions in search space of Petri nets. Developing automata models and swapping these models using Petri nets is our future work. The former gives details of the process to be modelled that would be helpful for analyses and finding optimal solutions in search space of automata, while the latter compresses these details in new modules to help the engineers to design and follow the behavior of the process to be modelled and supervised.

APPENDIX

A. *Appendix-A*

No one can gauge that the PSA or Least squares algorithm is the best for the ARX model, but we can say that each of them has its own philosophy. The latter is one of the gradient-based optimization methods. Basically, these gradient optimization methods usually suffer from local optima problem. On other hand, the former is one of the intelligent evolutionary algorithms that can avoid the local minima problem. The PSA has a good memory compared with other evaluation optimizers such as genetic algorithms. It is used in many optimization problems e.g. parameters learning of neural networks, modeling and control schemes. Although least squares algorithms are employed in similar problems, PSA has the superiority to least squares in the sense of its speed to find the global solution. This speed inherited virtue of its memory that remembers the best locations was visited.

Generally, systems can be modeled in the form of auto regressive moving average models (ARMA) that are classed into three types: ARMA, ARMA with additional eXogenous input, and Auto Regressive with eXogenous input (ARX). These models can be described as follows: First, the stochastic systems can be modelled as Auto Regresive Moving Average (ARMA) model defined below:

$$A(z^{-1})y(t) = C(z^{-1})e(t) \qquad \text{(A1)}$$

where

y(t) is the system output,

e(t) is a random white noise,

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + . \quad . \quad . + a_n z^{-n}$$

$$C(z^{-1}) = 1 + c_1 z^{-1} + c_2 z^{-2} + . \quad . \quad . + c_r z^{-r}$$

Second, models of the stochastic systems with additional eXogenous input (u(t)) can be defined as ARMAX model:

$$A(z^{-1})y(t) = B(z^{-1})u(t) + C(z^{-1})e(t) \qquad \text{(A2)}$$

where

$$B(z^{-1}) = b_o + b_1 z^{-1} + b_2 z^{-2} + . \quad . \quad . + b_m z^{-m}$$

Third, ignoring noise term in model (2) results the Auto Regresive with eXogenous input (ARX) model. In this paper the ARX model of the industrial valve are developed using the PSA as detailed in subsection IV-G.

When hybrid systems are modeled using Petri nets, continuous modules may initiate a problem if they have no models. For this reasons, this paper employs the PSA to model a continuous part inside hybrid systems. This algorithm is employed to obtain the Auto Regressive with eXogenous (ARX) model; linearized model, of a continuous activity inside the hybrid systems. The definite structure of the ARX model is difficult to obtain for complex processes. Accordingly, the harmony between the process to be modeled and its ARX model is not guaranteed. So, this paper suggests that the ARX model of a process should be initially structured with minimal parameters (first order) that are optimized with the PSA. Then, the harmony between the process be modeled and its ARX model is evaluated using a certain performance index. If the harmony between the process and the ARX model obtained is quit good (the error between their outputs is minimum), then the modeling process is terminated. Otherwise, the order of the ARX model is gradually increased and the optimization process using the PSA is continually performed until a good harmony between the process and its ARX model is achieved. The overall optimization process of the ARX model by the PSA is as follows:

Coding: A floating point coding scheme is employed in this paper. In the ARX model, the parameters *n* and *m* are the number of *a*th parameters, and *b*th parameters respectively. Thus, the parameters can be coded into a particle as follows:

$$P = \left[A_p \;\; B_p\right]^T \qquad \text{(A3)}$$

where, A_p, B_p, are vectors that are given by:

$A_p = \left[a_1\, a_2 \ldots a_n\right]^T$, and $B_p = \left[b_1\, b_2 \ldots b_n\right]^T$

Initialization: In this stage, a population of particles with random positions and velocities was initialized. The position of each particle was assigned to the values of the parameters to be optimized and the velocity was assigned to the change of these parameters as defined in (8), and (9). For the purpose of generality, the input/ output domain of the ARX model is normalized in the range [-1, 1], then the parameters are randomly initialized within [-1, 1].

$$V_{id}(t+1) = \Pi V_{id}(t) + c_1 r_1 \left(PL_{id}(t) - P_{id}(t)\right) + c_2 r_2 \left(PL_{gd}(t) - P_{id}(t)\right) \qquad \text{(A4)}$$

$$P_{id}(t+1) = V_{id}(t+1) + P_{id}(t) \qquad \text{(A5)}$$

where PL_{gd} *(t)* is the best particle in the swarm; *d=1, 2, . . ., i=1,2, . . . , N* , (*N* is the size of the swarm); Π is a weight, c_1 & c_2 are positive acceleration constants, and r_1 & r_2 are uniformly distributed random numbers in the range [0,1]. The weight Π is defined as: $\Pi = \frac{\pi_{\max} - \pi_{\min}}{t_{\max}} t$. where $\pi_{\min}$ is the initial weight, $\pi_{\max}$ is the final weight, and $t_{\max}$ is the maximum iteration number. Equation (A4) describes how the velocity of a particle is dynamically updated while equation (A5) updates its position.

***Evaluation*:** For each particle, the fitness value is calculated. In this paper, the fitness function that measures the quality of each particle is given by the reciprocal of the Root Mean Square (*RMS*) errors defined in (8).

Adaptation: In this stage, the best position for each particle, *PL*, and the best position swarm, *Pg*, are calculated. The original PSA algorithm and the PSA algorithm with the inertia weight are employed here to adapt the position and the velocity of each particle. This process is carried out until the best parameters (solutions) are obtained for the process to be modeled.

B. Appendix-B

The design procedure of PNs and LDs are borrowed from [28] to clarify the superiority of the former to the latter. In this example, the condition of a product is checked when the product enters a test zone in a transport line. If a defected product is found, it will be expelled; otherwise it will pass through and go to the next station.

The control recipe for the test zoon is listed in Table B-1 and its PN model shown in Fig. B-1. In this figure, five places (P_1 to P_5) represent the progressive states for operating the test zoon, while six transitions (T_1 to T_6) decide the sequence of these states. Three actions depicted in Table B-3 are activated in state P2 to P5 respectively, to perform the external actions.

TABLE B-1. THE RECIPE OF THE TEST ZOON IN A TRANSPORT LINE

Place #	Description	Transition #	Description
P_1	waiting for a coming production (A3)	T_0	power on
P_2	trigger the defect detector (A1)	T_1	production in position
---	---	---	---
P_5	deactivate the defect detector (A1)	T_6	production has left

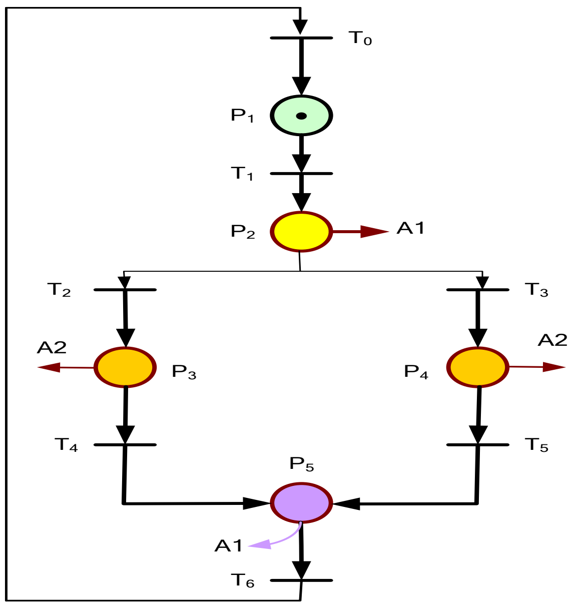

Fig. B-1. A PN model of the test zoon in a transport line

The descriptive tables of events and places for the PN model shown in Fig. B-1 are listed in Table B-2 and Table B-3 respectively. The converted LD of the PN model and corresponding actions are depicted in Fig. B-2 and Fig. B-3 respectively.

TABLE B-2. AN EXTERNAL CONDITION TABLE

Transition #	Address	Description
T_1	0000.00	Product in position
T_2	0000.02	Detect fail
---	---	---
T_6	0000.05+ 000.06	Deactivate the defect detector

TABLE B-3. AN EXTERNAL CONDITION TABLE

Coil #	Actions #	Switch ON	Switch OFF	Description
0001.00	A1	P2	P5	Activate defect detector
0001.01	A2	P3	P4	Set the position of the expelling rod
0001.02	A3	The others	P1	Turn on the busy indicator

This example shows how LD gets more complicated in sense of its rungs used to represent the PN-model. For complex systems, these rungs are increased in an inconvenient configuration that leads to several bugs during run time. This insure the superiority of PNs to other graphical tools.

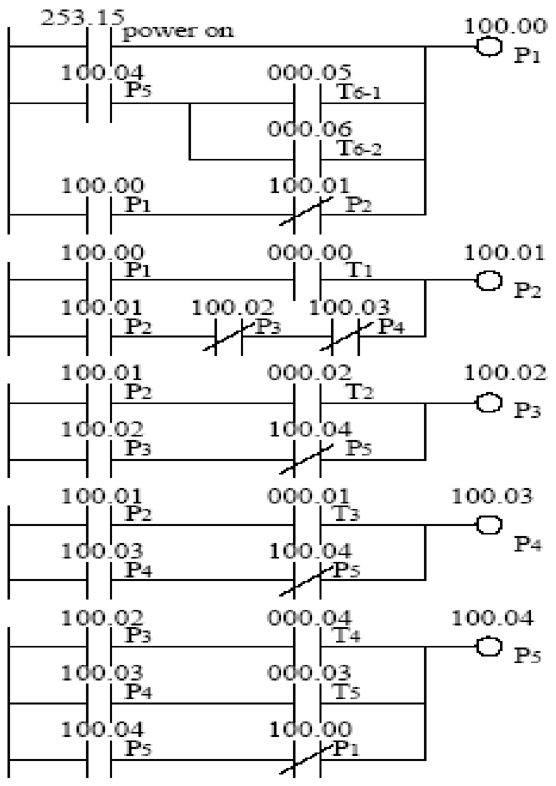

Fig. B-2. Converted LD for sequential control

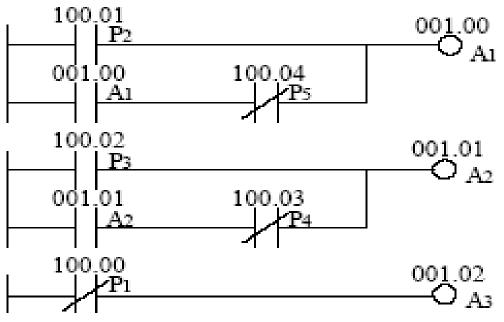

Fig. B-3. Converted LD for output actions

References

[1] X.-R. Cao and Y.-C. Ho, "Models of discrete event dynamic systems." *IEEE Control Systems Magazine*, vol. 10, 69–76, 1990.

[2] R. David, "Grafcet: A powerful tool for specification of logic controllers," *IEEE Transactions on Control Systems Technology*, vol. 3, pp. 253-268, 1995.

[3] T. Murata, "Petri Nets: Properties, analysis and applications," *Proceedings of IEEE*, vol. 77, pp.541-580, 1989.

[4] R. David and H. Alla,, (2005). *Discrete, Continuous, and Hybrid Petri Nets*, ISBN: 3-540-22480-7, Springer-Verlag, Berlin Heidelberg.

[5] I. Demongodin and N.T. Koussoulas, "Differential Petri Net Models for Industrial Automation and Supervisory Control," *IEEE Transactions on Systems, Man, And Cybernetics-part c: Application and Reviews*, vol. 36, pp. 543-553, 2006.

[6] M.F. Russo, "Modeling, analysis, simulation, and control of laboratory automation systems using Petri nets analysis and control," *The Association for Laboratory Automation, JALA*, vol. 13, pp. 103-115, 2008. [DOI: 10.1016/j.jala.2007.12.008].

[7] D. Andreu, J.C. Pascal, H. Pingaud, and R. Valette, "Batch process modeling using Petri nets," *IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, USA, pp.314-319, 1994. [DOI: 10.1109/ICSMC.1994.399857].

[8] S.U. Guan and F. Zhu, "An incremental approach to genetic algorithms based classification," *IEEE Transactions on Systems, Man, and Cybernetics*, Part-B, vol. 35, pp. 227-239, 2005.

[9] P. Daihee and K. Abraham, "Genetic-based new fuzzy reasoning models with application to fuzzy control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, pp. 39-47, 1994.

[10] W.K. Lennon and K.M. Passino, "Genetic adaptive identification and control," *Engineering Applications of Artificial Intelligence*, vol. 12, pp. 185-200, 1999.

[11] P.J. Anitha, C. Vijila, and D. Hemanth, "Comparative analysis of genetic algorithm and particle swarm optimization techniques for SOFM based abnormal retinal image classification," *International Journal of Recent Trends in Engineering*, 2:143-145, 2009.

[12] C.H. Chen and J.H. Dai, "Design high-level synthesis of hybrid controller," *Proceedings of the IEEE International Conference on Networking, Sensing, and Control*, Taipei, Taiwan, vol. 1, pp. 433-438. [DOI: 10.1109/ICNSC.2004.1297477], 2004.

[13] M.V. Iordache and P.J. Antsaklis. *Supervision Based on Place Invariants: A Survey*, Technical Report of the ISIS Group at the University of Notre Dame ISIS-2004-003, 2004.

[14] J. Arámburo-Lizárraga, A. Ramírez-Treviño, E. López-Mellado, and E. Ruiz-Beltrán, (2008). *Fault Diagnosis in Discrete event Systems using Interpreted Petri Nets*, Advances in Robotics, Automation and Control, Book edited by: J. Arámburo and A. Ramírez-Treviño, ISBN 78-953-7619-16-9, pp. 69-84, I-Tech, Vienna, Austria.

[15] B. Hrúz and M.C. Zhou, (2007). *Advanced Textbooks in Control and Signal Processing: Modeling and control of discrete-event dynamic systems*, ISBN-13: 9781846288722, Springer-Verlag London Limited.

[16] M.C. Zhou and E. Twiss, "Design of industrial automated systems via relay ladder logic programming and Petri nets", *IEEE Transactions on Systems, Man, and Cybernetics - Part C*, vol. 28, pp. 137–150, 1998. [DOI: 10.1109/5326.661096].

[17] R. David and H. Alla, "Petri Nets for Modeling of Dynamic Systems :A survey, *Automatica*, ISSN:0005-1098, vol. 30, pp. 175-202, 1994. [DOI bookmark: 10.1016/0005-1098(94)90024-8].

[18] W. Bolton, (2006). *Programmable logic controllers.* ISBN-13: 978-0-7506-8112-4, W. Bolton. Published by Elsevier Newnes.

[19] P. Zhang, (2008). *Industrial Control Technology: A handbook for engineers and researchers*, ISBN: 978-0-8155-1571-5, William Andrew Inc.

[20] E. Villani, P.E. Miyagi, and R. Valette, "Landing system verification based on Petri nets and a hybrid approach", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, pp. 1420-1436, 2006.

[21] E. Villani, P.E. Miyagi, and R. Valette, (2007). *Advances in Industrial Control: Modeling and analysis of hybrid supervisory systems*, ISBN: 978-1-84628-650-6, Springer-Verlag London Limited.

[22] T. Sobh and B. Benhabib, "Discrete Event and Hybrid Systems in Robotics and Automation: An overview", *IEEE Robotics and Automation Magazine*, pp.16-19, 1997.

[23] W.-L. Yao, H.-T. Liao, S.-C. Chi, and S.-S. Peng, "A Petri net based offline simulation and online diagnostic platform for manufacturing systems", *Journal of the Chinese Institute of Industrial Engineers*, vol. 22, pp. 64-75, 2005.

[24] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time Petri nets," *IEEE Transactions on Software Engeering*, vol. 17, pp. 259–273, 1991. [DOI Bookmark: http://doi.ieeecomputersociety.org/10.1109/32.75415]

[25] A.T. Sava and H. Alla, "A control synthesis approach for time discrete event systems," *Mathematics and Computers in Simulation*, vol., 70, pp. 250–265. [DOI: 10.1016/j.matcom.2005.11.001]

[26] W.M. Wonham, "Supervisory control of discrete event systems," *Systems Control Group*, Dept. of Electrical and Control Engineerin, University of Toronto, ECE 1636F/1637S, 2007.

[27] M.M. Gomaa, **H.A. Awad**, and A.R. Anwar, "Design and implementation of supervisory control schemes in industrial automation systems", *The 2008 International Conference on Computer Engineering & Systems* (ICCES'08), pp. 389-404, Cairo, Egypt, 2008. [DOI: 10.1109/ICCES.2008.4773035].

[28] J-L. Chirn and D.C. McFarlane, "Petri Nets based Design of Ladder Logic Diagrams", *Submitted to Control 2000*, Cambridge, UK.

Dr. Hamdi Ali Ahmed Awad received the B.Sc. degree in Industrial Electronics, and the M.Sc. degree in Adaptive Control Systems from Faculty of Electronic Engineering, Minuf, Minufiya University, Egypt in 1988 and 1994 respectively.
Dr. Awad received the Ph.D. degree in Artificial Intelligent Systems in 2001 from School of Engineering, Cardiff University, Cardiff, Wales, England.
Dr. Awad has got Associative Prof. degree in Computer Sciences and Control Systems in 2007. Since August 2007, he has been with the Faculty of Electronic Engineering, Minufiya University, Egypt.
Dr. Awad interests in intelligent control systems and their applications in control systems, Mechatronics, Petri nets, Discrete event systems, Hybrid systems, Fault detection & isolation, and Medical engineering.