v. 3, n. 34, 2023

Scientific Journal of Applied Social and Clinical Science

PETRI NET APPLICATIONS IN SYSTEMS

Karen Hernández Rueda
Information Systems Department
``Universidad de Guadalajara``
https://orcid.org/0000-0002-7209-2907

Abstract: Petri nets are a tool that has been booming mainly in flexible manufacturing engineering and its processes, to perform modeling, control and diagnosis of systems. This work presents a description of Petri nets with their mathematical and graphic representation to introduce the tool and facilitate its understanding, and then establish a general overview of its applications as a result of the respective bibliographic review. It also provides examples as part of the results of its application. Finally, it is concluded that it can be incorporated into various areas where complex systems or processes are used, and even as part of a teaching tool.
Keywords: Petri nets, system modeling and simulation, software development, software verification, concurrency

INTRODUCTION

Currently, most of the systems that are developed are complex because they integrate various functionalities, for example, Industry 4.0 requires the design of embedded and event-driven systems, the Internet of Things considers comprehensive systems in the home to control various devices at home from cell phones, different communication networks (local, metropolitan and extensive) can be accessed from computing facilities where applications are deployed to perform distributed processing of tasks or interact with various information systems in a distributed manner, among other things. The part of the occurrence of events that can consider parallelism, concurrency, asynchrony or sequence, are part of many complex and distributed systems. Therefore, the tools that consider these aspects become important in the design, modeling and simulation of these systems, and Petri nets represent an option to be applied to these needs (HADDAD and YAKOLEV, 2014), since they are useful for the modeling, analysis and simulation of complex, distributed, workflow systems, among others.

This contribution presents an overview of Petri nets and their properties, and at the same time, presents them as an area of opportunity as an alternative application in various areas that have an impact on society, whether for process improvement, optimization, control or diagnosis of systems, such as teaching topics.

The next section introduces Petri nets with their mathematical and graphical notation, as well as the types of Petri nets to model different systems. Applications are then presented as part of the results of the literature review and their applications are shown. Finally, the conclusions are presented.

METHODS AND MATERIALS

The research methodology was exploratory with a documentary approach that consisted of the review of various bibliographic to establish the theoretical framework, definition and classes of Petri nets, as well as learn about their different applications. Given the limitation of space, only some of the applications that could be interesting were selected.

Petri networks or Petri Nets (PNs) were proposed by Carl A. Petri in 1962 as part of a computing model to simulate concurrent tasks in distributed and complex systems, particularly in the area of systems analysis and verification (JIMÉNEZ, 2022)., then they were applied in flexible manufacturing, in computer networks, processes, communication protocols among other areas.

PNs have a mathematical and graphical representation que se usa para representar Sistemas de Eventos Discretos which are dynamic systems that function through events in discrete intervals (JIMÉNEZ, 2022). Systems can be distributed, concurrent, parallel, simultaneous events or with multiprocesses, among others (DÍAZ, 2013). Below, the notation of the PNs and their different classes

is presented. An interested reader can consult the references (DESSEL and ESPARZA, 1995) and (MURATA, 1989) for more information.

PETRI NETS

A PN according to (RUIZ-BELTRAN, RAMÍREZ-TREVIÑO and OROZCO-MORA, 2014) is represented by two types of nodes (figure 1): circles, which represent places (p_1, p_2, p_3) and are associated with actions or outputs of the system to be modeled and bars or rectangles (t_1, t_2) that represent transitions and are associated with events and actions.

The places and transitions are connected by arcs, both are finite and non-zero. An arc connects from a place to a transition or from a transition to a place, so an arc has a node at each of its terminations and this refers to identifying input transitions or output transitions associated with a place of particular entry or exit into the network. A place can have an integer (positive or zero) of marks. The number of marks (x) at a location i (p_i) is represented as $M(p_i)$=x and an initial mark in the network $M0(p_i)$ =x would be an initial distribution of marks (black dots or number in the places), in figure 1 the initial marking is $M_0(p_1)$ =1. The presence or absence of a mark at a location can indicate whether a condition associated with this location is true or false. At some point the marking defines the current state of the system modeled by the PN. The evolution of the state corresponds to the evolution of the marking caused by the firing of transitions.

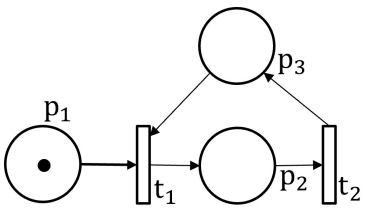


Figure1. Petri net. Own source.

The PN is used to simulate the dynamic behavior of a system, the marking on a PN is changed as it evolves. A transition can only be triggered if each of the input locations of this transition contains at least one flag, and the transition is said to be triggerable or enabled. Triggering a t_x transition consists of removing a mark from each of the input places of that transition and adding a mark to each of the output places of that transition. When a transition is enabled it does not imply that it will be triggered immediately, but it maintains the possibility of triggering it. The transition trigger is considered to have zero duration. The formal definition of PN is presented below and was taken from (HERNÁNDEZ, MEDA and MARTÍNEZ, 2018).

Definition 1. A structure of a PN is a bipartite digraph defined by the 4-tuple N=(P,T,I,O), where $P = \{p_1,p_2,...,p_n\}$, $T = \{t_1,t_2,...,t_m\}$ are finite sets of places and transitions respectively. $Q \cup T \neq \emptyset$ *and* $P \cap T = \emptyset$. *I:* $P \times T \rightarrow \{0.1\}$ *and O:* $P \times T \rightarrow \{0.1\}$ *are* the input and output functions that describe the arcs that go from places to transitions and from transitions to places respectively.

Definition 2. A marking is a function M: P→ {0,1,2, 3, ...} that assigns each location a non-negative integer, named after the number of markings residing within each location. M_0 is the initial marking distribution.

Definition 3. A PN is an N structure along with an initial marking, this is denoted as (Q, M_0).

The n×m incidence matrix C of N is defined by C{i,j}= O(tj,pi)−I(pi,tj). The notation •t={p|I(p,t)≠0}, t•={p|O(p,t)≠0},•p={t|O(p,t)≠0} and p •={t|I(p,t)≠0} represents the entry and exit locations of t, and the entry and exit transitions of p respectively.

A transition tj is said to be enabled in marking M_k if it has $M_K(p_i) \geq I(p_i,t_j)$ markings at each input p_i location to t_j. An enabled transition t_j can be triggered, removes $I(p_i,t_j)$ marks from pi and adds $O(t_j,p_k)$ marks to p_k producing a new mark M_k+1 (represented by MkMk+1) which can be calculated using the equation state $M_{k+1} = M_k + C$ where C is the incidence matrix and (i) =1 if i=j and (i)=0 otherwise.

Note that M0M1 can be extended to a sequence of transitions M0Mq where $\sigma = t_a t_b ... t_r$. In this case M_q is said to be reachable from M_0. The reachability set of N, denoted by R (Q, M_0), is the set of all possible markings reachable from M_0, triggering only enabled transitions.

INTERPRETED PETRI NETS

An Interpreted Petri Net or Interpreted Petri Net (IPN) is an extension of PN, according to (HERNÁNDEZ-RUEDA and MEDA-CAMPAÑA, 2015) based on the input and output signals of the system being modeled. Entries are associated with transitions and exits with places.

The event and condition are associated with the transition. Graphically, an IPN is presented in Figure 2. The measurable locations $\{p_1, p_3\}$ of the IPN are unfilled circles while the non-measurable location {p2} is a filled circle. The same thing happens with transitions, but instead of indicating whether they are measurable or not, they are said to be controllable or not.

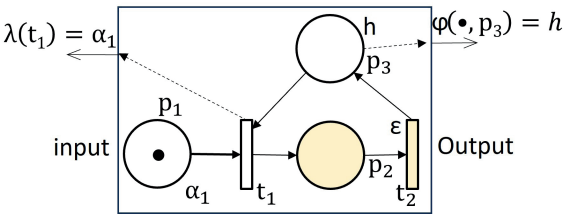


Figure2. Interpreted Petri Net. Own source.

The formal definition of PN is presented below (RUIZ-BELTRÁN. RAMÍREZ-TREVIÑO, and OROZCO-MORA, 2018), (RAMÍREZ-TREVIÑO et al., 2012).

Definition 4: An IPN is the 4-tuple Q = (G, Σ, λ, φ) where: N = (G, M0) is a PN. Σ = {α1, α2,..., α_r} is the input alphabet of the network, where αi is an input symbol. λ: T → Σ∪{ε} is a transition labeling function, it assigns an input symbol or null symbol to each transition, with the following restriction:∀*tj, tk*∈*T, j* ≠ *k*, Yeah∀pi I(pi, tj) = I(pi, tk) ≠ 0 and so λ(tj) ≠ ε, λ(tk) ≠ε, then λ(tj) ≠ λ(tk). In this case ε represents an uncontrollable event of the system. φ: R (Q, M0) → {Z+}q is a function that assigns an output symbol to each reachable mark of the IPN, where q is the number of places that have been assigned the signal of a sensor h of the system.

Definition 5: a transition ti is controllable if λ(ti)≠ε otherwise it is uncontrollable. Likewise, a place pi is measurable if it has a signal from a sensor assigned to it, otherwise it is not measurable.

COLORED PETRI NETS

A Colored Petri Net or Colored Petri Net (CPN), according to (OJO et al., 2018) is a network in which the marks do not represent objects but a type of data such as Boolean or integer, among others. The colors represent the data contained in the marks. It has some characteristics such as its simple graphical representation for people not familiar with PNs, it shows the control and use of data, CPNs can be created by connecting other smaller CPNs that can resemble subroutines, procedures and programming modules, they are general and can describe processes, communication protocols, software and hardware systems, among others. The formal definition is presented in the following line.

Definition 6: A CPN is a nine-element tuple CPN= (Σ, P, T, A, N, C, G, E, I) where:

Σ is a finite non-empty set of colors called a color set, which specifies the colors for each mark, P is a finite set of places that can contain colors and T is a finite set of transitions that can modify the color of P such that P∩T = ∅, A is a finite set of arcs that join P and T so that N is the node function: $N: A \rightarrow (P \times T) \cup (T \times P)$ $y \ P \cap T = P \cap A = T \cap A = 0$, C is color function (set of colors a P) C: P → Σ, G is a guard function (guards T), E expression function of the arc that is evaluated as a color of a set of colors of P or as a set multiple of colors, and I is an initialization function (initial marks of P) of P to closed terms such that ∀p ∈ P: [Type $(I(p)) = C(p)_{MS}$].

Definition 7: ΣMS is a multiset of all colors. Let m ∈ ΣMS, with m: S → N0 where S returns the occurrence of its elements in Σ, e. i., s∈ St t. q. s∈ Σ sii m(s)>0.

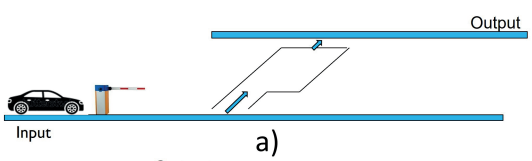
The function G maps each transition t∈ T to an expression of g and excludes some elements.

RESULTS AND DISCUSSION

The review of the literature shows the usefulness of using PNs to meet new needs to model or simulate various systems, which can be complex, distributed or represent software or hardware. The use of PN, IPN or CPN can facilitate the solution of various problems or avoid errors when modeling systems. Below are applications of networks in some systems.

APPLICATION IN SYSTEM MODELING

Systems can be modeled with PNs, and possible failures can be modeled. A simple example of a system is represented in Figure 3.



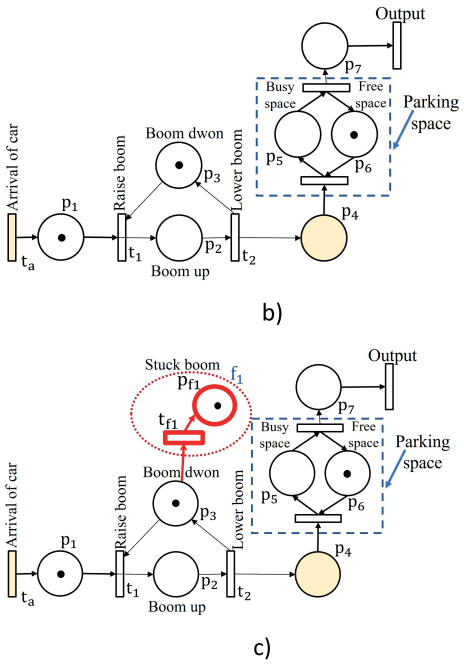


Figure3. Capacity parking one and its model in RP. Own source.

Figure 3a represents a parking lot that has an entrance, an exit and a parking space. Figure 3b shows its model in IPN, the parts to highlight are the arrival section, the entrance section where the pen is displayed with the states of being down or up, the drawer that can be free or occupied, and the respective output. In Figure 3c, a possible fault (tf1 and pf1 represent the fault) highlighted in red is considered, in which the boom gets stuck, that is, it fails to rise when a car wants to enter, this particular fault is called permanent, if they happen it causes the system to stop because they do not allow cars to access the parking lot if they are not repaired. The fault is modeled with a transition and a place, since if it happens the mark that causes this part of the system to stop is lost. T2 is considered a pre-risk transition (before the fault occurs) and t1 is considered a post-risk transition (after the fault occurs), both transitions are used to determine the risk zone, for example: the area in which the fault is likely to occur. The IPN can be simulated in various tools depending

on its characteristic and modeled systems, for example, HPetriSim, PetriNet or MATLAB, PetrA, HiPS, TAPAAL, among others.

APPLICATION IN FAULT DIAGNOSIS

When subsystems are added to the systems that are modeled to help perform fault diagnoses and these are also represented in PN, depending on the approach used, then we have a diagnostic application. Diagnosis involves determining the occurrence of a fault in a finite number of steps with information from the input and output of the system. For example, in (RAMIREZ-TREVINO et al., 2012) a reduced diagnostician approach was proposed with a place and several transitions that are constructed if the system is diagnosable, and is based on the incidence matrix C and the function output ϕ of the system (RUIZ and OROZCO, 2009). Figure 4 (HERNÁNDEZ-RUEDA and MEDA-CAMPAÑA, 2014) shows a simple example of its application. Figure 4a shows an example of a system modeled in IPN that may have a fault (represented by tf1 and pf1). Figure 4b is the diagnostic for that system. Another approach, for example, uses two places with initial flags that are added to the IPN network so that a diagnosis can be made and it does not matter that the system is not diagnosable.

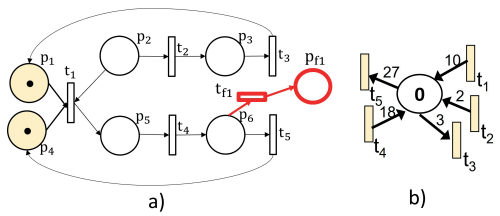


Figure 4: A system modeled in IPNs and its respective diagnostician. Source (HERNÁNDEZ-RUEDA and MEDA-CAMPAÑA, 2014)

Another approach in (HERNÁNDEZ-RUEDA, MEDA-CAMPAÑA and ARÁMBURO-LIZÁRRAGA, 2015) uses two sites as a subsystem, which are added to transitions in the IPN that have common entry sites (Cr regulation circuit) to make a diagnosis. of the system. Figure 5 shows a part of a system that is not diagnosable because transitions t4 and t5 can continue to trigger, even if the fault occurs.

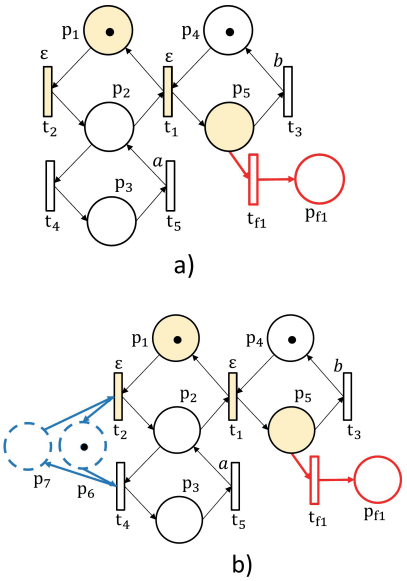


Figure 5: IPN with non-diagnosable a) and diagnosable fault b). Source (HERNÁNDEZ-RUEDA and MEDA-CAMPAÑA, 2015)

In figure 5a the IPN models a system that is not diagnosable because a part of the system can continue working, even if a fault has occurred (represented by tf1 and pf1), but in figure 5b the IPN is made diagnosable with Cr (represented by p6 and p7) because it forces the system to stop completely to check if a fault occurred. Another version of this approach (HERNÁNDEZ-RUEDA et al., 2021) considers that the behavior of the system is not limited with the use of more marks in the Cr.

APPLICATION IN SOFTWARE DEVELOPMENT

According to (BAQUERO, ARGOTA VEGA, and RODRÍGUEZ, 2016) the use case helps to understand the way a system behaves, as well as obtaining the user's requirements, and that is useful for analysts. Therefore, use case diagrams are a technique to determine requirements of a system or obtain information about how a system works and considers actors (an entity, something with some behavior, person, etc.) and use cases. In the diagram, the actors and cases are part of the set of places P, the intermediate subprocesses or state changes are represented as a set of transitions T, and the relationships that exist between the actors and use cases are integrated into a set defined as α and β. The figure shows the representations used to represent the symbologies of the use case diagrams. If you want to test the use case diagram in the PN, you follow its evolution of states that represent its behavior. The set of places must be greater than 1 because the diagram does not exist with just an actor or a use case.

The initial marking must have m rows that will be equal to the number of places, each row will have the value of 1 that is associated with the actors that initialize the system, and zero otherwise.

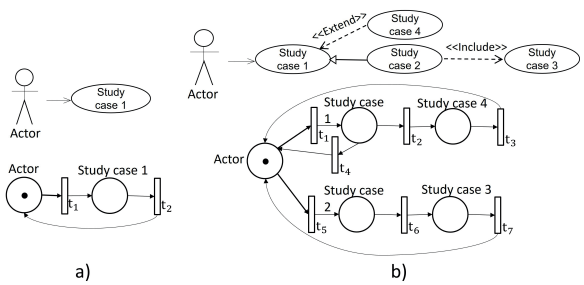


Figure 6: Relationship between the use case diagrams and their symbology in PN. Based on (BAQUERO, ARGOTA VEGA, and RODRIGUEZ, 2016).

The marking of the network for two consecutive iterations will be different Mk≠ Mk+1 and a place must not remain marked for two consecutive iterations of the same, if it occurs it is considered that the actor or use case is not correctly related to the others, and they must also be checked at least once. An example is the representation of an online printing system, Figure 6 represents the use case diagram and Figure 7 the system modeled in PNs.

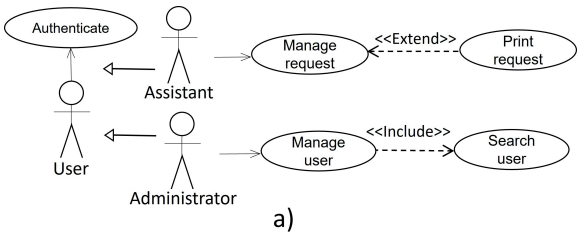


Figure 7: Use-case diagram of an online impressions system. Source (BAQUERO, ARGOTA VEGA, and RODRIGUEZ, 2016).

The representation meets the established conditions, so the method proposed by the authors allows errors to be detected in the modeling of the use case diagrams.

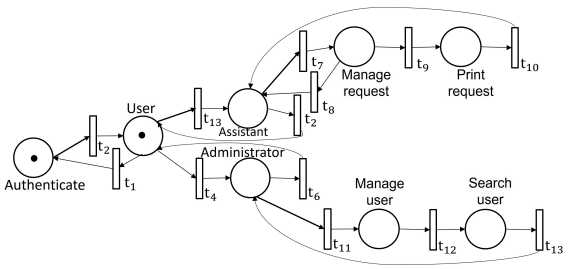


Figure 8: PN of Figure 6. Source (BAQUERO, ARGOTA VEGA, and RODRIGUEZ, 2016).

Likewise, the PNs can be used to test the activity diagram (BAQUERO et al., 2015) as shown in figure 9, figure 9a represents the diagram and figure 9 the PN that models the diagram.

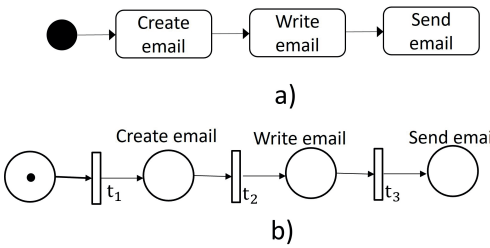


Figure 9: Diagram of activities a) and its model in PN b). Source (BAQUERO et al., 2015)

In the activity, the absence of an end point is identified and that helps avoid errors in the analysis and design of the system's behavior. In software development, particularly in distributed and concurrent systems, as indicated in (CABAC, HAUSTERMANN, and MOSTELLER, 2018), PNs are used together with technologies such as RENEW or The Reference Net Workshop, which is a simulation engine and editor for PNs, and the PN-based agent-oriented software engineering approach technique (PAOSE) that depends on MULAN (Multi-Agent Nets). RENEW is used for the development of applications based on PNs and in particular high-level PNs are used as a programming language. Figure 10 shows an example of modeling a Web server that implements an HTTP protocol.

On the other hand, CPNs have also been used in software development, in some cases to simulate information sending and processing processes (BETZ et al., 2014).

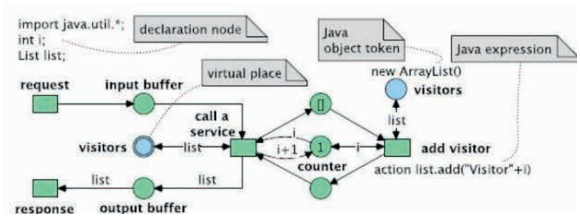


Figure 10. Reference network modeling a Web server with Java registrations and PNs. Source (CABAC, HAUSTERMANN, and MOSTELLER, 2018).

For example, Figure 11 shows an example of simulation screens carried out with CPN Tools technology, of a device to detect student fingerprints to take their respective attendance, which was modeled in CPN and presented in (OJO et al., 2018).

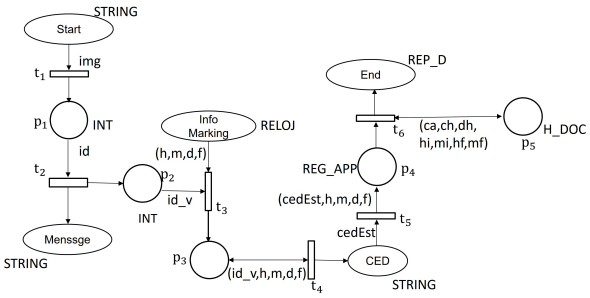


Figure 11. CPN for the proposed device to detect fingerprints. Based on (OJO et al., 2018).

Unlike the other models, each arc is identified with information and in this case, some were omitted because programming instructions were indicated. The diagram is a simplified representation that indicates the data types such as string (string) or int (integer), some variables such as id, id_v, h, m, etc., and colors such as closet REP_D (product STRING*INT *INT*STRIN*STRING), REG_APP, among others. The process begins when the fingerprint is placed on the device that is identified as img, which is verified in a database (t1) that returns an id to know if it is within an allowed range (t2), and sends a message if it is correct or not. The id goes to a verification with id_v that has data related to the time, day, and marked date, in addition to CLOCK data (t3), and it goes to the REG_ARD color (p3) that saves a new record. The fingerprint cards are saved in different variables ced (t4), which is sent to cedEst (t5), and the new data that is recorded (p4) to be compared (t6), with the timetable and subject (p5) with those stored (p5). This is simulated to evaluate performance.

CONCLUSIONS

Since their inception, Petri nets have had a great impact on applying them to various systems, since in addition to modeling them, with different methods that have been proposed, they allow simulating behavior and establishing proposals for improvements to automate systems, control or diagnose them, among others. things. However, one class of PN is not used, there are several, and it depends on their characteristics to represent, model and simulate systems that have a specific application and a specific area (VENTRE, MICOLINI and MICOLINI, 2023) (MORALES, 2015), (BERNARDINELLO, 2014), (LAKOS, 2009). Although at the beginning it was mainly applied in the computing area, for example, at the end of 1980 there were many publications on applications of the use of PNs, particularly in the verification of software properties (MURATA, 1989), they were extended to other areas. for its potential. On the other hand, it has been shown how PNs can be used to improve the representation of a system with diagrams made with Unified Modeling Language (UML), as well as to carry out the respective tests. Furthermore, the (extended) RENEW framework integrates modeling techniques, means for generating code from abstract models, and means for monitoring and control. This has gone from being a simple simulator and editor of a high-level Petri net formalism to becoming an integrated development environment for applications based on Petri nets and Java (CABAC, HAUSTERMANN, and MOSTELLER, 2018). PNs have had a boom in recent years, and as mentioned in (JIMÉNES, 2020), many Industrial Engineering programs from foreign universities for operations research, decision making and systems simulation. Therefore, they are a highly recommended application option.

REFERENCES

BAQUERO HERNÁNDEZ, L.R., ARGOTA VEGA, L. E and RODRÍGUEZ VALDÉS, O. (2016). Método para el modelado y prueba de Diagramas de Casos de Uso mediante redes de Petri. *Revista Cubana de Ciencias Informáticas*, *10*, 138-149.

BAQUERO HERNÁNDEZ, L.R., ARGOTA VEGA, L. E, RODRÍGUEZ VALDÉS, O. (2016), CIUDAD RICARDO, F.A. (2015). Método para el modelado y prueba de Diagramas de Actividades mediante redes de Petri. *Revista Latinoamericana de Ingeniería de Software 3*(5), 206-212.

BERNARDINELLO, L., KILINC, G., MANGIONI, E., POMELLO, L. (2014). Modeling Distributed Private Key Generation by Composing Petri Nets. In: Koutny, M., Haddad, S., Yakovlev, A. (eds) Transactions on Petri Nets and Other Models of Concurrency IX. *Lecture Notes in Computer Science*, vol 8910. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-45730-6_2

BETZ, T., CABAC, L., DUVIGNEAU, M., WAGNER, T., WESTER-EBBINGHAUS, M. (2014). Software Engineering with Petri Nets: A Web Service and Agent Perspective. In: Koutny, M., Haddad, S., Yakovlev, A. (eds) Transactions on Petri Nets and Other Models of Concurrency IX. *Lecture Notes in Computer Science*, 8910. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-45730-6_3.

CABAC, L., HAUSTERMANN, M. and MOSTELLER, D. (2018). Software development with Petri nets and agents: Approach, frameworks and tool set. *Science of Computer Programming*, *157*, 56-70. https://doi.org/10.1016/j.scico.2017.12.003

DENSEL J. and ESPARZA J. (1995). Free Choice Petri Nets. University Press. Cambridge.

DÍAZ, M. (Ed.). (2013). *Petri nets: fundamental models, verification, and applications*. John Wiley & Sons.

HADDAD, S., and YAKOVLEV, A. (2014). Petri Nets and Other Models of Concurrency IX LNCS 8910. Springer New Yoork Dordrecht London DOI 10.1007/978-3-662-45730-6

HERNÁNDEZ-RUEDA, K. and MEDA-CAMPAÑA, M.E. (2014). Necessary Condition to make a correct diagnoser of faults. *Memoria Electro, 36*, pp.6-11.

HERNÁNDEZ-RUEDA, K. and MEDA-CAMPAÑA, M.E. (2015). Fault Diagnosis in Discrete Events Systems Not Diagnosable. *Memoria Electro, 37*, pp.141-146.

HERNÁNDEZ-RUEDA, K., MEDA-CAMPAÑA, M.E., and ARÁMBURO-LIZÁRRAGA, J. (2015). Enforcing diagnosability in Interpreted Petri Nets. *IFAC-PapersOnline, 48*(7), pp. 56-63.

HERNÁNDEZ RUEDA, K., MEDA CAMPAÑA, M.E. and MARTÍNEZ HARO, B. (2018). Detección activa de faltas en sistemas de eventos discretos. *Pistas Educativas*, *39*(128).

HERNÁNDEZ-RUEDA, K., GONZÁLEZ-CASTOLO, J.C., RAMOS-CABRAL, SILVIA, and MARTÍNEZ-VARGAS, M.P. (2021). Forcing faults diagnosis in K steps in discrete event systems. *ECORFAN Journal-Taiwan, 5*(10), pp. 11- 21. https://doi.org/10.35429/EJT.2021.10.5.11.21

HERNÁNDEZ, L. B., VEGA, L. A., and VALDÉS, O. R. (2015). Método para el Modelado y Prueba de Diagramas de Actividades Mediante Redes de Petri. *Archivo de la revista Latinoamericana de Ingeniería de Software*, *3*(5), 206-212. https://doi.org/10.18294/relais.2015.206-212

JIMÉNEZ SERRANO, E. (2022). Las redes de Petri y su adopción en programas de la Universidad de Sonora. *Epistemus (Sonora)*, *16*(33), 26-32. https://doi.org/10.36790/epistemus.v16i33.221

LAKOS, C. (2009). Modelling Mobile IP with Mobile Petri Nets. In: Jensen, K., Billington, J., Koutny, M. (eds) Transactions on Petri Nets and Other Models of Concurrency III. *Lecture Notes in Computer Science*, vol 5800. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04856-2_6

OJO, K., GONZÁLEZ, Y., CANO, E. E., & ROVETTO, C. A. (2018, August). Modelado del funcionamiento de un dispositivo para el control de la asistencia estudiantil mediante Redes de Petri Coloreadas. In *Memorias de Congresos UTP* (pp.1320).https://revistas.utp.ac.pa/index.php/memoutp/article/view/1836

RAMIREZ-TREVINO, A., RUIZ-BELTRAN, E., ARAMBURO-LIZARRAGA, J. and LOPEZ-MELLADO, E. (2012). Structural Diagnosability of DES and Design of Reduced Petri Net Diagnosers. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 42(2), pp. 416-429. https://doi.org/10.1109/TSMCA.2011.2169950.

RUIZ-BELTRÁN, E., RAMÍREZ-TREBIÑO, A., and OROZCO-MORA, J. L. (2018). Fault diagnosis in Petri nets. *In Formal Methods in Manufacturing* (pp. 27-652). CRC Press, Boca Raton. Editors J. Campos, C. Seatzu, and Xiaolan Xie.

RUIZ BELTRÁN, E. R., and OROZCO MORA, J. L. (2009). Reduced Petri Net Diagnosers for Detecting and Locating Faults. *Conciencia Tecnológica*, (37), 48-53.

MORALES VERAL, A., ROJAS RAMÍREZ, J. A., HERNÁNDEZ GÓMEZ, L. H., MORALES GONZÁLEZ, A, and JIMÉREZ REYES, M.Y. (2015). Modelo de un sistema de producción esbelto con redes de Petri para apoyar la toma de decisiones. *Ingeniare. Revista chilena de ingeniería*, *23*(2), 182-195. http://dx.doi.org/10.4067/S0718-33052015000200004

MURATA, T. (1989). Petrinets: properties, analysis, and applications. Proceedings of IEEE. Vol.77. No.4. Pp.541-580.

VENTRE, L. O., MICOLINI, O., and MICOLINI, A. (2023). Utilización de Redes de Petri para la enseñanza de la concurrencia en la Ingeniería en Computación. *Memorias de las JAIIO*, *9*(11), 19-22.