# FUNDAMENTOS DE TEORIA DE LA COMPUTACIÓN

FACULTAD DE INFORMATICA

UNIVERSIDAD NACIONAL DE LA PLATA

Profesor: Dr. Ramón García-Martínez

GUIA DE ESTUDIO

## REDES DE PETRI

Material: "Murillo Soto, L. 2008. *Redes de Petri: Modelado e implementación de algoritmos para autómatas programables*. Tecnología en Marcha, 21(4): 102-125"

1. Señale usos iniciales de las Redes de Petri.
2. Describa controladores lógicos programables
3. Justifique la importancia de los métodos de diseño formales.
4. Señale la utilidad de las Redes de Petri como formalismo matemático
5. Defina y ejemplifique las redes de Petri clasicas.
6. Defina formalmente una red de Petri.
7. Defina marcado inicial de una red de Petri.
8. Describa como se manifiestan los comportamientos mediante redes de Petri. Describa tipos de reglas de transición.
9. De un ejemplo de reglas de transición.
10. Defina: red de Petri ordinaria, red de Petri pura, transición fuente y transición sumidero.
11. Describa la propiedad dinámica de alcanzabilidad.
12. Describa la propiedad dinámica de Limitable o acotada.
13. Describa la propiedad dinámica de Vivacidad.
14. Describa la propiedad dinámica de Reversibilidad y Estado Inicial.
15. Describa la propiedad dinámica de Cobertura.
16. Describa la propiedad dinámica de Persistencia.
17. Describa la propiedad dinámica de Distancia Sinbcrónica
18. Describa el método de análisis de las propiedades dinámicas conocido como Árbol de Cobertura.

19. Describa el método de análisis de las propiedades dinámicas conocido como Matriz de Incidencia y Ecuación de Estado.

20. Describa el método de análisis de las propiedades dinámicas conocido como Reglas de Reducción.

21. Enuncie la propiedad estática de Vivacidad Estructural.

22. Enuncie la propiedad estática de Controlabilidad.

23. Enuncie la propiedad estática de Limitación o Acotado Estructural.

24. Enuncie la propiedad estática de Conservabilidad.

25. Enuncie la propiedad estática de Repetibilidad.

26. Enuncie la propiedad estática de Consistencia.

27. Plantee la notación de Murata y Pessen para clasificar redes de Petri-

28. Defina Maquinas de Estado.

29. Defina Grafico de Marcados

30. Defina Red de Libre Escogencia.

31. Defina Red de Libre Escogencia Extendida.

32. Defina Red de Escogencia Asimétrica o Red Simple.

33. Desarrolle un cuadro resumen sobre las interpretaciones de una red de Petri.

Murillo Soto, Luis Diego.
Redes de Petri: Modelado e implementación de algoritmos para autómatas programables
**Tecnología en Marcha,** *Vol. 21, N.° 4, Octubre-Diciembre 2008, pp. 102-125*

# Redes de Petri: Modelado e implementación de algoritmos para autómatas programables

*Fecha de recepción: 21/07/08*
*Fecha de aceptación: 05/09/08*

*Luis Diego Murillo Soto*[1]

## Palabras clave

Redes de Petri (PN), Controlador Lógico Programable (PLC), Automatización.

## Key words

Petri Nets, Programmable Logia Controller (PLC), Automatization.

## Resumen

El presente trabajo es una monografía orientada hacia la utilización del formalismo de las Redes de Petri, propuesto por Carl Petri en la descripción de Sistemas Dinámicos de Eventos Discretos (DEDS). Las Redes de Petri, cuyo acrónimo en inglés es PN, fueron utilizadas inicialmente para el análisis de algoritmos en la computación paralela o concurrente, pero dada la complejidad de los procesos productivos actuales, las PN son un método alternativo de diseño tanto para el proceso industrial como para el controlador. En este sentido, este estudio hace una revisión de las referencias bibliográficas donde se indica cómo realizar el modelado y la implementación de algoritmos de control en Controladores Lógicos Programables (PLCs por sus siglas en inglés).

## Abstract

The present work is a monograph oriented to the use of the formalism of Petri networks, proposed by Carl Petri in the description of Discrete Event Dynamic Systems (DEDS). The Petri Nets, whose acronym is PN, were used initially for the analysis of algorithms in the parallel computation or concurrent, nevertheless, given the complexity of the present productive processes, the PN are alternative methods of design as much for the industrial process as of the controller. In this sense, this study makes a revision of the bibliographical references where it is indicated how to make modeled and how to implement of control algorithms in PLCs.

## Introducción

Los controladores lógicos programables (PLCs) son computadoras diseñadas para trabajar en ambientes industriales con la finalidad de controlar una amplia gama

1. Profesor de la Escuela de Ingeniería Electromecánica del Instituto Tecnológico de Costa Rica. Correo electrónico: *lmurillo@itcr.ac.cr*.

de procesos productivos. David, García y Peng [6, 16, 34] mencionan que en el año 1968, la división hidronomática de *General Motors* trabajó en un proyecto para desarrollar un sistema digital programable que permitiera flexibilizar las líneas de producción de automóviles y librarse del mantenimiento que requerían los paneles de relés. Además, la programación del computador industrial debía ser hecha en forma gráfica, simulando los diagramas escalera de los relés, con la finalidad de que la lógica fuera transparente a los técnicos e ingenieros de planta.

La construcción y funcionamiento del PLC es un tema estudiado en la literatura [9, 16, 29, 35] y su desarrollo se ha nutrido de los avances en microprocesadores, memorias y lenguajes de programación, con la diferencia de que el PLC está diseñado para ambientes hostiles donde la humedad, las vibraciones y el polvo son condiciones inherentes de los procesos productivos.

Ciertos procesos productivos industriales corresponden a sistemas dinámicos de eventos discretos (DEDS), los cuales pueden ser maniobrados y/o supervisados por uno o varios controladores de eventos discretos. A nivel industrial, el controlador de eventos discretos (DEC) se implementa usualmente con PLCs. Sin embargo, la obtención del algoritmo de control y su implementación en software es hecha con métodos heurísticos carentes de formalismo, donde la experiencia del diseñador es clave para la obtención de la solución. La figura 1 muestra la interacción entre un proceso industrial descrito como DEDS y su respectivo PLC representado como un DEC.

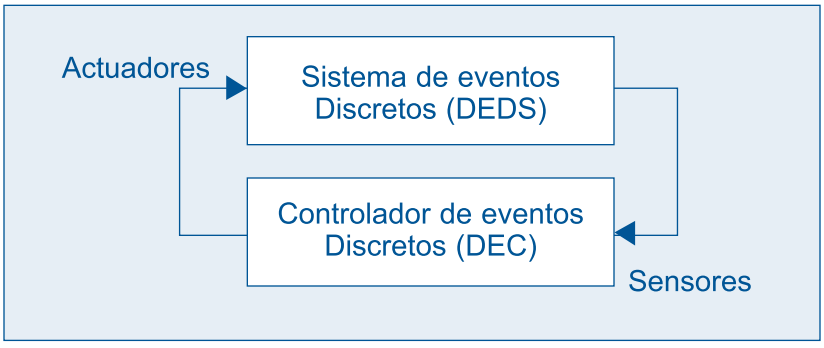



*Figura 1*. Relación entre un DEDS y su controlador DEC.

Existen varios métodos formales para definir algoritmos de control, entre ellos, la teoría de autómatas finitos, la lógica temporal, el lenguaje Z, etc. Sin embargo, estas técnicas no son utilizadas ni aceptadas por fabricantes de PLC para la programación de sus productos. Los fabricantes han estandarizado sus lenguajes en la norma IEC 61131-3 [21], pero dicha norma sólo describe los lenguajes, no las metodologías de diseño. Se ha adoptado el Grafcet como aproximación en la norma IEC 60848 [19], pero ésta carece de partes esenciales en el diseño de algoritmos para controladores, tal como el levantamiento de requerimientos, los criterios de aceptación, el proceso de validación del algoritmo, etc.

La importancia de los métodos de diseño formales, validados y verificados, es que permiten contar con algoritmos de control computacionalmente más eficientes, más seguros y más fáciles de implementar. Dichos métodos formales pretenden resolver los problemas en forma sistemática.

En el área de las ciencias de la computación, surgen en 1962 las Redes de Petri (PN) en la tesis de disertación de Carl Petri. Las PN son un formalismo matemático para el modelado del flujo asincrónico de información [36]. Según Murata [31], las PN se utilizan para el modelado de sistemas que demandan procesos de cómputo concurrentes.

Actualmente, las PN se constituyen en una herramienta fundamental para el modelado de sistemas dinámicos de eventos discretos [4, 23, 24, 25, 33, 39, 40]. El objetivo de este trabajo es determinar referencias donde se desarrolle o utilice alguna familia de PN para la obtención de algoritmos de control para PLC.

Este trabajo se divide en cuatro secciones: la sección dos consiste en una introducción a las PN, su definición, sus propiedades dinámicas, estáticas y su clasificación. La sección tres consiste en una revisión bibliográfica, ordenada cronológicamente, de los artículos que desarrollan algoritmos con PN para PLC. La sección cuatro es una reflexión sobre las tendencias y en qué campos se requiere más investigación. En la sección cinco, se presentan las conclusiones del estudio.

## Redes de Petri

Las Redes de Petri clásicas se conciben como un grafo dirigido que posee dos tipos de nodos principales: los *lugares* representados por círculos y las *transiciones* representadas por barras rectangulares (figura 2). Entre los nodos se ubican los arcos dirigidos, los cuales se encargan de unir las transiciones con los lugares y viceversa. Cada arco dirigido posee un número que indica su peso, el cual determina la cantidad de *marcas* que consume de un lugar o deposita en un lugar, siempre y cuando se haya disparado una transición habilitada. Los arcos dirigidos sin número se entiende que consumen o depositan una marca. Las marcas se representan en forma gráfica como puntos negros que se ubican dentro de cada lugar.

*Las Redes de Preti fueron utilizadas inicialmente para el análisis de algoritmos en la computación paralela o concurrente...*

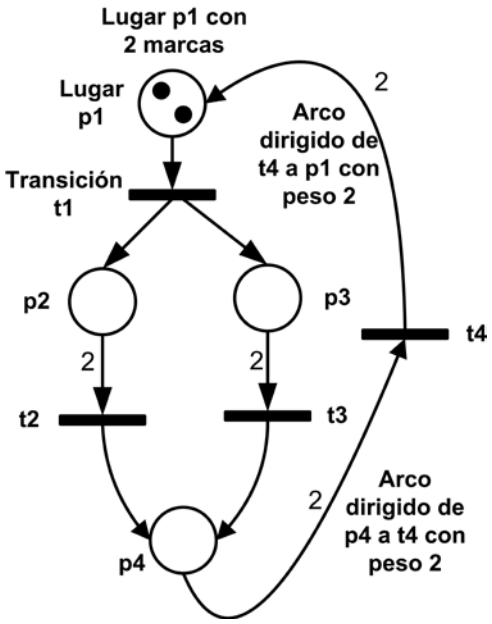



*Figura 2.* Red de Petri con cuatro lugares, cuatro transiciones, cinco arcos dirigidos de peso uno y cuatro arcos dirigidos de peso dos.

Formalmente, una Red de Petri se define como una quíntupla,

$$PN = (P,T,F,W,M_0)$$

Donde:

$P=\{p_1,p_2,\ldots,p_m\}$ es un conjunto finito de lugares.

$T=\{ t_1,t_2,\ldots,t_n \}$ es un conjunto finito de transiciones.

$F\subseteq(P \times T) \cup (T \times P)$ es un conjunto de arcos dirigidos.

$W:F\rightarrow\{1, 2, 3, \ldots\}$ es una función de pesos de los arcos.

$M_0:P\rightarrow\{1, 2, 3, \ldots\}$ es el marcado inicial de la red.

$P\cap T=\varnothing$ y $P \cup T\neq\varnothing$ .

El marcado inicial de una PN son las marcas que posee cada lugar de la red en su inicio. Una Red de Petri con la estructura (P, T, F, W) sin especificar su marcado inicial es denotada por N. Por otro lado, una PN con un marcado inicial dado es denotado por $PN=(N, M_0)$.

El comportamiento de los sistemas puede ser descrito en términos de sus estados y sus cambios. En las Redes de Petri, el estado del sistema, o mejor dicho, el marcado de la PN cambia de acuerdo con las siguientes reglas de disparo o transición:

1. Se dice que una transición es habilitada si cada lugar de entrada *p* de *t* es marcada con al menos *w(p,t)* marcas, donde *w(p,t)* es el peso del arco de *p* a *t*.

2. Una transición habilitada puede o no ser disparada (esto depende solamente del carácter no determinista del evento).
3. El disparo de una transición *t* habilitada remueve *w(p,t)* marcas de cada lugar de entrada *p* de *t* y agrega *w(t,p)* marcas a cada lugar de salida *p* de *t*, donde *w(t,p)* es el peso de los arcos de *t* a *p*.

En la figura 3 se muestra un ejemplo de las reglas de transición. Note que la transición *t* se encuentra habilitada, puesto que existen más marcas en los lugares de entrada que pesos de los arcos dirigidos (figura 3a). Cuando se dispara la transición *t*, existe un flujo de marcas desde los lugares de entrada hacia el lugar de salida (figura 3b).

Las Redes de Petri son nombradas según sus características, por ejemplo, una PN es *ordinaria* si el peso de sus arcos es siempre uno. Se dice que una PN es *pura* si no existen auto bucles. Los auto bucles están definidos como el par lugar y transición, donde el lugar *p* es el lugar de entrada y salida de la transición *t*. Además, existen dos tipos de transiciones adicionales, la transición *fuente* que es aquella que no posee un lugar de entrada y la transición sin un lugar de salida que es llamada transición *sumidero*.

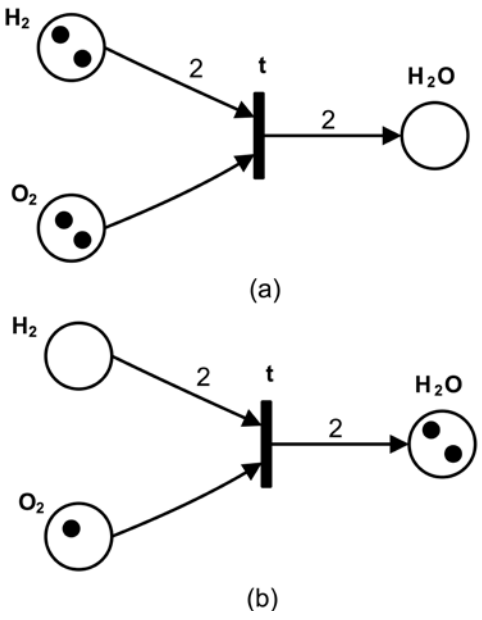



*Figura 3*. Red de Petri donde se muestra la representación de la reacción química $2H_2+O_2 \rightarrow 2H_2O$. a) Antes de la reacción química representada por *t*. b) Después de la reacción. [32]

La fortaleza del modelado de las PN radica en sus propiedades, que se dividen en dos grandes áreas, las dependientes del marcado inicial llamadas propiedades *dinámicas* o *del comportamiento* y las propiedades independientes del marcado, llamadas *estructurales* o *estáticas*.

## Principales propiedades dinámicas

*Alcanzabilidad*: La alcanzabilidad es la principal propiedad dinámica y consiste en que cada disparo de una transición habilitada modifica la distribución de los marcados dentro de la red, de acuerdo con las reglas de disparo. Una secuencia de disparos generará una secuencia de marcados. Se dice que un marcado $M_n$ es alcanzable desde el macado $M_0$ si y sólo si existe una secuencia de disparos que transforme $M_0$ en $M_n$. La secuencia de disparos se denota por sigma:

$$\sigma = M_0\, t_1\, M_1\, t_2\, M_2 \ldots t_n\, M_n$$

En este caso, $M_n$ es alcanzable desde $M_0$ por σ, lo cual se escribe de la siguiente forma: $M_0[\sigma> M_n$. El conjunto de todos los marcados posibles a partir de $M_0$ es denotado por $R(N,M_0)$ y el conjunto de todos los posibles disparos desde $M_0$ es denotado como $L(N,M_0)$. El problema de alcanzabilidad consiste en encontrar un $M_n \in R(N,M_0)$.

*Limitable o Acotada*: una red $PN=(N,M_0)$ se dice limitada si el número de marcas de la red en cada lugar no excede un número finito *k* para cualquier marcado alcanzable desde $M_0$ y existirá dentro de todos los posibles marcados de la red, $M(p) \leq k$,

y $M(p) \in R(N,M_0)$. Se dice que una red es segura si es acotada a uno, esto es si todos los marcados posibles de los lugares poseerán a lo más una marca.

*Vivacidad*: Se dice que una red es viva si no importa cuál marcado halla sido alcanzado, siempre será posible una nueva secuencia σ de disparos y alcanzar un nuevo marcado. Esta propiedad lo que indica es que una red viva garantiza una operación libre de bloqueos. Esta propiedad es deseable en la ejecución de programas.

*Reversibilidad y estado inicial*: De una Red de Petri $(N,M_0)$ se dice que es reversible si para cada marcado $M_n$ existente dentro de $R(N,M_0)$, $M_0$ es alcanzable desde $M_n$. De esta forma una PN reversible es aquella donde siempre es posible alcanzar nuevamente el marcado inicial o estado inicial del sistema.

*Actualmente las PN son un método alternativo de diseño tanto para el proceso industrial como para el controlador.*

*Cobertura*: Un marcado M dentro de una Red de Petri $(N,M_0)$ en un conjunto de marcados cubiertos o contenido, si existe un marcado M' dentro de $R(N,M_0)$ tal que $M'(p) \geq M(p)$ para cada lugar *p* dentro de la Red.

*Persistencia*: Una Red de Petri es persistente si para cualquiera de dos transiciones habilitadas, el disparo de una transición no deshabilitará a la otra transición.

*Distancia Sincrónica*: Es una métrica asociada al grado de dependencia mutua entre dos eventos en un sistema condición/evento. La distancia sincrónica de las transiciones $t_1$ y $t_2$ en la $PN=(N,M_0)$ está dada por:

$$d_{12} = \max_{\sigma} \left| \overline{\sigma}(t_1) - \overline{\sigma}(t_2) \right|$$

donde σ es una secuencia de disparos iniciando en cualquier marcado M perteneciente a $R(N, M_0)$ y $\overline{\sigma}(t_i)$ es el número de veces que una transición $t_i$ es disparada en σ.

## Métodos de análisis de propiedades dinámicas

Los métodos de análisis de las propiedades dinámicas son tres:

1. El árbol de cobertura.
2. Matriz de incidencia y ecuación de estado.
3. Reglas de reducción.

*El árbol de cobertura*: Dada la Red de Petri $(N,M_0)$ con marcado inicial $M_0$, se puede obtener tantos nuevos marcados como transiciones habilitadas disparadas. Este proceso resulta en un árbol de marcados infinito para una PN no acotada. Para redes acotadas, el árbol de cobertura es llamado árbol de alcanzabilidad. El algoritmo para calcular todos los posibles marcados es descrito en detalle por Murata [31]. La figura 11a y b, muestra una PN y su árbol de alcanzabilidad.

*Matriz de Incidencia*: Para una Red de Petri *N* con *n* transiciones y *m* lugares, la matriz de incidencia es una matriz de enteros de *n*×*m*, llamada $A = [a_{ij}]$, y $a_{ij}$ está dada por:

$$a_{ij} = a_{ij}^{+} - a_{ij}^{-}$$

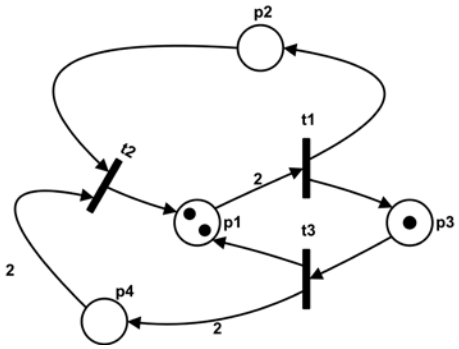
donde $a_{ij}^{+} = w(i,j)$ son los pesos de los arcos de las transiciones *i* a los lugares de salida y $a_{ij}^{-} = w(j,i)$ son los pesos de los arcos de los lugares de entrada *j* a la transición *i*.

Murata [32] expone la ecuación de estados de una PN como una ecuación matricial que define el estado de la red, dado un marcado inicial y una secuencia de disparos de transiciones habilitadas. La ecuación fundamental de una PN es:

$$M_d = M_0 + A^T \sum_{k=1}^{d} U_k$$

Donde $M_d$ es un vector columna de m×1, el vector de control o vector de disparo llamado $U_k$ es un vector también columna de n×1.

La matriz A es llamada de incidencia por que denota como cambiará el marcado. La figura 4 muestra la solución de la ecuación fundamental de una PN cuando se dispara la transición $t_3$ de la PN.




$$\begin{bmatrix}3\\0\\0\\2\end{bmatrix}=\begin{bmatrix}2\\0\\1\\0\end{bmatrix}+\begin{bmatrix}-2&1&1\\1&-1&0\\1&0&-1\\0&-2&2\end{bmatrix}\cdot\begin{bmatrix}0\\0\\1\end{bmatrix}$$

*Figura 4*. Red de Petri y la solución de su ecuación fundamental. El marcado inicial se presenta en la PN de la figura donde $M_0=(2\ 0\ 1\ 0)^T$, la transición habilitada disparada es $t_3$, esto hace que el vector de disparo sea $U=(0\ 0\ 1)^T$ y el marcado resultante es $M_1= (3\ 0\ 0\ 2)^T$ [31].

A la sumatoria de todos los vectores de disparo se le nombrará **x** y se denota:

$$A^T\sum_{k=1}^{d}U_k = A^T\,x$$

Igualando a cero el término $A^Tx$, donde **x** es la secuencia de disparos que se busca, el resultado es un vector conformado por números enteros, que reciben el nombre de invariantes T (transiciones invariantes). En forma análoga, si buscamos un vector ***y*** resolviendo la ecuación Ay=0, se obtienen los invariantes P (lugares invariantes, conocidos también como invariantes S). Las invariantes T y P son útiles para determinar propiedades estructurales de una PN en forma analítica; además, son utilizadas para la obtención de soluciones mínimas mediante PN.

*Reglas de reducción*: Las reglas de reducción permiten convertir sistemas complejos en sistemas más simples de menos lugares y transiciones, preservando sus propiedades originales. La figura 5 presenta las seis reglas de reducción, las cuales son:

a) Fusión de lugares en serie

b) Fusión de transiciones en serie

c) Fusión de transiciones paralelas

d) Fusión de lugares parelelos

e) Eliminación de auto bucles en lugares

f) Eliminación de auto bucles en transiciones

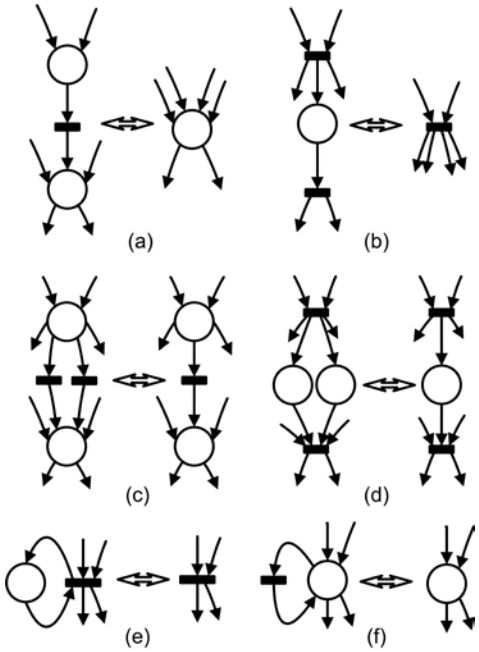



*Figura 5.* Seis formas de reducción que preservan vivacidad, seguridad y acotado. [31]

## Propiedades estáticas

Las propiedades estáticas que se plantean a continuación solo aplican a PN ordinarias y puras.

*Vivacidad Estructural:* Una Red de Petri es estructuralmente viva si tiene un marcado inicial para N.

*Controlabilidad*: De una PN, se dice que es completamente controlable si cualquier marcado es alcanzable desde cualquier otro marcado. Para una red completamente controlable se cumple que

$$\text{Rango } (A) = m$$

donde *m* es la cantidad de lugares de la red.

*Limitación o acotado estructural*: Una PN es limitada estructuralmente si es limitada para cualquier conjunto finito de marcados iniciales $M_0$.

*Conservabilidad*: Una PN es conservativa si existe un entero positivo *y(p),* para cada lugar *p* tal que la sumatoria de marcas sea constante para cada $M \in R(N,M_0)$.

*Repetibilidad*: Una PN es repetible si existe un marcado $M_0$ y una secuencia de disparos $\sigma$ desde $M_0$, tal que las transiciones se disparan infinitamente en la secuencia definida por $\sigma$.

*Consistencia*: Una PN es consistente si existe un marcado $M_0$ y una secuencia de disparos reversible $\sigma$ desde $M_0$ hacia $M_0$, tal que cada transición halla sido disparada al menos una vez en $\sigma$.

## Clasificación de PN

Las clasificaciones realizadas por Murata y Pessen [31, 35] utilizan criterios estructurales o de topología, para lo cual se define la siguiente notación:

$^\circ t = \{p | (p,t) \in F\}$ = lugares de entrada antes de *t*

$t^\circ = \{p | (t,p) \in F\}$ = lugares de salida posteriores a *t*

$^\circ p = \{t | (t,p) \in F\}$ = transiciones de entrada antes de *p*

$p^\circ = \{t | (p,t) \in F\}$ = transiciones de salida posteriores a *p*

Con la notación anterior, se clasifica las Redes en:

a) Máquinas de estado

b) Gráfico de marcados

c) Red de libre escogencia

d) Red de libre escogencia extendida

e) Red de escogencia asimétrica

Las máquinas de estado son una PN ordinaria tal que cada transición *t* tiene exactamente un lugar de entrada y un lugar de salida. Recordando, una PN es ordinaria si el peso de sus arcos siempre vale uno:

$$|^\circ t| = |t^\circ| = 1 \quad \text{para todos los } t \in T$$

Un gráfico de marcados es una PN ordinaria tal que cada lugar p tiene exactamente una transición de entrada y una única transición de salida:

$$|^\circ p| = |p^\circ| = 1 \quad \text{para todos los } p \in P$$

Una Red de libre escogencia es una PN ordinaria tal que cada lugar de la Red posee un arco de entrada proveniente de una transición y posee un arco de salida hacia una transición:

para todo $p \in P$, $|p^\circ| \leq 1$, o $^\circ(p^\circ)$ o alternativamente

para todo $p_1, p_2 \in P$, $p_1^\circ \cap p_2^\circ \neq \varnothing \Rightarrow |p_1^\circ| = |p_2^\circ| = 1$

Una Red de libre escogencia extendida es una PN ordinaria tal que:

$p_1^\circ \cap p_2^\circ \neq \varnothing \Rightarrow p_1^\circ = p_2^\circ = 1$ para todo $p_1, p_2 \in P$

Una Red de escogencia asimétrica o red simple es una PN ordinaria tal que:

$p_1^\circ \cap p_2^\circ \neq \varnothing \Rightarrow p_1^\circ \subseteq p_2^\circ$ ó $p_1^\circ \supseteq p_2^\circ$ para todo $p_1, p_2 \in P$

## Evolución de las pn y su impacto sobre el modelado

Las PN tienen más de 45 años de existencia desde su formulación en 1962 y en este periodo se han realizado miles de publicaciones sobre o relacionadas con: teorías de PN, desarrollo de nuevos tipos de PN, aplicaciones exitosas de PN, nuevos ámbitos de aplicación, nuevas formas de implementación, etc. En esta sección se presentan en forma cronológica distintos trabajos que contribuyen al desarrollo de programas para PLCs.

Uno de los primeros recuentos sobre PN fue realizado por J. Peterson [36] y publicado en 1977, y consiste en un repaso histórico de las Redes de Petri en sus primeros quince años y cómo éstas habían sido utilizadas en ciencias de la computación para modelados de sistemas paralelos. Se expone que las PN son un mecanismo de modelado matemático donde los gráficos están constituidos por dos tipos de nodos y arcos con pesos asociados.

Explica que las Redes de Petri, poseen propiedades estáticas y dinámicas, donde las propiedades estáticas se asocian a la topología de la Red, mientras que las dinámicas se refieren a la ejecución y evolución de las marcas en la Red. La ejecución de una Red de Petri se refiere al movimiento de las marcas dentro de la Red, la cual requiere de un conjunto de marcas iniciales al que se conoce como marcado inicial, el cual define el estado de la Red. El marcado cambia como resultado del disparo de transiciones. En dicho trabajo aparecen numerosos ejemplos sobre PN aplicados a algoritmos. La figura 6 modela el problema de mutua exclusión de dos operaciones independientes mediante un semáforo.

Peterson expone por qué las Redes de Petri pueden verse como una secuencia de eventos discretos cuyo orden de ocurrencia puede ser una de muchas posibilidades permitidas. Esto brinda la característica de ser aptas para modelar eventos no deterministicas. Peterson detalla algunas propiedades de las PN, así como las tres clases de PN existentes en ese momento: el gráfico de marcados, las redes de simple escogencia y PN simples, pero no utiliza ninguna notación matemática para definirlas, simplemente realiza una descripción.

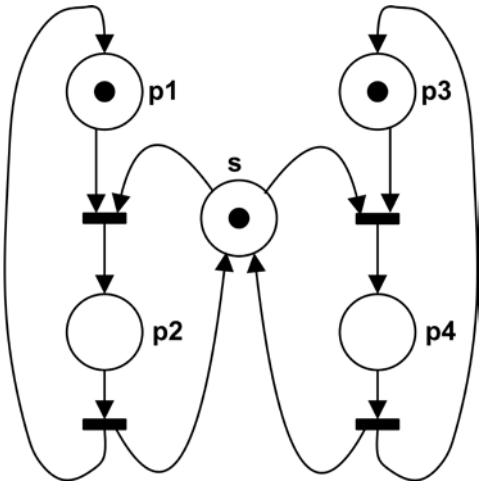



*Figura 6.* Red de Petri que representa procesos que comparten recursos. Se muestran los lugares en forma de círculos, las transiciones como barras, las marcas como puntos negros y los pesos de los arcos se entienden como unitarios [36].

En España, J. Martínez y M. Silva, et al. en 1982, [27] plantean que las Redes de Petri se pueden implementar mediante tres formas: las técnicas cableadas, las técnicas programables con ROM o PLA y las técnicas programables con PLC. En esta investigación, se concentran en las técnicas cableadas. Para ello definen el concepto de celda modular sincrónica y asincrónica construidas con flip-flop J-K, D y compuertas digitales. Las Redes de Petri que construyen son las llamadas PN binarias, donde los lugares sólo almacenan una marca a la vez. Cabe rescatar que este artículo fue pionero en plantear las PN como alternativa de diseño para algoritmos de PLC.

En ese mismo año 1982, T. Murata [31] establece las propiedades de los gráficos de marcados, que es una subclase de las Redes de Petri. Estudia la vivacidad de la Red, su seguridad, su controlabilidad y el problema de su alcanzabilidad, a la vez que realiza un ejemplo detallado del modelo de un sistema de cómputo paralelo.

Dada la importancia de las Redes de Petri y su aporte en diversas áreas de investigación, se publica en 1989, a petición del Comité de *Proceedings of the IEEE*, un tutorial hecho por Murata [32]. En este trabajo se mencionan las áreas de aplicación más importantes, donde destacan: medición de rendimiento, protocolos de comunicación, bases de datos distribuidas, software distribuido, software paralelo, lógica programable VLSI, circuitos sincrónicos, estructuras asincrónicas, compiladores y sistemas operativos, sistemas informáticos de oficina, lenguajes formales, programas lógicos, redes locales, redes neuronales, filtros digitales y modelos de decisión. En dicho tutorial, [32] se estudian más de 310 referencias bibliográficas por lo que fue el estudio más preciso hasta entonces publicado sobre PN, por ejemplo, menciona la existencia de muy pocos artículos sobre la relación entre PN y PLC, nombra siete artículos sobre sistemas de control industrial y cuatro artículos sobre modelado de sistemas de eventos discretos.

En dicho artículo, se explica que el modelado de PN utiliza el concepto de condiciones y eventos, los lugares representan las condiciones y las transiciones los eventos. Una transición o evento tiene un número determinado de lugares de entrada y lugares de salida. Se establecen varias interpretaciones de una PN.

En el artículo se expone en detalle la teoría de las PN y se demuestra más de 35 teoremas sobre las Redes de Petri, así como numerosos problemas. El artículo finaliza con una exposición teórica detallada de las redes temporizadas utilizadas en la evaluación de rendimientos de sistemas, de las redes estocásticas y de las redes de alto nivel conocidas como Redes de Petri coloreadas, utilizadas para modelar la lógica de programas.

También en ese año, D. Pessen [35] publica en su libro el método cascada, el cual permite diseñar diagramas escalera (acrónimo en ingles, LLD). Los diagramas escalera son utilizados para diseñar circuitos lógicos implementados mediante relés electromecánicos o también el diagrama LLD se utiliza como lenguaje de programación para controladores lógicos programables. El método propuesto no garantiza que el circuito sea mínimo en la cantidad de componentes que utiliza, sino que las soluciones sean fácilmente obtenidas para problemas complejos. Pessen, sin mencionarlo en su trabajo, desarrolla una implementación práctica en contactos y relés de una PN ordinaria, binaria, segura, acotada, del tipo de libre escogencia, que es fácilmente implementada en un PLC.

En 1990, L. Holloway et al.[17] proponen una nueva red llamada Red de Petri Controlable (CPN), que se utiliza para modelar un sistema dinámico de eventos discretos. A partir del modelo obtenido con las CPN, se sintetiza el controlador utilizando un algoritmo cuyo resultado garantiza que el sistema no entrará en

*Cuadro 1*. Interpretaciones de una PN.

| Lugares de entrada | Transiciones | Lugares de salida |
|---|---|---|
| Precondiciones | Eventos | Postcondiciones |
| Datos de entrada | Paso de cómputo | Datos de salida |
| Señal de entrada | Señal de procesamiento | Señal de salida |
| Necesidad de recursos | Trabajo o tarea | Recursos liberados |
| Condiciones | Cláusula lógica | Conclusiones |
| Memorias | Procesador | Memorias |

estados prohibidos. En este trabajo, dada una planta y las especificaciones del comportamiento deseado, el objetivo es sintetizar el controlador apropiado. Esto es análogo a los problemas de control automático, donde dada una planta continua, se diseña el controlador. Antes de este trabajo, típicamente la lógica de control era incluida como parte del modelo y la evaluación o modificación de las políticas de control requerían de una modificación de la PN.

Holloway establece que la lógica de control se puede separar del modelo del sistema, introduciendo el concepto de lugares de control externos. Esto es un cambio novedoso de lo realizado hasta entonces con PN. Para ejemplificar el desarrollo propuesto, se realiza el modelado de una celda de manufactura flexible (FMC), que utiliza vehículos guiados automatizados (AGV). Se resalta que la solución obtenida es compacta y computacionalmente más eficiente, si se compara con el procedimiento obtenido mediante la teoría de autómatas finitos.

En 1992, L. W. Schruben [38] realizó una revisión de los modelos gráficos existentes para modelar sistemas dinámicos de eventos discretos (DES). Mencionó que los métodos gráficos con pocos tipos de objetos son más fáciles de aprender que aquellos con muchos tipos de objetos. Por otro lado, modelos con muchos tipos de objetos no garantizan que sean más poderosos que los métodos con pocos objetos. Los métodos estudiados fueron: las Redes de Petri, los gráficos de marcados, las redes de procesos y las máquinas de estado.

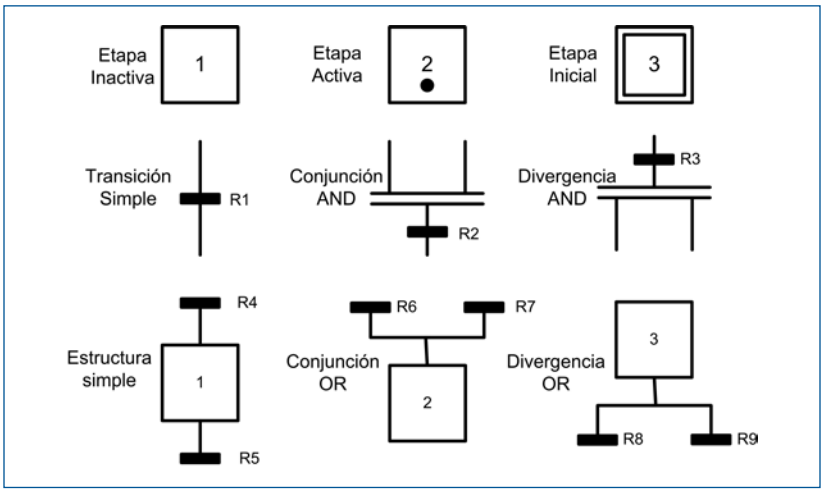
Paralelamente a la búsqueda de métodos de diseño formales, sencillos y poderosos, otros investigadores buscaban herramientas para verificar programas existentes de PLC, tal es el caso de I. Moon en 1994 [29]. Éste desarrolló un método para verificar la operabilidad y seguridad de los controladores lógicos programables, inspirado en el método para prueba de protocolos de comunicación y de circuitos VLSI de 1986 de E. M. Clarke. Para esto utilizó una representación booleana del programa, el cual se sometió a una serie de preguntas lógicas de verificación. Este método fue desarrollado por la carencia de métodos formales para la verificación de programas en PLC. Moon hizo un recuento de los distintos lenguajes para programar PLC, entre ellos, el diagrama escalera y los bloques de funciones; además, efectuó un repaso del funcionamiento y del ciclo de trabajo o de refrescamiento del PLC.

Y. Cai et al. en 1995 [3] propusieron el diseño de sistemas de control secuencial utilizando los invariantes *p* de las Redes de Petri. En este trabajo se encuentra una definición de las PN y de esas invariantes. Los autores tomaron una Red de Petri y la dividieron en módulos. Cada módulo debe cumplir la propiedad de ser invariante *p*, es decir, que la cantidad de marcas sea constante en un lugar. Los autores brindan una solución gráfica a un ejemplo de llenado de tanques de agua. Sin embargo, la solución obtenida no es corroborada en forma algebraica, ante lo cual aducen que la realizarán en posteriores trabajos.

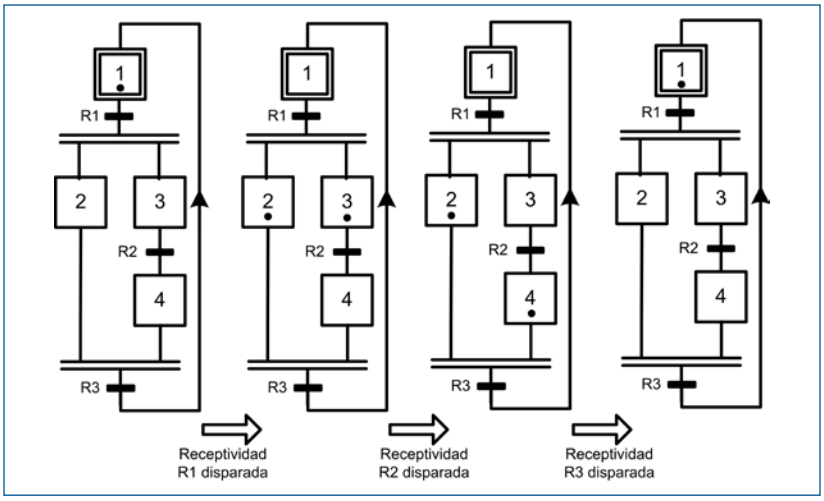
S. Pettersson et al. en 1995 [37] propusieron un modelo híbrido general, al separar la planta a lazo abierto para construir un controlador híbrido. El sistema híbrido puede ser modelado por una PN híbrida. El artículo desarrolla los conceptos de los sistemas continuos y de los sistemas de eventos discretos, y propone un conjunto de ecuaciones para modelar un sistema que combine los sistemas discretos como híbridos. Asimismo, definieron las propiedades de un controlador híbrido y cómo sintetizarlo, a la vez que presentaron las Redes de Petri híbridas como un mecanismo para modelado de sistemas híbridos y realizaron un ejemplo de modelado de un proceso industrial.

La Comisión Electrotécnica Internacional (IEC), en el año de 1988, oficializó el

estándar IEC 848, conocido como *Sequencial Function Chart* (SFC), el cual se basa en el estándar francés Grafcet de 1977. René David [6] en 1995 publicó un estudio comparativo entre las PN y el SFC como herramientas para especificar algoritmos para controladores lógicos programables. La importancia del estudio radica en que presenta la primera comparación de un formalismo matemático con un estándar europeo ampliamente aceptado, publicado en la IEEE.




(a)




(b)

*Figura 7*. a) Elementos básicos de SFC. b) Cuatro SFC donde se muestra la evolución de las marcas en las etapas, cuando se valida la receptividad de una transición.

En el estudio se concluye que el Grafcet o SFC se puede ver como una Red de Petri del tipo binaria, acotadas donde las transiciones son disparadas en forma determinista. En el artículo se realiza una exposición del estándar SFC y del álgebra de eventos, así como la solución para la automatización de unos tanques de almacenamiento de agua. La solución es obtenida con los métodos de PN, SFC, máquinas de estado y tablas de estado; al final se realiza una comparación entre las soluciones para determinar cuál método de diseño fue el menos elaborado.

La figura 7 muestra la representación básica de un SFC, los lugares de una PN se interpretan como etapas en los SFC y las transiciones, como receptividades. En la figura 7b se aprecia la evolución de un SFC cuando se disparan las receptividades R1, R2 y R3 en el orden mencionado.

En 1997, L.E. Holloway et al. [17] hicieron un estudio sobre Redes de Petri utilizadas en el control de sistemas de eventos discretos. El artículo estudia 107 referencias bibliográficas y establece tres importantes corrientes de investigación para controladores:

a) Aproximación de comportamiento controlado.

b) Aproximación de controlador lógico.

c) Aproximación teórica de control.

La aproximación de comportamiento controlado se refiere a modelos con Redes de Petri que describen el comportamiento del sistema a lazo cerrado, integrando tanto la planta como el controlador y cuando se obtiene el comportamiento deseado, se extrae la lógica del controlador para su implementación.

La aproximación de controlador lógico consiste en diseñar en forma directa el controlador de la planta, pero sin modelar la planta. El objetivo es definir el comportamiento de las entradas – salidas del controlador para obtener un comportamiento controlado del sistema a

lazo cerrado. El método gráfico Grafcet, definido en el estándar IEC 60848, es un ejemplo de esta aproximación.

La aproximación teórica de control proviene del paradigma de control automático, donde dado el modelo dinámico de una planta y las especificaciones deseadas del lazo de control, se sintetiza un controlador que cumpla con las especificaciones. En este esquema se distingue claramente la planta, el controlador y el flujo de información entre ambos. El control mediante autómatas finitos brinda un marco de trabajo conceptual para establecer las propiedades de los DEDS; sin embargo, no son convenientes debido a la gran cantidad de estados que introducen. En este sentido, las PN son una alternativa para el modelado de DEDS, pues los modelos con Redes de Petri son más compactos que los modelos con autómatas finitos.

Holloway, Krogh y Giua plantearon en su investigación cómo sintetizar los controladores para plantas modeladas por Redes de Petri. Exponen dos grandes aproximaciones: el control retroalimentado de estado o "*state feedback control*" que utiliza las Redes de Petri Controladas (CtlPN) y el control retroalimentado de eventos "*event feedback control*" que utilizan las llamadas Redes de Petri Etiquetadas (LPN). A continuación, se presenta la descripción formal de las CtlPN, que consiste en una tripleta:

$$N^c=(N, C, B)$$

Donde:

N = (P,T,F) es una PN ordinaria.

$C$ = Es un conjunto finito de lugares de control.

$B \subseteq (C \times T)$ es un conjunto de arcos dirigidos que conectan los lugares de control con las transiciones.

Otro punto tratado en el artículo es la definición de las LPN, la cual se entiende como una quíntupla:

$$LPN=(N, \Sigma, l, m_0, F)$$

Donde:

N = (P,T,F) es una PN ordinaria.

$\Sigma$= Conjunto finito de eventos (alfabeto).

$l:T \rightarrow \Sigma$ es la función de etiquetado donde se asigna a cada transición un evento.

$m_0$ = Marcado inicial.

$F$ = Es un conjunto finito de marcas finales.

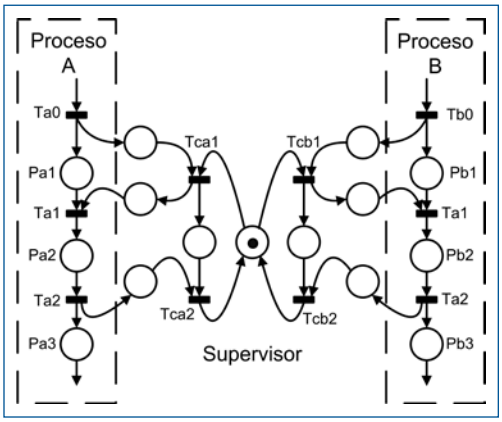
Finalmente, los autores desarrollan la teoría del control supervisor y el diseño del supervisor, así como las propiedades de este tipo de control con Redes de Petri. Los autores resaltan que hasta la fecha no existe una completa y formal comparación entre las soluciones obtenidas por métodos basados en PN y las basadas en autómatas finitos.

En 1997, Taholakian et al. [39] propusieron una metodología para diseñar, simular y codificar sistemas de control basados en PLC usando Redes de Petri, llamada PN⇔PLC. Debido a la complejidad creciente de los sistemas de manufactura, específicamente, de las celdas de manufactura flexible (FMC), los autores proponen cómo implementar la red en un lenguaje de programación estándar definido dentro del estándar de programación IEC 1131-3, mediante la implementación con diagramas escalera. Los autores mencionan que la automatización de una FMC o de sistemas más complejos no se podría realizar si las tareas de control no se distribuyeran entre dispositivos de control distribuido (DCS). El método desarrollado fue dividido en las siguientes partes:

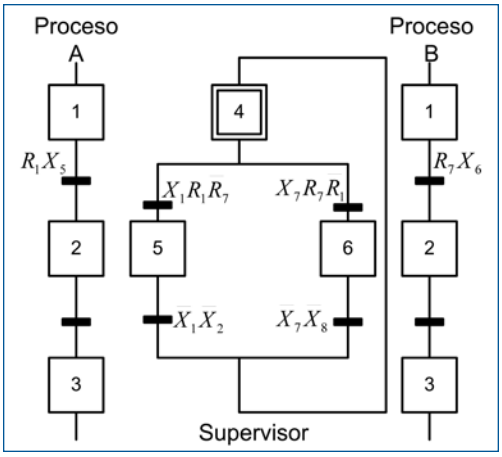
a) Especificación de requerimientos de los DCS
b) Especificación funcional de los DCS
c) Diseño de DCS que cumplan los requerimientos
d) Generación de código para los DCS
e) Implementación en software

Del método PN⇔PLC, se puede decir que es un método gráfico que consiste en descomponer una PN ordinaria y segura, en secciones que son mapeadas en forma casi directa a una estructura de contactos de diagrama escalera. Al final, la unión de las distintas secciones en el diagrama escalera conforma un circuito cuyo comportamiento es equivalente al descrito en la Red de Petri.

G. MuŠiΣ y D. Matko en 1998 [30] trabajaron la línea de investigación de la aproximación teórica de control definida en [18], donde se establece el control supervisor mediante una PN flexible para un proceso industrial por lotes (figura 8a). Este trabajo realiza un recuento histórico del control supervisor de sistemas DES que originalmente se basó en la teoría de autómatas finitos, cuya teoría de control supervisor pretende asegurar que la planta nunca alcance ciertos estados, ya sean inseguros o de bloqueo. El método que plantean se basa en el cálculo de los invariantes *p* conocidos como *S-invariant* o *P-Invariant*. La particularidad de este trabajo es que implementa el control supervisor con SFC, que es un lenguaje estandarizado de programación de PLC definido en IEC 1131-3, lo cual se aprecia en la figura 8b.




(a)




(b)

*Figura 8.* a) Red de Petri de dos procesos y su respectivo control supervisor b) Implementación en SFC de la PN, según estándar IEC 1131-3 [30].

G.Frey en ese mismo año propone en [8] la utilización de una nueva clase de PN llamadas Redes de Petri Interpretadas (IPN o SIPN) para la verificación y validación de algoritmos de control. Para esto, parte del hecho de que la mayoría de sistemas lógicos de control industrial utilizan especificaciones informales. El término "informal" se refiere a todo aquello que no está basado en un formalismo bien definido semántica y sintácticamente. Detalla los siguientes conceptos: formalización, síntesis, verificación, implementación, validación y simulación.

Define la formalización como el proceso de convertir una especificación informal en formal. Menciona ejemplos de especificaciones formales tales como: PN, SIPN, los sistemas de ecuaciones booleanas o los autómatas finitos. La síntesis es el proceso de generar el algoritmo de control de la especificación formal. La verificación consiste en aplicar métodos formales, con la finalidad de probar si el algoritmo de control cumple las especificaciones. Finalmente, la implementación se entiende como la obtención del código del sistema o circuito, según las especificaciones formales. La validación muestra si el proceso por controlar se comporta como debería, esto permite detectar errores en

la especificación formal. La simulación es utilizada como un método para realizar las validaciones.

Frey define formalmente las SIPN como una óctupla,

$$SIPN = (P, T, F, M_0, I, O, C, A)$$

Donde:

$(P,T,F,M_0)$ es una PN ordinaria, libre de auto bucles, con P lugares, T Transiciones, F arcos y un marcado inicial $M_0$.

I: conjunto finito de entradas binarias, $I \neq \varnothing$.

O: conjunto finito de salidas binarias, $O \neq \varnothing$, $I \cap O \neq \varnothing$.

C: un mapa que asocia cada transición $t_i \in T$ con una condición de disparo externa, expresión booleana en I.

A: un mapa que asocia cada lugar $p_i \in P$ con una acción.
A(i)={0,1,-} la cual es activada siempre que el lugar esté marcado.
A(i)[j] especifica el valor de la señal de salida $o_j$.

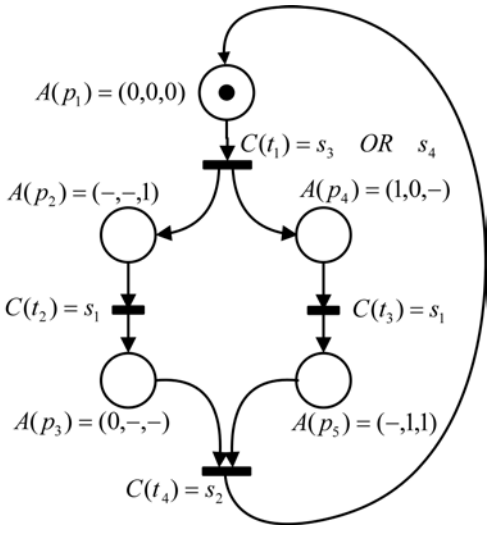



*Figura 9.* Red de Petri SIPN. [8]

Junto a la definición de SIPN, Frey realiza la automatización formal de un reactor químico, pero no realiza ni establece lineamientos para la implementación. La figura 9 muestra la gráfica de una SIPN.

En 1998, M. Uzam [40] realizó el diseño de un sistema de control de eventos discretos y su implementación a partir de una nueva clase de PN que define y llama APN (*Automation Petri Nets*). Esta nueva clase de Redes de Petri para automatizar procesos industriales es más robusta que la de las SIPN, puesto que no son ordinarias ni binarias, ya que sus arcos pueden tener pesos mayores a uno y sus lugares pueden acumular más de una marca, lo cual hace que las APN no sean acotadas.

El autor realiza un recuento de los cinco lenguajes de programación estandarizados por la IEC 1131-3, gráfico de funciones secuencial (SFC), diagrama escalera (LLD), diagrama de bloque de funciones (FBD), lista de instrucciones o nemotécnico (IL) y texto estructurado (ST). Uzam explica que típicamente un sistema de control de eventos discretos (DECS) se compone de dos bloques, el sistema de eventos discretos (DEDS) análogo a la planta en sistemas continuos y el controlador de eventos discretos (DEC) análogo al controlador. La figura 10 muestra este esquema.

La definición de la APN es una tupla de orden nueve definida como,

$$APN = (P, T, Pre, Post, In, En, \dot{\times} Q, M_0)$$

Donde:

$(P,T,M_0)$: P es el conjunto de lugares, T el conjunto de transiciones y $M_0$ el marcado inicial.

Pre : $(P \times T) \rightarrow N$ es una función de entradas que define los arcos desde los lugares y hacia las transiciones, con un peso definido por N.

Post: $(T \times P) \rightarrow N$ es una función de salidas que define los arcos desde las trasiciones y hacia los lugares, con pesos de N.

In:$(P \times T) \rightarrow N$ es una función de entradas inhibidoras que define los arcos inhibidores

entre lugares y transiciones, estos son representados como pequeños círculos.

En:(P×T)→ N es una función de entradas habilitadoras que define los arcos habilitadores entre lugares y transiciones, estos son representados como una flecha, pero sin rellenar.

$X= \{x_1, x_1,..., x_m\}$ se define como un conjunto finito de condiciones de disparo no vacío, asociado a las transiciones.

$Q= \{q_1, q_1,..., q_m\}$ se define como un conjunto finito de acciones asignadas a los distintos lugares.

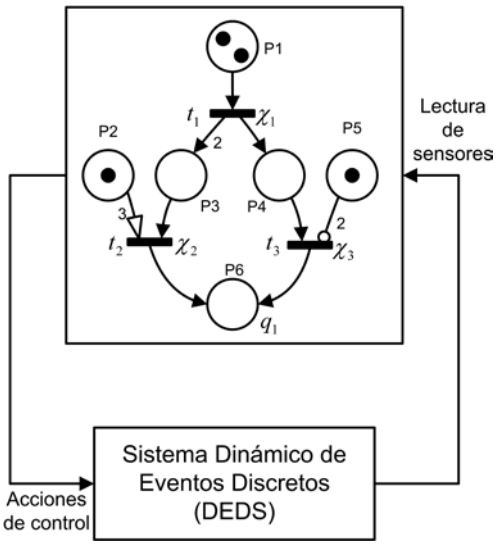



*Figura 10*. La figura muestra una Red de Petri del tipo APN, que modela el controlador de Sistemas de Eventos Discretos (DES). [40]

El modelo APN es apropiado para el manejo de señales de sensores y actuadores. Este artículo explica cómo modelar con APN y cómo realizar la implementación en un PLC S-200 de Siemens, para lo cual desarrolla un método de transformación llamado TPL o "*Token pass logic*", el cual permite obtener la implementación en el lenguaje estandarizado LLD.

Las redes APN utilizan posiciones de memoria o contadores para representar lugares que pueden tener más de una marca o "*Tokens*", clara diferencia entre las APN y otros tipos de PN utilizadas para automatizar, ya que las APN, como se dijo, no son acotadas. Finalmente, el autor señala que no se trató el problema de la verificación del algoritmo de control.

G. Frey en el 2000 [10], hizo una exposición de las propiedades de las SIPN propuestas en [8], pero expandió el modelo y desarrolló sus propiedades dinámicas y estáticas; además de definir las reglas para la representación gráfica del modelo. Éste es un modelo sincronizado con el ambiente, cuya mejora respecto al previo consiste en que las señales de salida son de varios tipos y pueden activarse según los marcados de un lugar.

La definición formal de una SIPN se describe como una 9-tupla tal que

$$SIPN = (P, T, F, m_o, I, O, \phi, \omega, \Omega)$$

Donde:

$(P, T, F, m_0)$ es una PN ordinaria con P lugares, T transiciones, F arcos y un marcado inicial $m_0$.

I: conjunto lógico de señales de entrada $|I|>0$.

O: conjunto de señales de salida.

$\phi$: mapeo de asociación entre las transiciones $t_i \in T$ y las condiciones de disparo de la transición $\phi(t_i)$= función booleana dentro de I.

$\omega$: mapeo que asocia cada lugar $p_i \in P$ con una salida $\omega(p_i) \in (0,1,-)^{|O|}$ donde (-) significa 'no importa'.

$\Omega$: la función de salida combina las salidas $\omega$ de todas los lugares marcados $\Omega$: $m \rightarrow (-, 1, 0, c, r_0, r_1, c_0, c_1)^{|O|}$. La salida combinada puede ser indefinida (-), uno (1), cero (0), contradictoria (c), reiterativa en cero o en uno ($r_0$, $r_1$) o una combinación entre contradictoria y reiterativa ($c_0$, $c_1$).

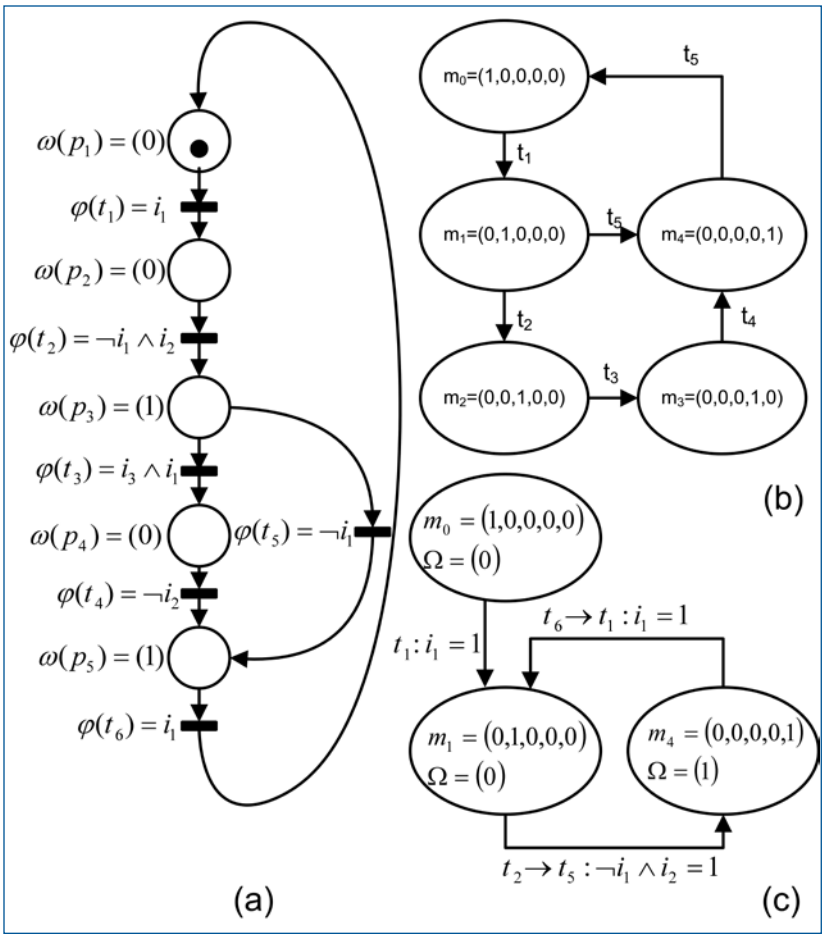



*Figura 11.* a) SIPN b) Árbol de alcanzabilidad para la PN c) Árbol de alcanzabilidad para la SIPN [10].

La figura 11a muestra una Red de Petri tipo SIPN, la 11b muestra todos los marcados que se puede alcanzar, lo cual es conocido como el árbol de alcanzabilidad definido en [32], y la 11c muestra el árbol de alcanzabilidad para SIPN propuesto por Frey.

Frey explica las principales diferencias entre las SIPN y el SFC definido en el estándar IEC 1131-3. Por ejemplo, señala que en un SFC no existen estados transitorios, pero debido a los tiempos tan cortos del ciclo de operación de un PLC, las etapas de un SFC se comportan como etapas quasi-transitorias. En los modelos de SFC sólo se permite un estado inicial y en un SIPN, no. Dentro de un SFC, las etapas posteriores no son revisadas con las reglas de disparo. Además, señala que las transiciones son activadas por eventos, los cuales por definición no son simultáneos, mientras que las SIPN son activadas por señales, lo que permite esta última posibilidad.

Frey [9] propone en el mismo Congreso un método automático que utiliza las SIPN para generar vía software diagramas escalera (LLD) según la norma IEC 1131-3. La generación automática se logra mediante un programa que toma las especificaciones del SIPN y genera un código optimizado.

Frey vuelve a expandir el modelo en [11], para poder modelar sistemas híbridos que permitan trabajar con retardos de tiempos. El modelo ampliado es definido como tSIPN =( SIPN, $\nu$, $\tau$) donde $\nu$ es la definición del tipo de dato (booleano, entero, real) y $\tau$ es un mapeo de cada arco $f_i \in (P \times T) \cap F$ con el tiempo de retardo $\tau_i \in R^+$. Es decir, $f_i=(p_j, t_k)$, $\tau_i$ es el tiempo que una marca debe permanecer en el lugar $p_j$ antes de que sea removido por la transición $t_k$.

Frey [12] realiza un análisis exhaustivo de los métodos formales para programar PLC y de los estándares que se han generado para su programación. El artículo explica en detalle un modelo genérico para diseñar algoritmos de control de procesos industriales, los cuales serán implementados en PLC. Adicionalmente, el artículo muestra los distintos estándares generados en el transcurso de los años para regular la programación de PLC.

Frey propone el diseño de controladores como los que se aprecia en la figura 12. El proceso de diseño se debe dividir en tres actividades:

a) Reinterpretación y formalización.

b) Síntesis y la implementación.

c) Verificación y validación (V&V).

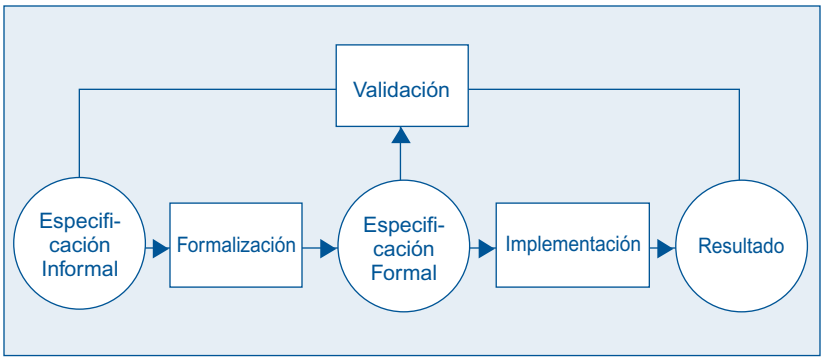



*Figura 12*. Esquema general del método de diseño para algoritmos de control de PLC [12].

La formalización es dividida en tres actividades independientes y son realizadas según casos particulares. Las tareas que menciona son: formalización de propiedades específicas, modelado formal de procesos sin controlar modelado formal directo de algoritmos de control.

La síntesis y la implementación utilizan como insumo la especificación formal obtenida en las actividades anteriores. La síntesis del algoritmo pretende obtener un diseño del algoritmo y la implementación de un algoritmo codificado de acuerdo con un lenguaje de programación de PLC estandarizado. Las corrientes actuales lo que pretenden es que ese código estándar sea realizado automáticamente utilizando un software especializado.

Las actividades de verificación y validación (V&V) son actividades que se aplican a los resultados de la especificación formal y al resultado final. La V&V de especificaciones informales se divide en: basadas en el modelo, basadas en el no modelo y las basadas en las limitantes. La verificación de la especificación formal puede ser realizada de las siguientes seis formas: Redes de Petri, sistemas condición/evento, autómatas finitos e híbridos, lógica de alto nivel, lenguajes sincronizados, sistemas generales de transición y ecuaciones algebraicas. Finalmente, la validación del resultado final se puede realizar utilizando: simulación, análisis de alcanzabilidad, chequeo del modelo y métodos de prueba de axiomas del modelo utilizando lógica matemática.

Por otra parte, P. Deussen en el 2001 [7], se dedicó a verificar formalmente programas para PLC. Para esto creó una nueva subfamilia de Redes de Petri llamada RN o "Register Net". Con este nuevo tipo de PN, los autores desarrollan un método para extraer del programa del PLC, una red que será utilizada para lo siguiente: verificar la ausencia/presencia de bloqueos, verificar la ausencia de errores de ejecución tales como rebases y divisiones entre cero, y verificar la seguridad del sistema programado.

Deussen en su artículo, define la RN como una óctupla,

$$RN= (P, T, F, R, \rho, G,P, s),$$

Donde:

P, T, F son: lugares, transiciones y arcos de una red ordinaria.

R: es un conjunto finito de registros $R \cap (P \cup U)=\varnothing$.

$\Sigma$ un mapeo $T \rightarrow R \times R$.

*G*: mapeo parcial $G_T$ , $\{G_T\}$ $t \in T$.

*P*: mapeo parcial $P_T$ , $\{P_T\}$ $t \in T$.

s: es un estado inicial, un subconjunto de P.

T. Mertke en el 2001 [28], se apoyó en el trabajo de Frey [ 8, 9, 10] para desarrollar una propuesta para la verificación de modelos de algoritmos de control para que sean utilizados por no especialistas. Él define como "no especialistas" a ingenieros de control con nulo conocimiento en ciencias de la computación. Establece el proceso de verificación en seis etapas:

a) Diseño de un controlador y generación de código escalera (LLD)

b) Compilar el programa del PLC a un modelo de red SIPN, usando un generador de Redes de Petri.

c) Modelado del sistema del PLC.

d) Modelado de la planta por PN.

e) Combinar los tres modelos anteriores y generar un modelo general de todo el sistema en términos de PN.

f) Especificar los requerimientos funcionales y de seguridad en la semántica de la lógica temporal, pues con estas reglas se evalúa el modelo general.

En [28] se desarrolla un caso de estudio de una automatización a nivel industrial, donde verifican el programa para unos compresores de aire y unos tanques de almacenamiento.

En el año 2001, E. Jiménez [20] presentó en su tesis doctoral cómo utilizar las Redes de Petri para automatizar procesos industriales. A cada red le realiza su respectiva implementación en un lenguaje normado de programación de PLC. Adicionalmente, también realiza implementaciones en Matlab con el propósito de evaluar los rendimientos de los algoritmos. Finalmente, en este trabajo se realiza la automatización completa de una planta industrial.

En ese mismo año, K. John [21] publicó un libro que explicaba en detalle la norma IEC 61131-3, la cual entraría a regir en el 2003. Dicha norma se ocupa de los cinco lenguajes estandarizados de programación y a su vez, es una actualización de la IEC 1131-3 de 1992.

G. Frey en el 2002 [12], estableció la necesidad de utilizar los estándares de desarrollo de software para asegurar la calidad del código generado para los PLC. Con base en los criterios de calidad de software definidos en la ISO/IEC 9126 (funcionamiento, confiabilidad, usabilidad, eficiencia, portabilidad y mantenibilidad), el autor establece cómo desarrollar programas que cumplan con lo estipulado en el estándar. Por ejemplo, propone una serie de métricas basadas en las SIPN para medir cada criterio.

Al año siguiente, Frey [13] vuelve a expandir las SIPN para poder modelar sistemas híbridos mediante una 10-tupla llama hSIPN. En ese mismo artículo, define y desarrolla la metodología para utilizar las SIPN en diseños jerárquicos de programas para PLC. La filosofía que utiliza para el modelado es la llamada de abajo hacia arriba (*bottom-up*). Este nuevo tipo de Red de Petri es una tupla de orden once y recibe el nombre de $SIPN^H$.

S.Peng y M. Zhou en el 2003 [33] propusieron un nuevo método para modelar programas de PLC, al cual llaman SBSPN "*Sensor Based Stage Petri Nets*". Este nuevo tipo de Redes de Petri es capaz de simplificar modelos de complejos procesos. Para esto se define la Red y su representación gráfica. Este nuevo tipo de PN permite:

a) Que las entradas y salidas sean representadas por lugares y expresen el estado del programa de control del DECS (Controlador del Sistema de Eventos Dinámicos). Esto es diferente a los SIPN, APN, etc.

b) Los SBSPN permiten evaluar la lógica de control antes de la implementación y eliminar el inter-bloqueo entre elementos del programa de control.

c) No existen recursos compartidos, debido a que se elimina la ocurrencia de las entradas/salidas al mismo instante.

d) Para el modelado de DECS complejos, las SBSPN permiten el desarrollo e implementación de módulos.

Formalmente:

$$SBSPN = (RTPN) \cup (SCN)$$

Donde RTPN "Real time petri nets" se define como

$$RTPN = (P,T, I, O, m_0, D, X, Y)$$

$(P, T, I, O, m_0)$: es una Red de Petri definida.

D: es función del retardo de disparo.

X: es una función booleana de señales de entrada.

Y: es una función de salida booleana.

y SCN "Stage control net" es

$$SCN = (P_{SI}, T_{SI})$$

$P_{SI}$: donde son un conjunto de lugares de estado.

$T_{SI}$: son un conjunto de transiciones de estado dentro de SCN.

*En nuestras universidades es necesario replantear los cursos concernientes a la enseñanza del funcionamiento y programación de PLCs.*

En este trabajo se realiza un estudio de un caso práctico del modelado de una celda de manufactura flexible que utiliza un brazo robot.

M. Bani y G. Frey [2] realizaron en el 2003 un estudio sobre trabajos de investigación dedicados a formalizar programas de PLC. La formalización la entiende como la definió Frey en [9,12]: proceso de transformar el código no formal existente, en uno formal, a lo cual se le conoce también como reinterpretación. Ellos clasifican la formalización de acuerdo con tres criterios:

*En el plan de estudio se debe prestar mayor atención al levantamiento formal de los requerimientos y a su validación, al modelado formal de la planta y del controlador, a la validación de los modelos, a la implementación del software y a su verificación.*

a) Formalización de módulos o partes del sistema de control.
b) Formalización del programa completo (para un PLC).
c) Formalización de toda la configuración de control (para varios PLC en modalidad jerárquica).

Los trabajos analizados en este estudio utilizan los modelos de autómatas finitos o Redes de Petri.

En el 2003, Klein et al. [15] desarrollaron un editor gráfico donde se "dibuja" una SIPN y se genera automáticamente el código para PLC según lo estipulado en IEC 61131-3. En el artículo se presenta el editor desarrollado con la herramienta DiaGen. Luego se muestra cómo el editor se utiliza para verificar y validar diseños. Finalmente, se muestra el código generado por el software.

En el 2004, E. Ávila et al. [1] publicaron un estudio de verificación de programas de PLC utilizando las propiedades estáticas o topología de la red. Las principales propiedades que se comprueban en el trabajo son la vivacidad y la seguridad. Para esto, se valen de un método matemático que utiliza el teorema de rango y el método gráfico que utiliza las llamadas reglas de reducción para transformar una red compleja en otras simples. El uso de estas técnicas está descrito en Murata [31, 32]. El aporte de esta investigación radica en demostrar la utilidad práctica en problemas de automatización reales.

Enmarcado en la aproximación teórica de algoritmos de control definida en [18], M. Uzam, 2004 [41], propone la síntesis de un controlador de eventos discretos a partir de Redes de Petri y la teoría de regiones. Para esto retoma el ejemplo desarrollado por Holloway en [17], que consistía en la automatización de una celda de manufactura flexible del tipo AGV. El método requiere que se defina previamente el modelo del sistema mediante una Red de Petri y se especifique el conjunto de estados prohibidos del modelo. El autor demuestra la aplicabilidad del método con un ejemplo de la teoría de control supervisor de autómatas finitos.

Los ocho pasos para sintetizar el controlador son los siguientes:

a) Diseñe el modelo del sistema de eventos discretos con una Red de Petri sin controlar.
b) Defina los estados prohibidos de la Red.
c) Defina las subredes (conjunto de lugares y transiciones) de la PN asociadas con las marcas prohibidas.
d) Genere el árbol de alcanzabilidad de las subredes.
e) Del árbol de alcanzabilidad, identifique los marcados buenos y los prohibidos.
f) Especifique las transiciones controlables que en la evolución de

la PN hacen que las marcas vayan de un marcado permitido a un marcado prohibido.

g) Considere esas malas transiciones como eventos por separar del marcado bueno del árbol de alcanzabilidad. Use la teoría de regiones para generar el controlador.

h) Añada el controlador generado al modelo para obtener el Sistema Controlado por Petri Nets (CPNM).

G. B. Lee, 2004 [21], propone un método de diseño para generar automáticamente diagramas escalera a partir de las Redes de Petri controladas (CPN), utilizadas para el control de sistemas de eventos discretos. Antes de desarrollar su método, se realiza un repaso de la CPN, así como su definición formal. El método consiste en convertir la red tipo CPN en un sistema de ecuaciones booleanas. Posteriormente, cada ecuación generada es trasladada a una representación en diagramas de escalera, la cual es fácilmente implementada en lenguaje de programación LLD para PLC. A continuación, se presenta la ecuación para modelar las CPN:

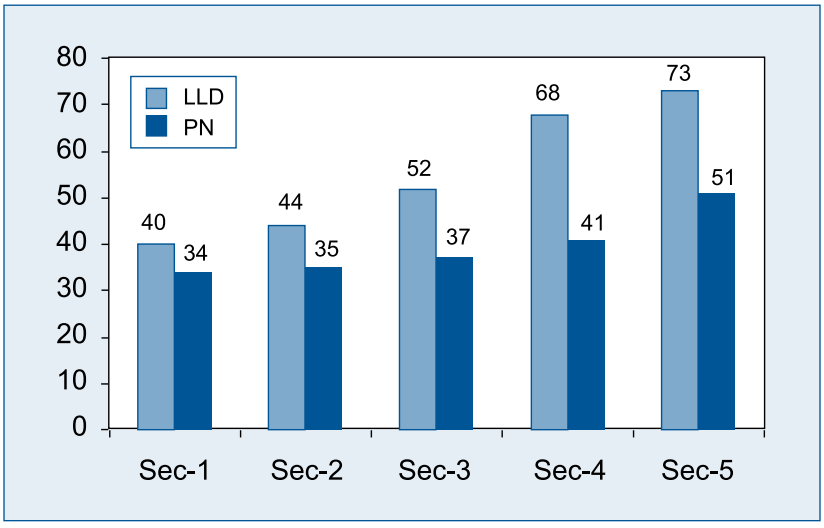



*Figura 13.* La gráfica muestra el consumo de estructuras If-Then por parte de LLD y PN para cinco secuencias que aumentan su grado de complejidad. [24]

$$P_i = \left( P_i + \sum_{\left\{ t_j \middle| t_j \in {}^\circ p_i \right\}} \left( \left\{ \prod_{p_k \middle| p_k \in {}^\circ t_j} P_k \cdot C_j \cdot E_j \right\} \right) \right) \cdot \prod_{\left\{ t_j \middle| t_j \in p_i^\circ \right\}} \left( \overline{\left\{ \prod_{p_k \middle| p_k \in {}^\circ t_j} P_k \cdot C_j \cdot E_j \right\}} \right)$$

Los términos $P_i$ y $P_k$ son lugares, $C_j$ son condiciones y $E_j$ corresponden a eventos. Además, se propone dos ecuaciones generales que modelan temporizadores y contadores.

Si bien en la bibliografía del artículo no aparece referencia a artículos de Pessen [35], existe una similitud en la implementación final de los circuitos que automatizan procesos productivos en ambos trabajos. Por ejemplo, esta ecuación genera un diagrama escalera donde un lugar almacena únicamente una marca. La desabilitación del lugar actual es hecha por la transición posterior. Por otro lado, en el método cascada de Pessen, la desabilitación de la bobina actual la realiza la bobina siguiente, lo cual podría traducirse como el lugar actual es deshabilitado por el lugar posterior, por lo que las soluciones en el diagrama LLD son similares. Finalmente, dicho artículo desarrolla un ejemplo práctico donde prueba el método. En este caso, la automatización de un sistema de mezclado de líquidos en una línea de producción.

J. S. Lee, 2004 [24], realizó un estudio comparativo entre dos métodos de diseño de automatización, la primera obteniendo un diagrama escalera con el método cascada expuesto en [34] y la otra, diseñando con redes APN según lo expuesto en [40]. Para esto, utiliza cinco procesos secuenciales por automatizar, donde el grado de complejidad es creciente. El autor realiza la automatización de las secuencias en diagrama escalera y en Red de Petri. Cada solución obtenida

la transforma a estructuras IF-THEN y realiza un conteo de las transformaciones realizadas, así como de la cantidad de operaciones lógicas requeridas.

Lee demuestra que conforme aumenta la complejidad del proceso, las Redes de Petri utilizan una menor cantidad de operaciones lógicas y transformaciones IF-THEN. La siguiente figura muestra el consumo de estructuras lógicas IF-THEN para los cinco procesos. Note que los algoritmos desarrollados por PN muestran menor número de transformaciones.

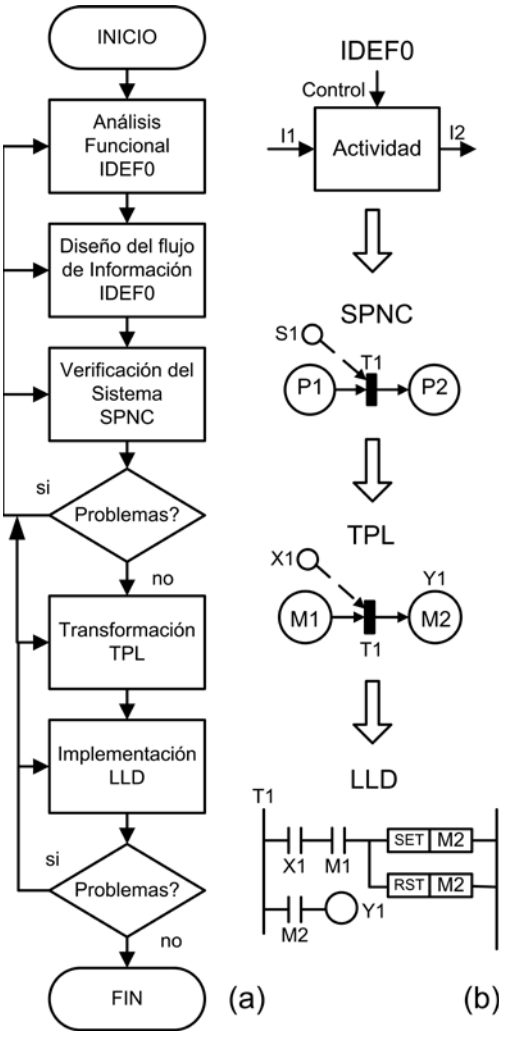



*Figura 14.* a) Proceso de diseño propuesto en [25]. b) Resultados de cada actividad del proceso [25].

S. S Peng publicó en el 2004 [33] un resumen de los distintos tipos de PN para el diseño de controladores de eventos discretos. Hizo un recuento de los tipos más comunes, entre ellos, las PN ordinarias, las PN temporizadas, las PN coloreadas y las PN controladas. Para las PN controladas, brinda cómo transformar las estructuras básicas de la red en un diagrama escalera. Se menciona que existen muchos subtipos de PN controladas y además se recalca que se debe realizar más estudios comparativos al respecto.

J.S. Lee, 2005 [25], propuso una nueva metodología para convertir las Redes de Petri controladas (CPN) en diagramas escalera. La novedad de este procedimiento es que incorpora el lenguaje IDEF0 definido por la NIST en 1993. Este lenguaje permite definir formalmente los requerimientos, antes de generar algún modelo del sistema. A partir de las IDEF0, se genera la CPN y se simplifica de tal forma que se establece una nueva familia de PN llamada SPNC. La transformación o síntesis de las SPNC a diagramas escalera es realizada con el método TPL o *Token Pass Logic*, el cual es definido por Uzam en [40]. La figura 14 muestra el proceso de diseño planteado por el autor.

M.R. Lucas y D. M. Tilbury, 2005 [26], propusieron un método para medir la complejidad y el tamaño de los programas para PLC que han sido diseñados bajo distintas metodologías. Los autores proponen seis índices para comparar algoritmos de control desarrollados con los métodos SIPN, PN y máquinas modulares de estado finito. Los autores concluyen que las métricas presentadas son una base para futuros trabajos y los resultados que arrojan no son indicativos de cuál técnica de modelado es la más apropiada para cada caso particular.

R. David, 2005 [5], presenta una monografía de las Redes de Petri, sus clasificaciones, sus usos, etc. En cuanto a las Redes utilizadas para modelar programas para

controladores industriales, presenta una exposición detallada de las redes de Petri Controladas, sus aplicaciones y ejercicios para resolver.

G. Cansever, 2006 [4], propone una nueva aproximación para el diseño de controladores de automatismos secuenciales utilizando Redes de Petri. El método propuesto es probado con el mismo problema que fue resuelto ocho años antes por Uzam [40] con las APN. La obtención final del código, según la norma IEC 1131 es producto de la utilización del software Expresso.

## Perpectivas futuras

Diversos autores coinciden en que deben existir metodologías para diseñar los algoritmos de control que son programados en PLCs, pero esto ha generado muchas variantes de Redes de Petri. El problema radica en la selección adecuada de la PN específica y su metodología. Por ejemplo, en [34] se señala que los estudios comparativos entre modelos son casi nulos, por lo que se requiere cotejarlos y clasificarlos para su óptima utilización.

Es necesario un gran esfuerzo para cerrar la brecha entre los sectores académicos y los sectores industriales que manufacturan PLCs. Por un lado, los fabricantes europeos han estandarizado un método de diseño para controladores secuenciales bajo el estándar IEC 848 en 1988 y la respectiva actualización en el 2002, generando el IEC 60848; también, casi en forma paralela, estandarizaron los lenguajes de programación de PLCs con la normativa IEC 1131-3 a partir de 1993 y diez años después la actualizaron a la norma IEC 61131-3. No obstante, estos esfuerzos conciernen sólo a un método particular derivado de las PN y a la implementación final. Los estándares no proponen métodos sistemáticos e integrales de diseño, donde la formalización, la validación, la verificación, etc. del sistema diseñado esté contemplados.

El sector académico ha tratado de cerrar esta brecha utilizando, por ejemplo, editores gráficos que modelan el conjunto sistema-controlador tal como en [4, 15], de modo que a partir de una APN, SIPN o software especializado, se genera código según la norma respectiva. Por otro lado, investigaciones como la de J.S. Lee [25] brindan métodos sistemáticos, pero éstos no son vinculantes para la industria.

Otro problema que no ha sido tratado es cómo desarrollar herramientas computacionales que permitan verificar formalmente los millones de programas que se ejecutan en los PLCs industriales. Esto último brindaría información sobre el cumplimiento de las especificaciones formales de cada programa.

Un área prioritaria en el futuro, dada la complejidad creciente de los procesos productivos, son la utilización de sistemas de control distribuido y el desarrollo de metodologías modulares con Redes de Petri para este tipo de sistemas. En este sentido, el interbloqueo entre módulos, como lo señala [34], es un área abierta a la investigación.

Se nota a lo largo de este trabajo que las técnicas algebraicas de simplificación y verificación de algoritmos realizados con PN no son utilizadas mayoritariamente, dado que prevalecen las técnicas gráficas. Se podría realizar estudios de costos sobre cuándo aplicar las técnicas algebraicas o gráficas.

En nuestras universidades es necesario replantear los cursos concernientes a la enseñanza del funcionamiento y programación de PLCs. En el plan de estudios se debe prestar mayor atención al levantamiento formal de los requerimientos y a su validación, al modelado formal de la planta y del controlador, a la validación de los modelos, a la implementación del software y a su verificación. Además, es necesario incorporar los criterios de calidad de software definidos en el ISO/IEC 9126, lo cual es un problema común

a algunas universidades foráneas, tal como lo plantea [34].

## Conclusiones

Las PN han demostrado desde hace más de 45 años su utilidad práctica en muchas áreas y en especial, en el modelado de sistemas complejos. A pesar de la gran cantidad de publicaciones existentes, su divulgación en el ámbito costarricense es muy limitada.

Este trabajo permite al lector una visión panorámica y cronológica de las familias de PN vinculadas con los PLCs. Además, permite generar conciencia en el lector sobre el impacto de las mismas en el diseño e implementación de algoritmos de control para controladores lógicos programables.

## Bibliografía


[1] E. Ávila, I. Benítez, J, Reinaldo. "Verificación de programas de PLCs modelados sobre Redes de Petri. Métodos gráficos vrs algebráicos". *XI Congreso Latinoamericano de Control Automático*, Cuba, 10-15 Mayo 2004.

[2] M. Bani Younis, G. Frey "Formalization of existing PLC programs: a survey" *Proceedings of CESA*, France, April 2003.

[3] Y. Cai, I. Nishi, T, Sekiguchi. *"Modeling by Petri nets with invariants for sequencial control systems". Electrical engineering in Japan*, Vol. 115, No 5. 1995.

[4] G. Cansever, I. B. Kucukdemiral. "A new approach to supervisor design with sequencial control Petri nets using minimization technique for discrete event systems". *International Journal of Advanced Manufacturing Technology*. Vol. 29, No. 11-12, pp 1267-1277. August 2006.

[5] R. David, H. Alla. *Discrete continuos and hybrid petri nets.* Ed. Springer-Verlag, Berlin, 2005.

[6] R. David. "Grafcet: A powerful tool for specification of logic controllers". *IEEE Transaction on Control Systems Technology*, Vol. 3, No. 3, pp 253-268, September 1995.

[7] P. Deussen "Partial order verification of Programmable Logic Controllers". *J-M Colom and M.Koutny: ICATPN 2001*, LNCS 2075, pp 144-163, 2001.

[8] G. Frey, L. Litz. "Verification and Validation of control algorithms by coupling a Interpreted Petri Net". *Proceeding of IEEE SMC´98*, San Diego, Vol. 1, pp 7-12, Oct 1998.

[9] G. Frey "Automatic Implementation of Petri nets based control algorithms on PLC" *Proceedings of ACC*, pp 2819-2823, June 2000.

[10] G. Frey. "Analysis of Petri nets based control algorithms: Basic properties". *Proceedings of ACC*, pp 3172-3176, June 2000.

[11] G. Frey. "PLC Programming for Hybrid Systems for SIPN." *Proceedings of ADPM*, pp 189-194, 2000.

[12] G. Frey, L. Litz. "Formal methods in PLC programming." *Proceeding of IEEE SMC 2000*, Nashville, pp 2431-2435, Oct 2000.

[13] G. Frey. "Software Quality in Logic Controller Programming". *Proceeding of IEEE SMC 2002*, Tunisia, Oct. 2002.

[14] G. Frey. "Hierarchical design of logic controllers using Signal Interpreted Petri Nets". *Proceeding of IFAC AHDS 2003*, France, pp 401-406, June 2003.

[15] G. Frey. "PLC programming with Signal Interpreted Petri Nets". *Proceeding of ICATPN*, LNCS 2679, pp 401-406, Netherlands, June 2003.

[16] E. García. *Automatización de Procesos Industriales.* Alfaomega. Mexico, 2001.

[17] L. E. Holloway, B. H. Krogh. "Syntesis of feedback control logic for a class of controllers Petri nets". *IEEE Transactions on Automatic Control*. Vol. 35, No. 5, pp 514-523, May 1990.

[18] L. E. Holloway, B. H. Krogh, A. Giua. "A survey of Petri nets methods for controlled discrete event systems". *Journal of Discrete Event Dynamic Systems: Theory and Applications*, Vol 7, No 2, pp 151-190. 1997.

[19] *IEC 60848 Ed. 2. Specification language Grafcet for sequential function charts.*2002.

[20] E. Jiménz. *Técnicas de automatización avanzada en procesos industriales.* PhD Tesis. Ed. Serv. Publicaciones de la Universidad de la Rioja. Logroño, 2001.

[21] K. John. *IEC 61131-3: Programming industrial automation systems. Concepts and programming languajes.* Springer-Verlag, Berlin, Germany, 2001.

[22] R. Johnsonbaugh, T. Murata. "Petri nets and Marked Graphs: Mathematical models of concurrent computation". *The American Mathematical Monthly*, Vol. 89, No. 8, pp. 552-566, Oct 1982.

[23] G. B. Lee, H. Zandong, J. Lee. "Automatic generation of ladder diagram with control Petri nets". *International Journal of Advanced Manufacturing Technology*. Vol. 15, No. 2, pp 245-252. 2004.

[24] J. S. Lee, P. L. Hsu. "A improved evaluation of ladder logic diagrams and Petri nets for sequence controller design in manufacturing systems". *International Journal of Advanced Manufacturing Technology*. Vol. 24, No. 3-4, pp 279-287. August 2004.

[25] J. S. Lee, P. L. Hsu. "A systematic approach for the sequence controller design in manufacturing systems". *International Journal of Advanced Manufacturing Technology*. Vol. 25, No. 7-8, pp 754-760. April 2005.

[26] M. R. Lucas, D. M. Tilbury. "Methods of measuring the size and complexity of PLC programs in different logic control design methodologies". *International Journal of Advanced Manufacturing Technology*. Vol. 26, No. 5-6, pp 436-447. September 2005.

[27] J. Martínez, M. Silva, M. Velilla. "Realización cableada de Redes de Petri binarias". *Questiió*, Vol. 6, No. 1, Març 1982.

[28] T. Mertke, G. Frey. "Formal Verification of PLC-Programs Generated from SIPN". *Proceedings of IEEE SMC 2001*, pp 2700-2705.

[29] I. Moon. "Modeling programmable logic controllers for logic verification". *IEEE Control Systems Magazine,* pp 53-59. April 1994.

[30] G. Mušič, D. Matko. "Petri nets based supervisory control of flexible manufacturing plants" *in Prepr. 8th IFAC Symp. On large Scales Systems: Theory & Applications*, Vol. 2, Rio Patras, Greece, 1998.

[31] T. Murata. "Petri nets and Marked Graphs. Mathematical models pf concurrent computation". *The American Matematical Monthly*, Vol. 89, No. 8, pp 552-566, Oct 1982.

[32] T. Murata. "Petri nets: Properties, Analysis and application". *Proceedings of the IEEE*, Vol 77, No 4, pp 541-580, April 1989.

[33] S. S. Peng, M. C. Zhou. "Sensor bases stage Petri nets modeling of PLC logic programs for discrete control design". *International Journal of Production Research*, Vol. 41, No. 3, pp 629-644, 2003.

[34] S. S. Peng, M. C. Zhou. "Ladder diagram and Petri net based discrete event control design methods". *IEEE Transactions on Systems, Man and Cybernetics-Part C*. Vol. 34, No. 4, Nov. 2004.

[35] D.W. Pessen. *Industrial Automation: Circuit Design & Components*. Ed. Wiley Interscience Publication, USA, 1989.

[36] J.L. Peterson. "Petri nets". *ACM Computer Survey*, Vol. 9, No. 3, pp 223-252, September 1977.

[37] S. Pettersson, B. Lennartson. "Hybrid Modeling Focused on Hybrid Petri Nets". *European Worshop on Hybrid Systems*, Grenoble (Fr), pp 303-309, 1995.

[38] L.W. Schruben. "Graphical model structures for discrete event simulation". *Proceeding of the 1992 winter simulation conference.* Ed J.J Swain.

[39] A. Taholakian, W. M. Hales. "PN⇔PLC: A methodology for designing, simulating and coding PLC based control system using Petri nets" *International Journal In Production Research*, Vol. 35, No. 6 pp 1743-1762, 1997.

[40] M. Uzam, H. Jones. "Discrete event control systems design using Automation Petri Nets and their ladder diagram implementation". *International Journal of Advanced Manufacturing Technology*. Vol. 14, No. 10 pp 716-728. Oct 1998.

[41] M. Uzam. "Synthesis of feedback control elements for discrete events systems using Petri nets models and theory of regions". *International Journal of Advanced Manufacturing Technology*. Vol. 24, No. 1-2, pp 48-69. 2004.

19. Describa el método de análisis de las propiedades dinámicas conocido como Matriz de Incidencia y Ecuación de Estado.

20. Describa el método de análisis de las propiedades dinámicas conocido como Reglas de Reducción.

21. Enuncie la propiedad estática de Vivacidad Estructural.

22. Enuncie la propiedad estática de Controlabilidad.

23. Enuncie la propiedad estática de Limitación o Acotado Estructural.

24. Enuncie la propiedad estática de Conservabilidad.

25. Enuncie la propiedad estática de Repetibilidad.

26. Enuncie la propiedad estática de Consistencia.

27. Plantee la notación de Murata y Pessen para clasificar redes de Petri-

28. Defina Maquinas de Estado.

29. Defina Grafico de Marcados

30. Defina Red de Libre Escogencia.

31. Defina Red de Libre Escogencia Extendida.

32. Defina Red de Escogencia Asimétrica o Red Simple.

33. Desarrolle un cuadro resumen sobre las interpretaciones de una red de Petri.