# Propuesta para diagnóstico y recuperación de fallas

**Karen Hernández Rueda**
Universidad de Guadalajara-CUCEA, Zapopan, Jalisco, México, khernandez@cucea.udg.mx

**María Elena Meda Campaña**
Universidad de Guadalajara-CUCEA, Zapopan, Jalisco, México, emeda@cucea.udg.mx

**RESUMEN**

Este artículo presenta una propuesta para el diagnóstico y recuperación de fallas de sistemas de eventos discretos usando redes de Petri. El objetivo es que estos sistemas puedan regresar a su comportamiento normal después que haya ocurrido una falla en los mismos, para evitar que se produzcan pérdidas tanto de recursos, tiempo y/o riesgos humanos. Aquí se define el problema y se hace una propuesta inicial para resolver el problema así como un ejemplo de aplicación. No se presentan resultados porque no es un trabajo concluido, sin embargo, se presentan los avances.

**Palabras claves:** diagnóstico de fallas, recuperación de faltas, detección de fallas, Sistemas de Eventos Discretos

**ABSTRACT**

This paper presents a proposal to fault diagnosis and recovery of discrete event systems. The objective is to make that this class of systems can to return to their normal behavior after an abnormal operation (fault) has occurred them, to avoid loss of as resources, time and/or human risks. Here it defines the problem and a inicial proposal to solve the problem so such an application example. There are not results here because this work is not finished, however, there are advances.

**Keywords:** fault diagnosis, fault recovery, fault detection, Discrete Event System

## 1. INTRODUCCIÓN

Un sistema falla cuando no cumple con su especificación. Dependiendo de la complejidad y la importancia del sistema, estas fallas pueden ser toleradas o provocar una catástrofe. Actualmente hay muchos sistemas complejos, estos sistemas inevitablemente pueden fallar, no importa cómo se realizó su diseño o cuál técnica de control se aplica o que tan buenos son los operadores. El problema de detección y recuperación de fallas es una necesidad existente, no sólo por el impacto que pueden tener en los sistemas sino en la misma sociedad, cuando pueden evitar una catástrofe.

La seguridad y la confiabilidad son los dos factores principales que motivan el estudio del problema de diagnóstico y recuperación de fallas en los sistemas de eventos discretos, así como lo relacionado con evitar las situaciones de riesgo que pueden ser catastróficas. Otra motivación es asegurar que los sistemas continúen trabajando, ya que algunos sistemas, como plantas de energía, no deben dejar de operar porque pueden generar interrupciones en el servicio que causan graves impactos económicos y un riesgo, incluso, como sucedió con el reactor nuclear en Japón tras el TSUNAMI en Marzo de 2011.

La necesidad de un diagnóstico exacto y oportuno de las fallas del sistema, en aras de la seguridad, confiabilidad y economía, ha provocado un gran interés en el área de diagnóstico de fallas tanto en la industria y en el mundo académico. Un gran esfuerzo en la investigación ha sido y está siendo invertido en el diseño y el desarrollo de sistemas automatizados de diagnóstico. Se han propuesto una variedad de sistemas que difieren tanto en su marco teórico, en su diseño y filosofía de aplicación. Desde el punto de vista conceptual la mayoría de los actuales

métodos de diagnóstico de fallas se pueden clasificar como: 1) los métodos basados en árboles de fallas, 2) los métodos cuantitativos, basados en modelos analíticos 3) los sistemas expertos, 4) los métodos basado en modelos de razonamiento, y 5) los métodos basados en modelos de sistemas de eventos discretos (Sampath et al, 1998). Nuestro interés en este trabajo es estudiar los sistemas de eventos discretos, usando el método basado en el modelo del sistema, y comparando el comportamiento actual del sistema y el esperado (modelo matemático) para el diagnóstico de fallas usando redes de Petri (Xiaoli1 et al 2009).

Existen muchos factores que pueden causar que un sistema falle, y el principal factor es cuando hay algún error en su diseño. Sin embargo, cuando un sistema está operando, resulta costoso en tiempo y/o dinero volver a rediseñarlo, por lo que se deben buscar alternativas para prevenir las fallas o en su caso, hacer que el sistema se recupere o continúe operando aún con las presencias de éstas fallas. La recuperación ha sido abordada a través de varios enfoques diversos. Por ejemplo, las basadas en la redundancia, como, la redundancia con los chekpoints (Cheng et al 2004), habilidades, capacidades y conocimientos de la redundancia de agentes (Mirian et al 2002). Otros trabajos se basan en el cálculo de las secuencias de acción, por ejemplo, mediante la búsqueda de secuencias de control (Bjreland and Fodor, 1998), (Seilonen et al 2002) o por secuencias de recuperación predefinidas en (Anton et al 2002), (López and Alami 1999). Nuestro interés de investigación es encontrar en línea los mecanismos capaces de adaptar el controlador, el planificador o la reconfiguración del sistema a las nuevas condiciones de operación después de haberse emitido situaciones de fallas, utilizando las redes de Petri.

A continuación se presentan algunos conceptos básicos para entender el problema (sección 2). Luego se plantea la conceptualización del problema (sección 3). Posteriormente, se presenta una propuesta (de forma general) para dar solución al problema (sección 4) y finalmente, se hace una conclusión.

## 2. CONCEPTOS CLAVE

A continuación se presentan algunas definiciones básicas para entender el problema de diagnóstico y recuperación de fallas en sistemas de eventos discretos.

**Un sistema de evento discreto** (DES) es un sistema dinámico, cuyo espacio de estados es numerable, aunque posiblemente finito, y donde el estado cambia abruptamente en respuesta a eventos que ocurren, en general, de manera asíncrona (Hopcroft and Ullman, 1979), (Silva 1985). Y se definen formalmente en (Cassandras and Lafortune 1999) como: un sistema de estado discreto y dirigido por eventos, es decir, su evolución depende enteramente de la ocurrencia de eventos discretos asíncronos sobre el tiempo.

Como los DES pueden presentar faltas, fallas o errores y se desea que estos se diseñen como sistemas tolerantes a faltas, es necesario definir los siguientes conceptos:

**Una falla** sucede cuando el comportamiento del sistema se desvía del comportamiento requerido por su especificación. Es decir, el sistema no puede proveer el servicio deseado.

**Un error** es la diferencia entre el comportamiento especificado y el comportamiento actual del sistema.

**Una falta** es una desviación del comportamiento requerido de un elemento o subsistemas, sin que esta desviación provoque un cambio en la especificación o comportamiento del sistema. Es decir, el sistema sigue cumpliendo con su contenido. Las faltas se pueden clasificar de acuerdo al tiempo que se encuentren presentes en el sistema como permanentes, transitorias o intermitentes. A continuación se definen:

- **Faltas permanentes** son las que inician en un tiempo particular y permanecen en el sistema hasta que éste se repara (Burns and Wellings, 2001).
- **Faltas transitorias** son de duración limitada, causadas por un al funcionamiento temporal del sistema o debido a alguna interferencia externa (Jalote, 1994).
- **Faltas intermitentes** son de tipo transitorias que ocurren de vez en cuando.

**Un sistema tolerante a faltas** es un sistema que puede manejar la presencia de faltas dentro del mismo a través de diferentes mecanismos (por ejemplo, redundancia) y continuar trabajando, aunque con cierto nivel de degradación o en el peor de los casos, detener la operación del sistema de forma temporal y segura.

Además de los conceptos anteriores, es importante definir las redes de Petri (RP) ya que los DES se pueden representar con ellas y se usarán para formalizar los problemas de diagnóstico y de recuperación de fallas. Acorde con (Silva 1985), las RP son un conjunto de herramienta adecuada para describir el comportamiento de los DES, ya que pueden representar las características de concurrencia, sincronización y decisión exclusiva, así como las relaciones causales entre eventos. Asimismo proveen un formalismo matemático para analizar las propiedades de modelado de sistemas como vivacidad, limitación, seguridad y falta de cerraduras. Una RP consiste de una estructura de red (un dígrafo bipartito), una descripción de estado (el marcado) y una regla de transición (el juego de marcas). Un ejemplo de una RP se muestra en la siguiente figura 1:

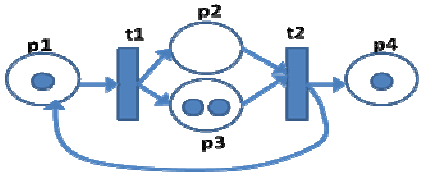



**Figura 1: Ejemplo de una red de Petri**

La red se representa por dos clases de vértices: círculos que representan lugares (p1,p2,p3), a los que se les asocian acciones o salidas del sistema que se desea modelar y barras o rectángulos (t1,t2) que representan transiciones, a los que se les asocian eventos y acciones o salidas. Los lugares a su vez pueden estar marcados, es decir, se distribuyen marcas (puntos) dentro de los lugares. Un marcado inicial sería una distribución inicial de marcas. Los lugares de entrada/salida son lugares cuyos arcos llevan a (salen de) una transición $t_j$ y se consideran de entrada a (salida de) $t_j$. ¿Cómo funciona? La evolución de marcas consiste en una simple regla de dos partes:

1. transición habilitada: los lugares de entrada deben al menos tener tantas marcas como el peso de los arcos de entrada
2. disparo de transición: elimina tantas marcas como el peso del arco de los lugares de entrada a una transición habilitada y agrega tantas marcas a los lugares de salida como el peso del arco del lugar de salida lo indica.

**Una estructura de una red de Petri** G es un dígrafo bipartito representado por la 4-dupla G= (P,T,I,O) donde:

- $P=\{p_1, p_2, ..., p_n\}$ y $T=\{t_1, t_2, ..., t_m\}$ son conjuntos finitos de vectores llamados lugares y transiciones, respectivamente.
- $I(O) : P \times T \rightarrow Z^+$ es una function que representa el peso de los arcos que van de lugares a transiciones (transiciones a lugares), donde $Z^+$ es un conjunto de enteros no negativos.

Por lo general, ${}^\bullet t_j$ representa el conjunto de lugares $p_i$, tal que $I(p_i,t_j)\neq 0$ y $t_j^\bullet$ el conjunto de lugares tal que $O(p_i,t_j)\neq 0$. Analogamente, ${}^\bullet p_i$ representa el conjunto de transiciones $t_j$ tal que $O(p_i,t_j)\neq 0$ y $p_i^\bullet$ representa el conjunto de transiciones $t_j$ tal que $I(p_i,t_j)\neq 0$. Sea X(Y) un subconjunto de lugares de P (transiciones de T), entonces ${}^\bullet X$ y $X^\bullet$ (${}^\bullet Y$ y $Y^\bullet$) denotan el conjunto de lugares $p_i$ (transiciones $t_j$) tal que $I(p_i,t_j)\neq 0$ y $O(p_i,t_j)\neq 0$, respectivamente, para todo $t_j \in X$ ($p_i \in Y$). La matriz de incidencia de G es $C= [c_{ij}]$, donde $c_{ij}= O(p_i,t_j) - I(p_i,t_j)$. La función de marcado $M: P \rightarrow Z^+$ es un mapeo desde cada lugar a un entero no negative representado por el numero de marcas ( representado en forma de puntos) dentro de cada lugar. El marcado de una red de Petri PN es generalmente expresado como un vector de entrada n.

**Un sistema de red de Petri** es el par (G,Mo), donde G es la estructura de RP y Mo es la distribución de marcado inicial. En un sistema RP, una transición $t_j$ es habilitada con el marcado $M_k$ si para todo $p_i \in P$, $M_k(p_i) \geq I(p_i,t_j)$. Una transición habilitada $t_j$ puede ser disparada alcanzando un nuevo marcado $M_{k+1}$, calculado como $M_{k+1} = M_k + Cv_k$ (ecuación de estado), donde $v_k(i)=0$, $i\neq j$, $v_k(j)=1$.

La alcanzabilidad de un conjunto R (G, Mo) de una PN es el conjunto de todos los posibles marcados alcanzables desde Mo disparando solo las transiciones habilitadas.

**Una red de Petri interpretada** (RPI) es la 4-tupla Q=(G,Σ,λ,φ). G=(G,Mo) es una estructura de la RP y Mo es el marcado inicial. $\Sigma = \{\alpha 1, \alpha 2, \ldots \alpha r\}$ es un alfabeto de la entrada de la red, donde $\alpha_i$ es el i-ésimo símbolo de entrada del alfabeto. $\lambda: T \rightarrow \Sigma \cup \{\varepsilon\}$ es una función de etiquetado de las transiciones con las restricciones: $\forall t_j \in T$, $j \neq k$ T si $\forall p_i$ $I(p_i,t_j) = I(p_i,t_k) \neq 0$ y ambos $\lambda(t_j) \neq \varepsilon$, $\lambda(t_k) \neq \varepsilon$, entonces $\lambda(t_j) \neq \lambda(t_k)$. En este caso ε representa un evento del sistema. Existe una matriz φ de q x n dimensiones, tal que $y_k = \varphi M_k$ es el mapeo del marcado $M_k$ en un vector de observación q-dimensional. La columna $\varphi(\bullet,i)$ es el vector elemental $e_h$ si el lugar $p_i$ tiene asociado el sensor h; o el vector nulo si $p_i$ no tiene asociado ningún sensor. En este caso un vector elemental eh es el vector q-dimensional con todas sus entradas iguales a cero, excepto en la entrada h, que es igual a 1. Un vector nulo tiene todas sus entradas iguales a 0.

## 3. CONCEPTUALIZACIÓN DEL PROBLEMA

El objetivo en el diseño y la construcción de un sistema tolerante a fallas es garantizar su funcionamiento continuo en su conjunto, incluso en la presencia de fallas y la degradación del sistema. La mayoría de los sistemas cuenta con un controlador que generada las entradas al sistema para que se comporte como la especificación (lo que se desea que realice), para asegurar su buen funcionamiento. Los controladores se diseñan pensando que el sistema nunca falla pero ¿qué pasa si falla?

Cuando el sistema falla (durante la operación), como puede verse en la figura 2, la diferencia entre la salida del modelo de la especificación (salida supuesta de datos) y la salida de datos del sistema es diferente de cero (ek≠0), por lo que existe un desvío de la especificación. Así que el conjunto sistema-controlador-especificación (mostrado en la figura 2) debe ser adaptado o reconfigurado para continuar con la operación del sistema como fue diseñado. Aquí el problema radica entonces en decidir qué se debe reconfigurar ¿el controlador o la especificación del sistema?

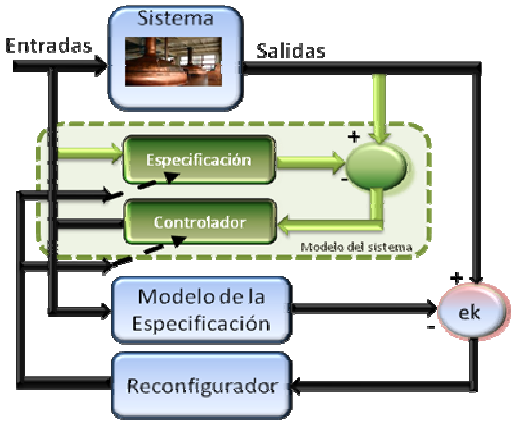



**Figura 2: Diagrama para reconfigurar el controlador o modificar las especificaciones del sistema**

Además, debe advertirse de la reconfiguración originada después de que hubo un error para que se reparen las máquinas o subsistemas que no funcionan. En los sistemas, sin embargo, no existe tal mecanismo de reconfiguración. De hecho hace falta desarrollar un marco teórico que defina bajo qué circunstancias un sistema puede detectar un error y recuperarse del mismo con la menor intervención humana posible y degradando lo menos posible al sistema.

Se identifican, por lo tanto, la necesidad de resolver dos problemas: un problema de diagnóstico de fallas y un problema de recuperación de fallas. A continuación se definen en qué consisten estos problemas.

### 3.1 DIAGNÓSTICO DE FALLAS

Este problema consiste en detectar cuándo un sistema alcanza un estado de falta, considera la identificación del componente fallido y de los límites de corrupción (evaluación del daño). Si bien la verificación de modelos garantiza que el sistema realiza las tareas para las cuales fue diseñado (especificaciones), todavía pueden existir fallas en el sistema; ya sean debidas a su implementación final, mala calidad en los materiales usados, interferencias externas, etc. Si se ha detectado un error en el sistema, entonces sabemos que hay faltas en nuestro sistema y que ha ocurrido algún tipo de falla. Sin embargo, el error se pudo haber propagado a otras partes del sistema en el lapso de tiempo ocurrido desde que hubo la falta en el sistema y la detección del error. La detección del error y el confinamiento del daño (identificación y delimitación del daño) se consideran parte del diagnóstico, y son etapas de un sistema de tolerancia a faltas, conocidas como confinamiento y evaluación del daño. La detección de errores y el diagnóstico de fallas son abordados en (Bjäreland and Fodor, 1998) en (López and Alami, 1990) en (Sampath et al 1998) en (Seilonen et al. 2002) y en (Alcaraz 2007), entre otros. Aunque ya existen algunos resultados para su análisis, éstos resultan muy complejos computacionalmente hablando y se limitan a sistemas muy pequeños.

### 3.2 RECUPERACIÓN DE FALLAS

Este problema consiste en permitir que el sistema siga funcionando aún en presencia de faltas, aunque tenga alguna degradación. La recuperación de faltas o errores es el proceso en el cual se remueven las faltas del sistema. Una vez que el error ha sido detectado y se ha diagnosticado la falta, lo siguiente es remover el error. Después de aplicar las dos primeras fases de tolerancia a faltas (detección y confinamiento del error) tenemos un sistema en un estado libre de errores. En este punto, es importante saber cuál tipo de faltas (permanentes, transitorias o intermitentes) originó el error. En el caso de que tengamos una falta transitoria, después de la recuperación del error, el sistema puede ser simplemente reiniciado. Como la falta ya no existe, entonces no volverá a ocurrir el error. Sin embargo, si el error fue causado por una falta permanente, además de remover el error del sistema, se debe evitar la utilización del componente fallido, actividad conocida como tratamiento de la falta y continuación del servicio. La recuperación del error y, el tratamiento de la falta y servicio continuo, se consideran parte de la recuperación de fallas y también son etapas de un sistema de tolerancia a faltas. La recuperación de errores es abordada en (Bjäreland and Fodor, 1998) (Cheng et al 2004) (López and Alami, 1990) (Seilonen et al 2002) (Zhou & Dicesare, 1989). Aunque ya existen resultados para dar solución a éste problema, sólo se considera la modificación del controlador del sistema con diferentes métodos.

## 4. PROPUESTA

Existen muchos factores que pueden causar fallas en un sistema y el primer factor es cuando existen errores de diseño. Sin embargo, cuando un sistema ya está operando y resulta costoso en tiempo y/o dinero volver a diseñarlo, es necesario que existan alternativas que permitan prevenir las faltas o en su caso, recuperarse o continuar operando a pesar de la ocurrencia de las mismas. A continuación presentan los puntos propuestos para resolver el problema planteado:

1. caracterizar la propiedad de diagnosticabilidad para abarcar más tipos de sistemas, considerar los tipos las faltas pueden ocurrir en el sistema y definir las metodologías de diseño para realizar el diagnóstico en línea.
2. tratar el problema de recuperación como de modificación de cualquiera de los tres siguientes mecanismos; controlador, planificador o reconfigurador, con el interés de encontrar nuevas formas de realizar la misma tarea.

El objetivo de la propuesta es contar con un modelo de diagnóstico y recuperación de fallas de un DES como se muestra en la siguiente figura 3. Bajo la hipótesis de que es posible implementar un sistema de diagnóstico fallas y de recuperación de fallas de DES con las RP, con una mínima degradación en el rendimiento del sistema. El modelo de diagnóstico debe detectar y localizar fallas en la línea en un tiempo finito (aunque el sistema no sea diagnosticable) basado en el modelo de diagnóstico presentado en la tesis doctoral de Ruiz (Ruiz, 2007). Y el

modelo de recuperación debe tener la posibilidad de modificar ya sea el controlador, planificador o reconfigurador del sistema para mejorar su desempeño del sistema.

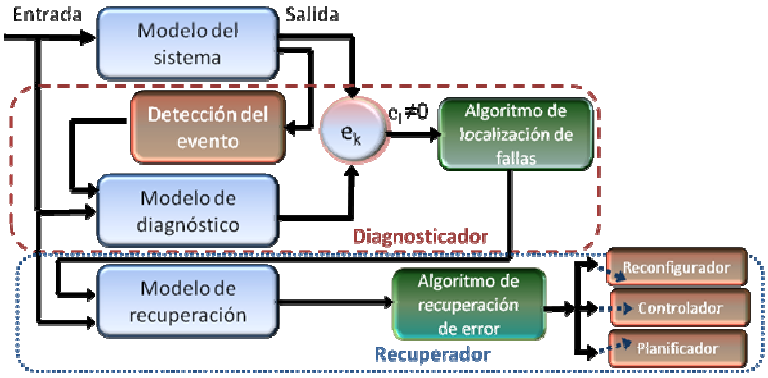



**Figura 3: Esquema propuesto para diagnóstico y recuperación de fallas**

Cuando el sistema se encuentra trabajando en línea, se usa una parte del esquema de la figura 3, correspondiente al algoritmo diagnosticador que se basa en la propuesta de (Ramírez et al, 2004). El funcionamiento del esquema diagnosticador es el siguiente; cuando ciertas entradas manipulables y no manipulables se dan en el sistema, esas entradas generan un cambio en el marcado del modelo del sistema. En la figura 3 se observa que esas entradas afectarán la salida del modelo de sistema, que contiene el comportamiento normal y de falla del sistema, y también afectarán la salida del modelo diagnosticador que sólo contiene el buen comportamiento del sistema. En el caso de que ninguna falla ocurra en el sistema, entonces el error ($e_k$) entre la salida del modelo del sistema y del modelo diagnosticador será cero ($e_k$=0), lo cual indica que ninguna falla se encuentre presente en el sistema. Ahora, si una falla ocurre, entonces el error entre el modelo del sistema y el modelo diagnosticador será diferente de cero ($e_k \neq 0$), lo cual implica la presencia de falla (se detecta) en el sistema, e inmediatamente se activa el algoritmo de localización de fallas, el cual se encargará de encontrar dónde se encuentra la falla en el sistema. Una vez que se localice la falla, se activará el algoritmo de recuperación de error para resolver el problema que haya ocurrido. Dependiendo del tipo de falla, el recuperador de la figura 3, decidirá cuál mecanismo de recuperación (controlador, planificador, reconfigurador) se necesita modificar para restaurar nuevamente el sistema. A continuación se explican estas tres etapas de forma general y se muestra un ejemplo visual.

### 4.1 DETECCIÓN DE LA FALLA

La existencia de una falla en el sistema se detecta cuando el error $e_k \neq 0$, por lo que es necesario realizar el cálculo del error $e_k$ para determinar si hubo o no una falla. Considerando el trabajo de (Ramírez et al 2004), si ya se tiene el modelo del comportamiento normal del sistema ($Q^N$,$M^N{}_0$), el del diagnosticador ($Q^d$, $M^d{}_0$) así como el marcado inicial del diagnosticador ($M^d{}_0 = B^T$ (φMo) donde $B^T$ es un vector de qx1 entradas no negativas) y reglas de disparo de las transiciones del diagnosticado (Si una transición $t_j \in T^R$(transiciones de riesgo) se habilita en (Q,Mo) y $\lambda(t_j)$se activa en (Q,Mo) entonces $t_j$ debe ser disparada en ($Q^d$,$M^d{}_0$). Así, si $t_j$ no se dispara en (Q,Mo), entonces existe un error en (Q,Mo) y el vector de observación del modelo del sistema y el vector de observación del modelo del diagnosticador será diferente) entonces se puede calcular el error.

La ecuación que representa la ecuación de estado de la red que contiene la parte medible del modelo del sistema junto con el modelo-diagnosticador es:$[ M^d{}_k, \varphi M_k]^T = [ M^d{}_0, \varphi M_0]^T + [ C^d, \varphi C]^T v_k$ *(ec.1)*. $C^d = B^T \varphi^N C^N$ es una matriz de incidencia de ($Q^d$, $M^d{}_0$) y $C^N$ es matriz de incidencia de ($Q^N$,$M^N{}_0$). Por lo que sustituyendo en la ec. 1 se obtiene que $C^d - B^T \varphi C = 0$ *(ec.2)*, tal que $[1-B]^T [C^d, \varphi C]^T = 0$ *(ec.3)*. Así que $[1-B]^T [M^d{}_k, \varphi M_k] = cte = 0$. Como el modelo-diagnosticador y el modelo del sistema no se sincronizan, entonces el error $e_k$ se calcula como: $e_k = M^d{}_k - B^T(\varphi M_k)$. Cuando no ocurre ninguna falla, entonces $e_k$=0, pero cuando ocurre

una falla, se tiene que $e_k \neq 0$, donde ese error será una columna de $C^d$ y debido a que todas las columnas de $C^d$ son diferentes entre sí y el vector nulo, se determinará que falta ocurrió.

### 4.2 LOCALIZACIÓN DE LA FALLA

Una vez que un error $e_k$ se detecta en el diagnosticador, es necesario ubicar dónde se encuentra la falla. Por lo que se usará el siguiente algoritmo propuesto en (Ruiz, 2007):

***Entradas:***
$M_k$ − *k-ésimo marcado del modelo del sistema,*
$M_k^d$ − *k-ésimo marcado del modelo-diagnosticador,*
$e_k$ − *error entre la salida del modelo-diagnosticador y la transformación de la salida del modelo del sistema,*
$\varphi(M_k)$ − *el k-ésimo vector de observación del modelo del sistema,*
$\varphi(M_{k-1})$ − *k-1-ésimo vector de observación del modelo del sistema,*
$\varphi C$ − *Matriz de incidencia evaluada en* $\varphi$.
***Salidas:***
$p$ − *lugar de falla aislado, por lo tanto el elemento culpable del sistema,*
$M_f$ − *el marcado de falla donde se encuentra el modelo del sistema,*
$t_f$ − *transición de falta que provocó el estado de falla.*

---

***Constantes:***
$C^d$- *es la matriz de incidencia del modelo diagnosticador de* $RPIVT$,
$i$ = *índice de la columna de* $C^d$ *tal que* $C^d(\bullet, i) = e_k$,

---

- *Si* $t_i \in T^R$ *entonces*

  1. $-\forall p \in \bullet t_i, M_k(p) = 0,$
     $-\forall p \in t_i\bullet, M_k(p) = 0,$
     $-\forall p^F \in (\bullet t_i)\bullet\bullet \cap P^{FP}, M_k(p^F) = 1,$
     $-M_f = M_k.$
     ***Regresar*** $(p,\ M_f,\ "falta\ permanente"((\bullet t_i)\bullet \cap T^{FP}\ ).$
- *sino*

  1. *Si* $t_i \in T^{FI}$ *entonces*
     *Si el error fué forzado por una transición de falta intermitente prematura, se indica* $tiempoincorrecto = 1$, *de lo contrario se indica con* $tiempoincorrecto = 0$ *que fué por una falta intermitente tardía.*
     *Si* $tiempoincorrecto = 1$ *entonces*
     − $\forall p \in \bullet t_i,\ M_k(p) = 1,$
     − $\forall p \in t_i\bullet,\ M_k(p) = 0$
     − $M_f = M_k,$
     − ***Regresar*** $(p,\ M_f,$ *falta intermitente temprana* $t_i)$
     *sino*
     − $\forall p \in t_i\bullet,\ M_k(p) = 1,$
     − $\forall p \in \bullet t_i,\ M_k(p) = 0,$
     − $M_f = M_k,$
     − ***Regresar*** $(p,\ M_f,$ *falta intermitente tardía* $t_i)$.

*sino*

$q = \varphi(M_k) - \varphi(M_{k-1})$ *(una columna de* $\varphi C$*),*

$i$ = *índice de la columna de* $\varphi C$ *tal que* $\varphi C(\bullet, i) = q$, *entonces* $t_i$ *fue disparada;*

$-\forall p \in t_i\bullet\ M_k(p) = 1$,

$-\ \forall p \in \bullet t_i\ M_k(p) = 0$,

$-\ M_f = M_k$,

$-$ ***Regresar*** *(*$p$, $M_f$, *"falta de control"*$t_i$*)*

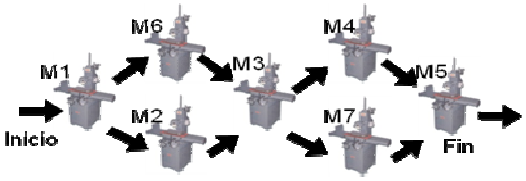
Donde RPIVT es una RPI con Ventas de Tiempos. Los subconjuntos de transiciones $T^{FP}$, $T^{FC}$, $T^{FI}$, y $T^{N}$ representan transiciones de: falta permanente, falta de control, falta intermitente; y transiciones normales, respectivamente.

### 4.3 RECUPERACIÓN DE LA FALTA

Para esta parte se toma como base el trabajo realizado en (Alcaraz, 2007) que considera recuperar el sistema a través de la modificación su controlador (controlador por regulación de salidas). Cuando una falta es detectada y diagnosticada, el diagnosticador envía el vector de lugares de faltas así como un vector de transiciones fallidas, al reconfigurador. El reconfigurador realiza un procedimiento completo para la modificación parcial del controlador con respecto al vector de transiciones fallidas que se obtiene del vector. El procedimiento de reconfiguración del controlador inicia verificando si el sistema tiene la redundancia necesaria para realizar la modificación parcial del controlador. Si esto es posible, se verifica si el controlador tiene la propiedad de reconfigurabilidad. Después, el reconfigurador efectúa el procedimiento de reconfiguración del controlador obteniendo el nuevo controlador.

### 4.4 EJEMPLO

Un sistema de fabricación de un producto X, está constituido por 7 máquinas que lo ensamblan, como puede verse la figura 4a. El modelo del sistema usando las RP se muestra en la figura 4b, los lugares representan las máquinas de ensamble y las transiciones indican la terminación del ensamblado de cada máquina. La figura 4b representa el modelo del comportamiento normal del sistema, es decir, funciona como fue especificado.



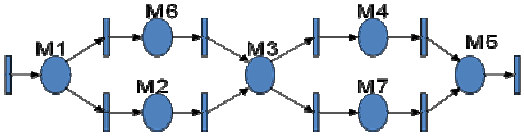
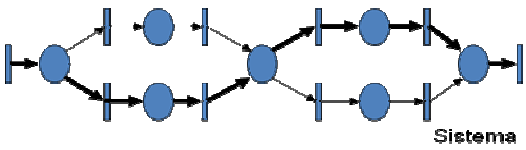




**Figura 4a: Sistema de fabricación del producto X** **4b: Modelado del sistema en RP**

El controlador del sistema eligió una ruta de producción que reduce el costo de fabricación del producto X. La ruta se resalta con flechas en negritas y puede verse en la figura 5a. Para que se pueda realizar el diagnóstico de fallas es necesario que también se obtenga el modelo del diagnosticador, éste se obtiene a través de una copia del modelo del comportamiento normal del sistema que ya fue optimizado (reducción del costo de fabricación).



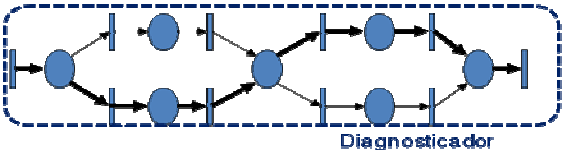




**Figura 5a: Modelo del sistema optimizado** **5b: Modelo del diagnosticador**

Una vez que se tienen ambos modelos, se deja trabajar al sistema así como el algoritmo diagnosticador. Si el sistema sigue funcionando de forma normal entonces se puede ver que conforme el producto se va ensamblando por la ruta óptima de producción (M1-M2-M3-M4-M5), los lugares se van marcando como puede verse en la figura 6. Al mismo tiempo este seguimiento se realiza en el diagnosticador.

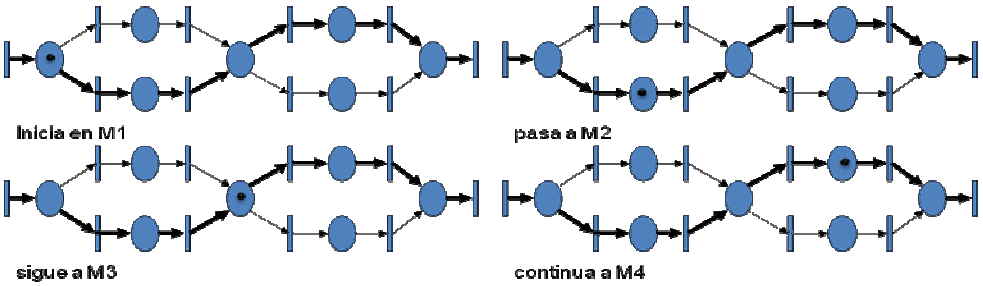



**Figura 6: Seguimiento del ensamble del producto X**

Cuando se presenta una falla en el sistema, hay una diferencia entre el seguimiento de las marcas del modelo del sistema y el diagnosticador (que se calcula con $e_k$). Por ejemplo, si la máquina M2 no tiene un plan de mantenimiento es posible que falle y ésta falla se detecta cuando no coincide el marcado del modelo del diagnosticador y del sistema como puede verse en la figura 7. La figura 7a representa el modelo del sistema con falla, el lugar de color naranja indica que la máquina M2 no está funcionando, y la figura 7b representa el diagnosticador, que funciona como el sistema lo debe hacer si éste no presenta ninguna falla. Como puede observarse, el producto X se queda en M2 y no pasa a la siguiente máquina M3 para continuar el ensamble.

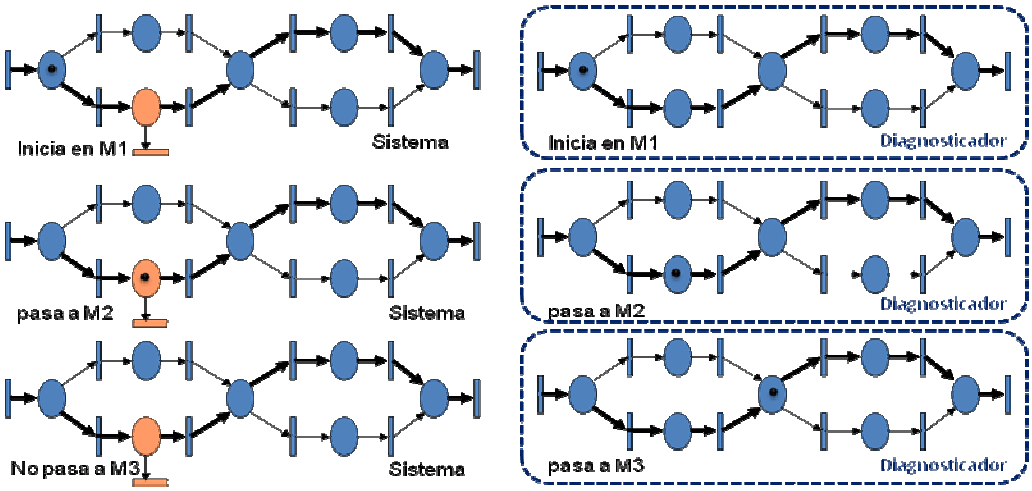



**Figura 7a: Sistema con falla en M2 7b: Seguimiento del diagnosticador**

Como $e_k \neq 0$, entonces se detectó la falla y el algoritmo de localización indicará que la falla ocurrió en la máquina 2. Posteriormente, se activará el algoritmo de recuperación, en este caso, elegirá reconfigurar el controlador del sistema para que seleccione una ruta alternativa para que se termine de ensamblar el producto X y el diagnosticador también debe modificar la ruta optimizada por la que el controlador haya elegido. Así el sistema continuará fabricando el producto X con la nueva ruta alternativa (M1,M6,M3,M4,M5), es decir, el sistema se recuperó después de una falta, como puede verse en la figura 8.

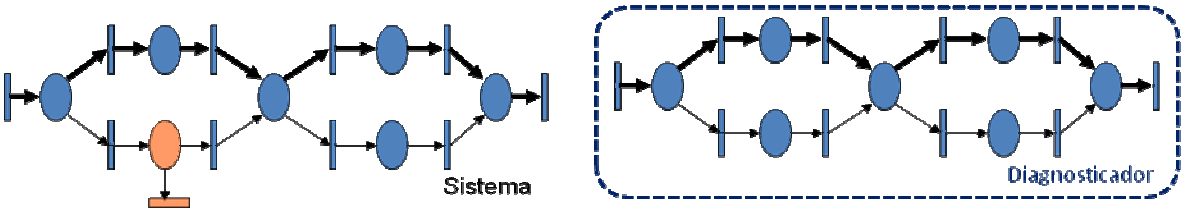



**Figura 8: Recuperación del sistema**

Es importante señalar que ése es un caso sencillo y para abordar sistemas más complejos es necesario analizar y determinar las propiedades de las RP que deriven las condiciones que se requieren para que implementar un algoritmo de recuperación de faltas de forma automática.

## 5. CONCLUSIONES

En este capítulo se presentó una propuesta para realizar diagnóstico y recuperación de fallas basada en los modelos presentados por (Ramírez et al, 2004) y (Ruiz 2007) usando las RPI, con la consideración de mejor estos trabajos para que el diagnosticador se aplique a sistemas más complejos y el recuperador tenga diferentes alternativas para recuperar el sistema después de una falla. Como trabajo futuro se contempla la modificación del algoritmo de localización de fallas para sistemas que no sean diagnosticables y la implementación del algoritmo de recuperación de fallas, que tome en cuenta la recuperación del sistema usando su controlador, asimismo como la conveniencia de reconfigurar el planificador o las especificaciones del propio sistema; para ello es necesaria la caracterización de la redirección de los recursos y replanificación de tareas en términos RPI. Además, se describió brevemente el formalismo de las RP en virtud de que el modelado de las RP ofrecen una naturaleza gráfica y un soporte matemático para representar de manera clara y compacta los comportamientos complejos de los DES. Y se explicó brevemente tres de las etapas del sistema tolerante a faltas como son la detección de la falla, la localización de la falla y la recuperación del sistema que se contemplan dentro la solución del problema de diagnóstico y recuperación de fallas. También se presentó un ejemplo de cómo se realiza el diagnóstico y la recuperación de fallas.

## REFERENCIAS


Meera Sampath, Stephane Lafortune, Kasim Sinnamohideen, and Demosthenis C. Teneketzis (1998). "Diagnosis of Discrete-Event Systems", *IEEE Transactions on Automatic Control*, Vol. 43, no. 7, pp. 908-929.

Wang Xiaoli1, Chen Guangju Xie yue Guo Zhaoxin (2009). "Fault Detection and Diagnosis Based on Time Petri Net", *The Eighth International Conference on Electronic Measurement and Instruments*, pp.3-259 - 3-26.

F. Cheng, H. Yang and J. Lin (2004). "Development of holonic information coordination systems with failure-recovery considerations". *Proc. of the IEEE Trans. On Automation Science and Engineering*, vol 1, pp. 58-72.

M.S. Mirian, M.N. Ahmadabadi and Z. Navadi (2002). "A decision-marking based approach for fault-handling in multi-agent systems". *Proc. of the 9th. Int Conf. on Neural Information Preccessing,* vol 4, pp. 1905-1909.

M. Bjreland and G. Fodor and D. Driankov (1998). "Ontological control". *9th Int. Workshop on Principles of Diagnosis. Loch Awe, Scotland, June.*

I. Seilonen, P. Appelqvist, A. Halme and K. Koshinen (2002). "Agent-based approach to fault- tolerance in process automation system". *Proc of the IEEE Int. Simposium on Intelligent Control*, pp. 473-478.

O. Anton, B. Lecher and G. Reinchart ( 2002)."Modelling of faults and fault recovery: An essential aspect of mechatronic system design". *Annals of 2002 CIRP Design Seminar.*

E. López and R. Alami (1999). "A failure recovery scheme for assembly workcells". Proc of the IEEE Int. Conf. on Rob. And Automation. Cincinnati, OH, USA, vol 1 pp. 702-707.

M.C. Zhou and F. Dicesare. "Adaptive design of Petri net controllers". Proc. Of the IEEETrans. On Systems, M

C.G. Cassandras and S. Lafortune (1999). "Introduction to Discrete Event Systems". *Kluwer Academic Publishers.*

J. E. Hopcroft and B.D. Ullman (1979). *Introduction to Automata Theory, Lenguajes, and Computation.* Addison-Wesley.

M. Silva (1985). Las redes de Petri: en la automática y la informática. Editorial AC, Madrid, España.

Elvia Ruiz Beltrán (2007). " Diagnostic Diagrams Discrete Event Systems". Ph.D. thesis, CINVESTAV-unidad Guadalajara, México. pp. 1- 150.

Mildrad Alcaraz (2007). " Recuperación de faltas en sistemas de manufactura discreto". Ph.D. thesis, CINVESTAV-unidad Guadalajara, México. pp. 1- 145

A. Ramírez-Treviño, E. Ruiz-Beltrán, I. Rivera-Rangel and E. López-Mellado (2004). "Diagnosability of Discrete Event Systems. A Petri Net Based Approach". *Proceedings of the IEEE International Conference on Robotic and Automation*, pp. 541-546, 2004.

### *Autorización y Renuncia*

*Los autores autorizan a LACCEI para publicar el escrito en las memorias de la conferencia. LACCEI o los editores no son responsables ni por el contenido ni por las implicaciones de lo que esta expresado en el escrito.*