CINVESTAV-IPN
CENTRO DE INVESTIGACIÓN Y ESTUDIOS AVANZADOS DEL IPN

LABORATORIO DE INGENIERÍA ELÉCTRICA Y CIENCIAS DE LA COMPUTACIÓN

SISTEMAS TOLERANTES A FALLAS EN SISTEMAS DE MANUFACTURA FLEXIBLE.

TESIS QUE PRESENTA
ING. CARLOS ORTEGA MOODY

PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS

EN LA ESPECIALIDAD DE
INGENIERÍA ELÉCTRICA

BECARIO DE CONACYT
Guadalajara, Jal., 1998

Bibliográficos

Agradecimientos.

A todas las personas que contribuyeron al buen término de esta tesis mis más sinceros agradecimientos.

Agradezco a Antonio Ramírez por su labor de asesor y su paciencia por corregir esta tesis, pero en especial por su valiosa y sincera amistad.

Deseo expresar mis agradecimientos a todos los compañeros del Laboratorio de Ingeniería Eléctrica y Ciencias de la Computación, en los cuales he encontrado una gran amistad.

2

Prefacio

En la industria existen procesos que no deben detener su actividad en caso de una falla porque las perdidas serían cuantificables. Por ejemplo un avión, en vuelo no debe detener su operación. Para tratar de diseñar sistemas tolerantes a fallas es decir sistemas que pueden seguir operando aún en la presencia de fallas, se utilizan dos técnicas, la tolerancia a fallas con redundancia dinámica y la tolerancia a fallas con redundancia estática [13] [22]. La técnica a utilizar depende del tipo de sistema donde se desea aplicar.

La tolerancia a fallas con redundancia estática consiste en tener N réplicas del sistema, un circuito llamado votante que compara la salida de las réplicas y toma como respuesta correcta la respuesta de la mayoría. Es utilizada para esconder los efectos de las fallas, para que éstas no se manifiesten como error en el sistema. Se utiliza en sistemas digitales, sistemas de compúto [13], [23].

La tolerancia a fallas con redundancia dinámica consiste en tener un conjunto de rutinas que sólo se van a activar cuando se detecte una falla en el sistema, a diferencia de la tolerancia a fallas con redundancia estática que enmascara la falla. Cuando ocurre una falla el sistema puede sufrir una degradación en su funcionalidad. Consiste de tres etapas detección del error o detección de la falla, diagnóstico de fallas y recuperación del error o recuperación de la falla.

Para la detección del error se han utilizado en los sistemas continuos, es decir los sistemas que se pueden modelar mediante ecuaciones diferenciales, diferentes aproximaciones para la detección de fallas (detección del error) usando su modelo matemático han sido desarrolladas en los últimos años. Por ejemplo [25], [26].

La detección del error o de fallas basado en un modelo consiste en obtener si existen discrepancias no aceptables entre las señales de entrada al sistema, las señales de salida con las señales generadas por el modelo.

Para la detección de fallas en sistemas de comunicaciones digitales tenemos la verificación de códigos son usados para comprobar que los datos sean los correctos, se basan en la información redundante contenida dentro del dato [13].

En la etapa de diagnóstico las aproximaciones que se utilizan son las basadas en el modelo y las basadas en síntomas.

La aproximación basada en síntoma consiste en encontrar cuándo cambia una variable que se encontraba en un comportamiento normal, dependiendo de la variable que cambio mediante métodos de clasificación o de inferencia se determina que elemento esta fallando.

La adquisición del conocimiento es difícil los ingenieros del conocimiento tienen problemas para la extracción y clasificación del conocimiento una vez adquirido, se deben de tomar en cuenta todas las posibles combinaciones. [20], [19].

El aproximación basada en el modelo utiliza un modelo del dominio del conocimiento estructural y funcional acerca de determinado sistema. La adquisición de los datos puede ser más sencilla. Durante el diseño del sistema a ser modelado, se hace un examen detallado de la estructura del sistema y mucha de la información para crear el modelo del sistema está disponible desde que el sistema fue diseñado.

Para la etapa de recuperación del error [13] propone dos métodos. La recuperación del error hacia adelante que consiste en llevar el sistema de un estado de error a un estado válido que se encuentra adelante. La recuperación hacia atrás que consiste en llevar el estado de error ha un estado válido que se encuentra antes del estado de error. Por ejemplo cuándo las operaciones a realizar a una pieza de acero no fueron las deseadas, se funde la pieza y se vuelve a procesar.

En [10] además de estos dos métodos propone dos métodos más, que son la entrada condicionada, este consiste en que si existe un elemento dañado no utilizarlo hasta que este reparado. El otro método es el de ruta alterna consiste en replanificar la estructura del Sistema de Manufactura Automático, por eso esté método no es tomado por otros autores [13] como un método de recuperación del error convencional.

Índice General

Capítulo 1

Panorama General.

Resumen: En este capítulo se presenta a los **Sistemas Tolerantes a Fallas** (STF) y los principales problemas que surgen al tratar de construirlos. Se definen los Sistemas Dinámicos de Eventos Discretos. También se presenta a los **Sistemas de Manufactura Flexible** (SMF) que es el campo donde se aplican en este trabajo los STF. De los SMF se estudiarán cómo nacen históricamente y la necesidad de contar con ellos. Finalmente se introduce formalmente el Problema de Tolerancia a Fallas.

1.1 Introducción

En la industria siempre han existido sistemas que deben operar continuamente, es decir, sistemas que nunca deben detener su funcionamiento y sólo deben ser detenidos ("disparados" en términos industriales) para mantenimiento y bajo una calendarización estricta. Generalmente se debe a que las constantes de tiempo de estos sistemas son muy grandes. Por ejemplo, los generadores de vapor de las termoeléctricas pueden tardar hasta 20 días en llegar a producción; otro ejemplo son los altos hornos que no es posible detener su operación ya que pueden llegar a tener fracturas en sus paredes.

También existen restricciones para detener un proceso a mitad de operación, ya que tales disparos de planta pueden dañar seriamente al sistema o poner en peligro la integridad de la planta y de los operadores. Tomemos como ejemplo la compañía Kendy, del norte de la Ciudad de México, que fabrica ropa de licra, cuando su proceso se detuvo involuntariamente, todo el polímero solidificó dentro de las tuberías dañando parte de la planta. Otro caso crítico se presenta en un avión, en vuelo no debe detener su operación.

En el mundo actual, que ha sido definido como el mundo de la información, detener las redes de comunicación sería catastrófico. Otros procesos de manufactura no deben ser detenidos simplemente por la pérdida de dinero que resulta el tener una planta sin funcionar.

Para tratar este tipo de sistemas se diseñan sistemas confiables, *que pueden seguir operando aún en la presencia de fallas.* Las aproximaciones para realizar estos tipos de sistemas van desde replicar hasta adaptar o modificar al sistema cada vez que varíe su estructura. Dentro de este espectro existen variantes que se mencionarán en este reporte. También cabe destacar que los sistemas que trabajan en la industria son sistemas en su mayoría en tiempo real, por tanto el sistema tolerante a fallas debe contemplar estas características.

1.2 Sistemas Tolerantes a Fallas.

Muchos intentos se han hecho para lograr sistemas confiables, algunas aproximaciones se basan en replicar los sistemas de manera que un circuito extra detecte en qué réplica se localiza la falla y la enmascara. Otra aproximación se basa en el principio de que el sistema fallará y que lo mejor es diseñar al sistema para que siga trabajando aún en presencia de fallas modificando su estructura. Estas dos aproximaciones son usadas en conjunto para aumentar la confiabilidad del sistema. Antes de seguir con el estudio de los STF introduciremos algunas definiciones básicas.

Definición 1.2.1 ***Sistema Tolerante a Fallas (STF).*** *Es un sistema que puede seguir operando aún en presencias de fallas por medio de una acción de control, aunque tal vez con una degradación en su desempeño.*

Definición 1.2.2 ***Confiabilidad.****[21] Es una medida de la capacidad de un sistema para comportarse de acuerdo a sus especificaciones.*

Una forma de medir la confiabilidad es [13]:

$$R(t) = e^{-\lambda t}$$

donde λ es el número de fallas por unidad de tiempo.

Definición 1.2.3 *Falla.* *[21] Es una desviación inaceptable de algún parámetro del sistema. El parámetro puede ser una variable física o el resultado de un algoritmo de computo.*

Definición 1.2.4 *Error.* *[13] Es la manifestación de una falla en el comportamiento externo del sistema.*

Al tratar de diseñar sistemas confiables se presentan dos posibles soluciones. A continuación se mencionan junto con sus principales características:

- **Sistemas tolerantes a fallas con redundancia estática**. También conocida como Sistemas Libres de Fallas [22]. Consiste en tener N réplicas del sistema, un circuito llamado votante que compara la salida de las réplicas y toma como respuesta correcta la respuesta de la mayoría. Es utilizada para esconder los efectos de las fallas, para que éstas no se manifiesten como error en el sistema. A continuación se presentan algunas de sus características:
  - Diseñar el circuito votante dependiendo del tipo de señales del sistema.
  - Replicación N veces del sistema.
  - El sistema es más pesado.
  - Cada réplica hay que diseñarla con diferentes herramientas y usar diferentes dispositivos, para garantizar que no fallarán del mismo modo.

- **Sistemas tolerantes a fallas con redundancia dinámica**. Consiste en tener un conjunto de rutinas que sólo se van a activar cuando se detecte una falla en el sistema, a diferencia de la tolerancia a fallas con redundancia estática que enmascara la falla. A continuación se presentan algunas de sus características:
  - Cuando ocurre una falla el sistema puede sufrir una degradación en su funcionalidad.
  - El sistema se hace más complejo, ya que la etapa de tolerancia a fallas sólo debe activarse cuando hay un error.
  - Una vez que se ha detectado el error se debe analizar al sistema para saber qué componente está dañado.
  - bloque de recuperación para garantizar que el sistema regrese a un estado especificado.

Se puede tener esquemas híbridos combinando las redundancias estática y dinámica.

El presente trabajo abordará a los Sistemas tolerantes a fallas y su aplicación a los Sistemas de Manufactura Flexible cuando son vistos como Sistemas Dinámicos de Eventos Discretos. Por está razón introducimos a continuación un pequeño resumen de éstos últimos.

1.3 Sistemas Dinámicos de Eventos Discretos.

Los Sistemas Dinámicos de Eventos Discretos (**SDED**) son sistemas cuyo espacio de estados es numerable y posiblemente infinito, la representación de este tipo de sistemas no es posible hacerla mediante ecuaciones diferenciales como es el caso de los sistemas continuos ya que los **SDED** se caracterizan por la presencia de cambios abruptos en los eventos y por consecuencia en los estados por lo cual, en este caso no se puede introducir el concepto de derivada.

Los diferentes eventos que representa un **SDED** pueden ser etiquetados con elementos de algún alfabeto $\sum$. Estas etiquetas usualmente indican el fenómeno físico que causa el cambio de estado. En esta tesis sólo se desea modelar el flujo de las piezas en el sistema y cuando las máquinas se encuentran trabajando. Las etiquetas serían las operaciones del sistema. Por ejemplo "transportar pieza", "fresar la pieza", etc. Un estado está representado por los valores de las características de los elementos. Por ejemplo que el torno se encuentra procesando una pieza o si un robot se encuentra se encuentra listo para tomar una pieza, si hay una pieza lista para ser procesada por una fresa o si la pieza se transformó en un producto final.

El estudio de los **SDED** es necesario porque una gran variedad de sistemas pertenecen a esta clase, por ejemplo los sistemas de cómputo (operativos, de procesamiento de información, etc.), sistemas de comunicación, sistemas de transporte y sistemas de producción (dentro de éstos una familia muy importante conocidos como Sistemas de Manufactura Flexible) entre otros.

Formalmente un **SDED** se define como se menciona a continuación:

Definición 1.3.1 *Los Sistemas Dinámicos de Eventos Discretos(SDED) son aquellos que están compuestos por elementos que manejan entidades discretas, es decir numerables y diferenciables entre sí. Su funcionamiento está caracterizado por una sucesión finita o infinita de estados delimitados por eventos. Cuando ocurre un evento se genera un cambio de estado, generalmente éstos cambios se presentan de manera asíncrona.*

A continuación presentamos como surgieron los SMF.

1.4 Sistemas de Manufactura Flexible.

Debido a la alta competencia que existe entre los países industrializados, uno de los mayores intereses que se ha despertado es el de incrementar la productividad utilizando la tecnología de manufactura de forma eficiente; ejemplos: producción continua de una mezcla variable de productos, flexibilidad para tratar los cambios en el diseño de un producto. Mejorar la

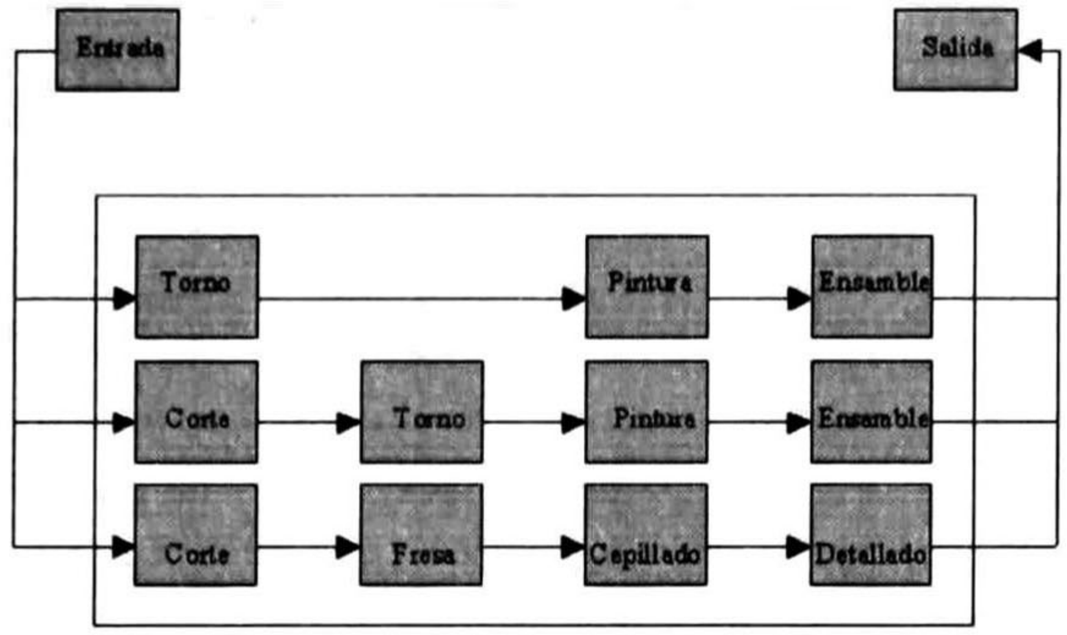


Figura 1.1: Sistema de producción Flow-Shop.

tecnología requiere de conceptos innovadores y soluciones originales a problemas que puedan presentarse, por eso el estudio de los Sistemas de Manufactura Flexible(**SMF**) debe ser abordado para hacer más eficiente la planta productiva.

Existen dos tipos de sistemas de producción estándar [2], de éstos los **SMF** aprovechan sus principales ventajas.

- **Flow-Shop:** Un Flow-Shop es un sistema de producción dedicado a la fabricación de diferentes tipos de productos con un alto volumen de producción. Este sistema presenta la misma secuencia de operaciones para cada línea de producción, en consecuencia el equipo es de propósito especial y no de propósito general. La inversión en máquinas y herramientas especializadas es alto y es por eso que cuando se requiere cambiar de producto, ésto lleva mucho tiempo e inversión. En cierto sentido, podemos decir que la destreza de producción se ha transferido del operador a la máquina. Este tipo de producción se representa en la figura 1.1 donde un proceso presenta la misma secuencia de operaciones para cada pieza. Ejemplos de este tipo de producción podemos verlos en productos plásticos, partes de automóvil, etc.

- **Job- Shop:** Un Job-Shop es un sistema de producción dedicado a la fabricación de una gran variedad de pequeños volúmenes de partes. La filosofía de producción Job-Shop es usada para satisfacer órdenes específicas y de gran variedad, por lo tanto, la maquinaria y equipo son diseñados para soportar cambios y especificaciones nuevas. Debido a estas características la habilidad de las personas y máquinas que trabajan en un Job-Shop es relativamente alta de tal manera que pueden desarrollar una gran variedad de tareas. La figura 1.2 muestra cómo diferentes partes fluyen en diferentes trayectorias a través de las máquinas. Ejemplo de este tipo de producción podemos verlos en la fabricación de herramientas.

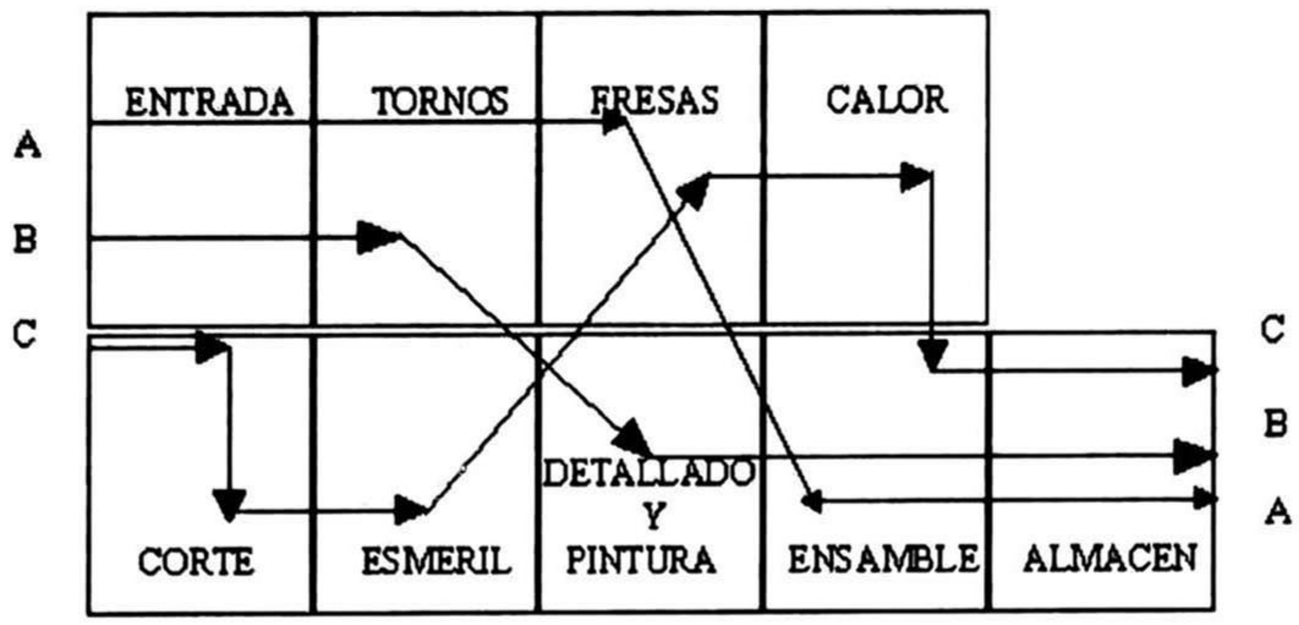


Figura 1.2: Sistema de producción Job-Shop.

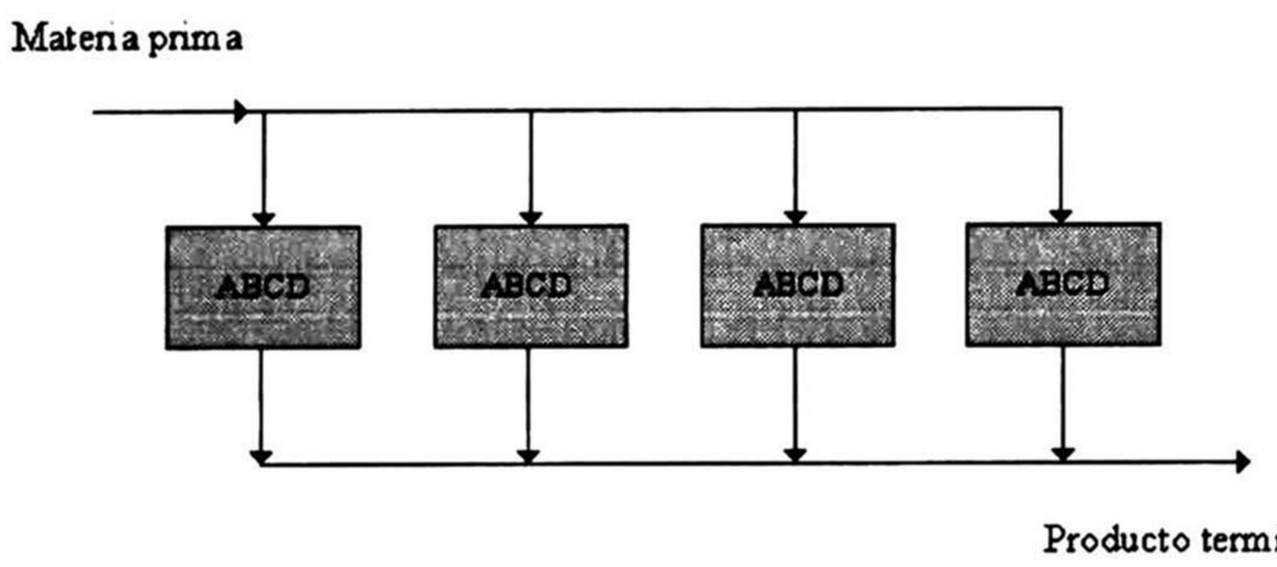


Figura 1.3:

- **SMF:** Un **SMF** es un sistema de producción que aprovecha las principales ventajas que ofrecen los anteriores sistemas de producción. Esta nueva filosofía de producción ha sido diseñada tomando la eficiencia de las líneas de producción bien balanceadas (altos volúmenes de producción), y al mismo tiempo se utiliza la flexibilidad que los job-shop ofrecen para maquinar simultáneamente diferentes tipos de partes. Lo que le da el nombre de **SMF** es la capacidad de procesar una gran variedad de diferentes tipos de productos o partes simultáncamente y hacer cambios rápidos en la fabricación de un producto a otro, además el sistema de transporte con que cuenta un **SMF** permite rutas alternas que compensan fallas que podrían ocurrir en diferentes máquinas.

En la figura 1.3 mostramos un esquema generalizado del funcionamiento de un **SMF**, donde A,B,C y D son las operaciones que recibe un tipo de pieza para su terminado.

Un Sistema de Manufactura Flexible está compuesto de un grupo de máquinas, donde se desarrollan las operaciones de producción, interconectadas con un sistema de almacenamiento y un sistema de manejo de materiales automático, todo ésto controlado por un sistema de cómputo.

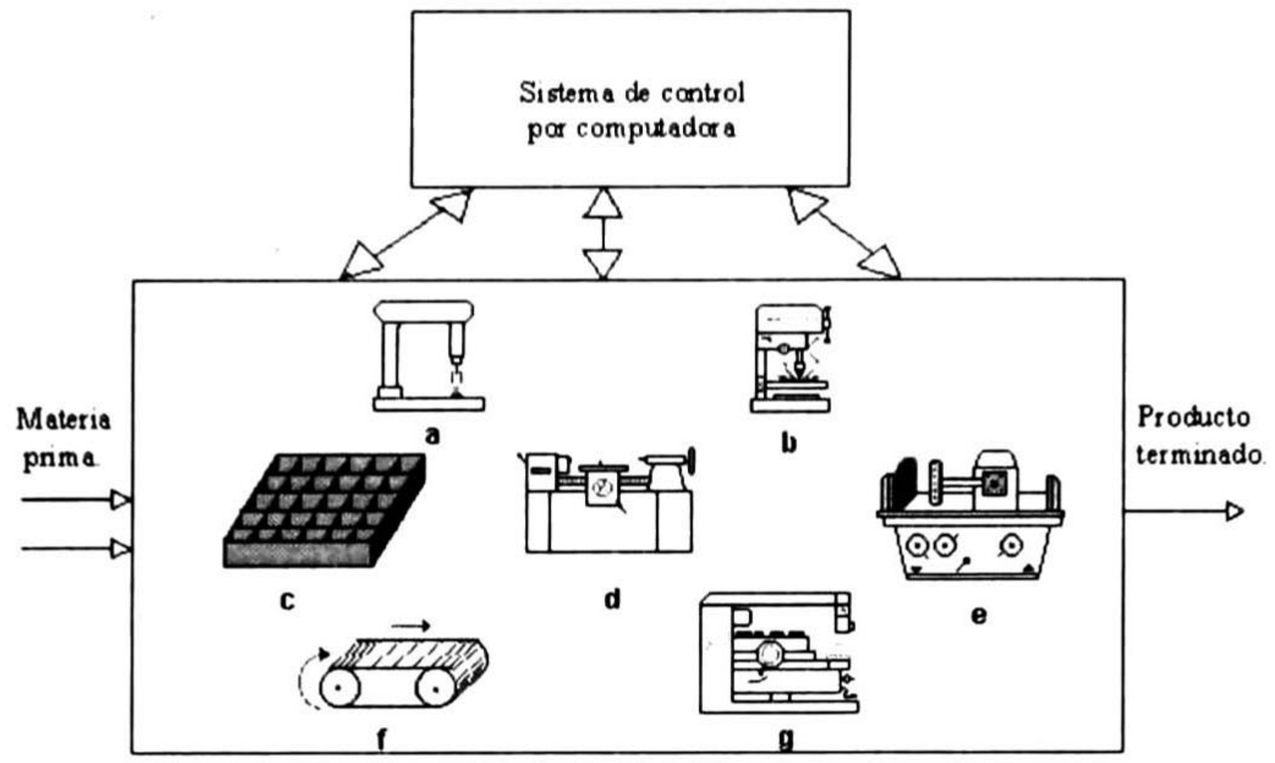


Figura 1.4: Como está compuesto un SMF.

Todo **SMF** tiene como objetivo la fabricación de una serie de productos finales, los cuales son obtenidos mediante los trabajos realizados sobre las piezas que llegan al sistema. Cada pieza tiene asociado un trabajo (plan de proceso), que es una serie de tareas (operaciones) que deben ser realizadas por los diferentes dispositivos del **SMF** sobre dicha pieza. Debido a las características discretas de las piezas procesadas y los dispositivos, el **SMF** puede ser modelado como un **SDED**. En un **SMF** real, el orden y tipo de trabajos que pueden recibir las piezas dependen del tipo de máquinas y de cómo éstas se relacionan entre sí, teniéndose lo que se llaman las restricciones de un **SMF**, las cuales vienen impuestas por la tecnología usada, por el plan de proceso o por la exclusión entre operaciones.

Ejemplo: en la figura 1.4 se muestra algunos elementos de un SMF: "a" robot, "b" taladro de banco, "c" pallet, "d" torno, "e" rectificadora de superficie plana, "f" banda transportadora, "g" fresadora, además del Sistema de control por computadora.

Los elementos de un SMF se clasifican como se indica a continuación [17]:

Definición 1.4.1 ***Efectores*** *(Ef): Son elementos del SMF capaces de transportar/almacenar las piezas o partes en procesamiento entre las diferentes máquinas y almacenes.*

Como ejemplo podemos mencionar a los robots, vehículos autoguiados (AGV's), bandas transportadoras, mesas giratorias, etc.

Definición 1.4.2 ***Máquinas*** *(M): Son un conjunto de dispositivos flexibles de transformación capaces de realizar diferentes operaciones y que disponen de almacenes de herramientas intercambiables junto con todos los mecanismos de acceso a las mismas.*

Como ejemplo, podemos tener tornos, máquinas fresadoras, esmeril, taladros, etc.

Definición 1.4.3 ***Sitios de almacenamiento*** *(SA): Son lugares cuya función es retener una pieza temporal o permanentemente, sin recibir ningún tratamiento adicional.*

Definición 1.4.4 ***Sistema de Computo*** *(SC): Es un sofisticado sistema de control y toma de decisiones. Fundamentalmente, toma de decisiones sobre que piezas deben entrar y que rutas deben seguir las piezas dentro del* ***SMF****, todo ésto de acuerdo con una política de producción establecida.*

Definición 1.4.5 ***Operaciones*** *(Op): Es la colección de las diferentes actividades que realizan los efectores o las máquinas como por ejemplo tornear, taladrar, etc.*

Un elemento adicional que se pueda tomar en cuenta para los SMF (para propósitos de este trabajo no se va a considerar) es la labor humana. Las personas también intervienen en el funcionamiento de un SMF realizando ciertas operaciones dentro del sistema como puede ser el suministro de la materia prima al sistema, la descarga de partes terminadas (o ensambles), etc.

Una definición alternativa para un SMF, utilizando las definiciones previas, es la siguiente:

Definición 1.4.6 ***Sistema de Manufactura Flexible(SMF):*** *Un SMF es una 6-tupla SMF= $\{Ef, M, SA, SC, Op, \phi\}$ teniendo como objetivo principal producir una familia de productos, donde:*

$Ef = \{x$ / x es un efector$\}$ = {robots, bandas transportadoras, vehículos guiados automáticamente (AGV's), mesas giratorias, grúas, elevadores,etc.}

M ={x / x es una máquina herramienta }={tornos, máquinas fresadoras, esmeriles, taladros, cepillos, rectificadoras,etc.}

SA ={x / x es un almacén automático}

SC ={x / x es un sistema de cómputo(control)}

Op ={x / x es una operación}={tomar, poner, avanzar, fresar, taladrar , tornear, ensamblar, alimentar/descargar, etc.}

$\phi : Ef \cup M \rightarrow \{2^{|Op|}\}$. Es una función la cual asigna a cada efector, máquina un conjunto de operaciones.

$2^{|Op|}$ = conjunto potencia de Op.

1.5 Planteamiento del problema.

La aproximación propuesta para implementar un sistema tolerantes a fallas depende del tipo de sistema. Para esta tesis se escogió un sistema tolerante a fallas con redundancia dinámica. En un SMF existen diferentes formas de realizar un producto. Además las máquinas deben ser aprovechadas de la mejor forma posible, por el costo de la máquina, el costo del mantenimiento, etc. Pensar en redundancia estática resulta imposible ya que no se pueden tener tres máquinas produciendo las mismas piezas para después tomar sólo una de ellas para enmascarar una falla sí se presentará; además la utilización de los recursos sería muy baja. Por otro lado, el hecho de que existan diferentes secuencias de procesado para realizar un producto resulta apropiado para un esquema de tolerancia a fallas con redundancia dinámica, ya que permite que todas las máquinas o efectores se utilicen productivamente,

y en caso de que se detecte un error, el sistema de tolerancia a fallas puede optar por otra secuencia de procesado.

Dado $R(SMF)$ el conjunto de estados alcanzables por el SMF; $S(SMF)$ el conjunto de estados válidos (considerados correctos en las posibles evoluciones del sistema), claramente $S(SMF) \subset R(SMF)$ y el espacio de estados de la señal de error $\mathcal{E} = \{e_i \mid e_i = r_e - s_j, e_i \neq [\vec{0}]\ \forall r_e \in R(SMF), \forall s_j \in S(SMF)\}$.

Los problemas de tolerancia a fallas que se estudian en esta tesis se definen de la siguiente manera:

- 1.-*Conocer cuando el sistema alcanza un estado r_e que no sea correcto, ya sea porque no está especificado o no apareció en el tiempo esperado.*

  - $e_i \neq [\vec{0}], e_i \in \mathcal{E}$.

- 2.*Encontrar la función, $f : \mathcal{E} \rightarrow \{2^{Ef \cup M}\}$ que dado un error determina en qué elementos del SMF se encuentran las fallas.*

- 3.-*Si $e_i \neq [\vec{0}], e_i \in \mathcal{E}$ se debe buscar si existe alguna secuencia de operaciones ρ que lleve el estado alcanzado r_e del SMF, a un estado válido s_j.*

El sistema de control de SMF se divide en control supervisor, control óptimo, y control robusto. El control supervisor analiza la eliminación de secuencias de eventos que llevan a estados indeseables en el SMF. El control óptimo selecciona las secuencias de eventos que son las mejores en algún sentido, por ejemplo el mínimo tiempo de procesado de una pieza. Finalmente el control robusto el sistema pueda seguir operando en presencia de fallas [8]. El control robusto se lleva a cabo por el STF.

- Hipótesis 1. En esta tesis el control óptimo del SMF, se lleva a cabo por medio de un *scheduling* dinámico [16], es decir que se calcula el scheduling óptimo con respecto a algún criterio, cada vez que ocurre un cambio en los estados del SMF.

El esquema general del SMF y su Sistema de Control es el mostrado en la figura 1.5.

El control supervisor analiza el modelo del SMF y la eliminación de secuencias de eventos que llevan a estados indeseables en el SMF es decir *eis*, el resultado es un modelo que realiza todo lo que el SMF puede hacer salvo los estados indeseables. El control óptimo utiliza el modelo entregado por el supervisor para no tomar en cuenta estados indeseables y encontrar secuencias de eventos que son óptimas en algún sentido U, como por ejemplo tiempo mínimo de procesado de una pieza, etc. Una vez calculada las secuencias óptimas de eventos se mandan al SMF para su realización. El control robusto o STF está comprobando los estados del SMF *eds*, para ver si no se ha alcanzado un estado de error (no especificados por el control supervisor), dependiendo de U y de las señales w_i que indican hacia donde debe evolucionar el estado *eds*. Si se presenta algún estado de error el control robusto debe identificar que máquina o efector tiene la falla mediante instrucciones de prueba *di* y los datos resultados

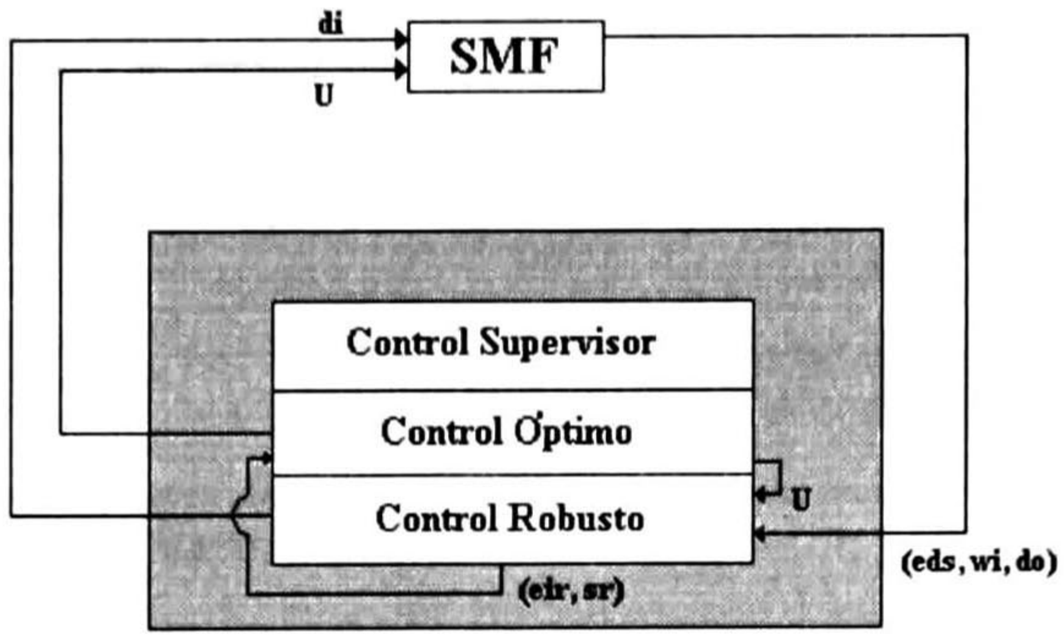


Figura 1.5: SMF y su Sistema de Control.

obtenidos a través de los sensores *do* . Debe eliminar las secuencias que habilitan al evento se localizó la falla *eir* y se debe calcular una secuencia de eventos *sr* que lleve al SMF a un estado válido.

Los objetivos de esta tesis son los siguientes:

- Dar una extensión a la clase Redes de Petri Sistemas de Procesos Simples con Recursos (SPSR) para que sean utilizadas en el problema de tolerancia a fallas con redundancia dinámica.

- Encontrar condiciones necesarias y suficientes, que determinen cuando el problema de tolerancia a fallas tiene solución, basadas en la topología de los modelos en Redes de Petri SPSR del sistema.

1.6 Organización de la memoria de la tesis.

En el capítulo dos presenta cómo utilizar las redes de Petri (RdP) para modelar los SMF. Se selecciona la clase de RdP SPSR porque han demostrado modelar eficazmente el flujo de piezas en una clase de SMF [4]. También presenta las RdP temporizadas porque serán utilizadas para medir el tiempo de las operaciones del SMF.

El capítulo tres introduce los principales conceptos de tolerancia a fallas, los niveles de tolerancia a fallas y las técnicas para implementar un sistema tolerante a fallas.

En el capítulo cuatro se propone una extensión a los SPSR para poder utilizarlos en sistemas tolerante a fallas. La extensión se basa en cuatro funciones que se agregan. Una de ellas determina el tiempo más temprano de finalización de una operación; la segunda, el tiempo más tarde de finalización; la tercera indica como evolucionan las piezas dentro del SMF evoluciona y la cuarta es una señal del controlador que indica que transición de la red de Petri se debe disparar. Además presenta cómo detectar cuando ocurre un error.

En el capítulo cinco se propone un mecanismo para diagnosticar las fallas. Principálmente se proponen dos algorítmos.. El primero para determinar el conjunto de elementos del SMF (efector o máquina) que posiblemente tenga fallas. el cual está basado fuertemente en la topología de la RdP. El segundo algoritmo sirve para localizar los recursos con fallas. el cual está basado en un sistema experto que usa rutinas de prueba para el diagnóstico.

El último capítulo presenta un esquema de recuperación del error en un SMF propuesto. Este esquema cuenta con tres puntos. El primero determina cuándo es posible seguir fabricando la misma la familia de productos. que productos no se pueden seguir fabricando (Teorema 6.1.1). El segundo determina cómo modificar la estructura de la red del SMF cuando un elemento se ha dañado. con el fin de evitar bloqueos. El punto final muestra cómo encontrar una secuencia de recuperación. la cual sirve para llevar al SMF de un estado de error a un estado válido. Finalmente. estos resultados son agrupados en un único algoritmo que determina cómo recuperarse de un error. Un ejemplo sirve para demostrar las principales ideas propuesto.

Por último las conclusiones y trabajos futuro son dados.

1.7 Conclusiones.

En este capítulo se presentó los **Sistemas Tolerantes a Fallas**, algunas definiciones que se utilizan en los **STF**. Se presentó los Sistemas de Manufactura Flexible (SMF), como nacen históricamente y la necesidad de contar con ellos. Se definen los Sistemas Dinámicos de Eventos Discretos, ya que en está tesis los SMF se verán como un SDED. Finalmente se presenta el Problema de Tolerancia a Fallas.

En el próximo capítulo se presentan las redes de Petri y la clase de redes de Petri llamadas Sistemas de Procesos Simples con Recursos que se utilizan para modelar el flujo de piezas de una clase de **SMF**.

Capítulo 2

Modelado de un SMF.

Resumen: En este capítulo se presenta a las redes de Petri (RdP) como una herramienta adecuada para modelar las características discretas de un SMF. Una serie de propiedades estructurales y dinámicas son introducidas. Dichas propiedades serán utilizadas posteriormente para el STF. Finalmente se introducen una subclase de las redes de Petri que son Sistemas de Procesos Simples con Recurso (SPSR), las cuales han demostrado ser adecuadas para modelar el flujo de piezas en un SMF, [4], [8], [15]. En un capítulo posterior se hará una extensión a los SPRS para utilizarlos en los Sistemas Tolerante a Fallas.

2.1 Introducción.

En el proceso de análisis de cualquier sistema es importante obtener un modelo que capture la mayor cantidad de información relacionada con el problema que se desea resolver. Los modelos matemáticos son los que, por su naturaleza y formalidad, permiten hacer una mayor abstracción sobre el problema.

En el capítulo uno se definió un SDED, sistemas en los cuales sus estados tienen valores lógicos o simbólicos asociados, en vez de numéricos, que cambian en respuesta a eventos, los cuales también pueden ser descritos en términos no numéricos..

Las herramientas matemáticas, que se utilizan para modelar SDED, van desde procesos de Markov, autómatas finitos, la teoría de colas hasta las redes de Petri. En esta tesis se utiliza como herramienta matemática para a las redes de Petri para modelar un SMF como un SDED. Las RdP han sido utilizadas para modelar, controlar y analizar un SMF. En 1984, Manuel Silva presentó un trabajo para modelar y controlar parte del sistema de ensamble de la Renault [27]; en [2] muestra como las redes de Petri pueden ser usadas para modelar, controlar y analizar el desempeño de sistemas de manufactura, utilizando para el análisis de desempeño utiliza RdP estócasticas y temporizadas; en [11] se utilizan las redes de Petri temporizadas para el detectar fallas en la etapa del diseño de un controlador de un sistema de manufactura automatizado; en [18] se utiliza las redes de Petri temporizadas para modelar sistemas concurrentes asíncronos y después analizar su desempeño; en [16] se aplica las RdP a la optimización de un SMF, éste se modela con redes de Petri Temporizadas y el criterio de optimización seleccionado es el tiempo de ejecución cíclica de la red; en [8] se presentó un trabajo llamado "Scheduling en un SMF", éste se basa en el estudio de la optimización de SDED, en especial de un SMF, la herramienta que utiliza para modelar los SMF son redes de Petri temporizadas SPSR para evaluar el desempeño del sistema y los criterios de optimización son minimizar los recursos de un SMF; por mencionar algunos investigadores que han trabajado con redes de Petri. En la próxima sección se presentan las RdP y las ventajas que presentan al modelar **SDED.**

2.2 Redes de Petri.

En este apartado se presentan las redes de Petri y sus principales propiedades. Las RdP se utilizan debido a que capturan las principales características de un **SMF**, como son acotamiento, no determinismo, concurrencia, asincronía, relaciones causales, etc. Además las RdP nos proporcionan un punto de vista gráfico que nos permite una comunicación visual agradable y desde el punto de vista matemático existen técnicas de análisis y síntesis que permiten el estudio de propiedades del SMF [5].

Una característica importante de las RdP respecto a su capacidad de análisis es la separación entre la estructura de la red y el marcado. La estructura establece las relaciones entre las entidades que forman el sistema, mientras que el marcado define la dinámica del

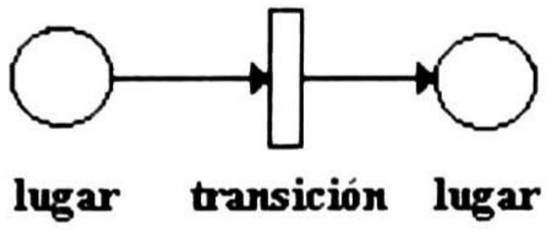


Figura 2.1:

marcas que contienen los lugares. Las marcas se representan gráficamente por puntos en los lugares.

Definición 2.2.3 *Un sistema* $\sum = \langle N, M_o \rangle$ *o red marcada, es una red de Petri N con un marcado* M_o. *El estado de una RdP es representado por el marcado.*

Las funciones de pre-, post-incidencia son representadas por las matrices $PRE=[a_{ij}]$ y $POST=[b_{ij}]$ respectivamente, donde $a_{ij} = Pre(l_i, t_j)$ y $b_{ij} = Post\ (l_i, t_j)$. *PRE* y *POST* también son llamadas C^- y C^+ respectivamente.

Definición 2.2.4 *La matriz de incidencia* $C= [c_{ij}]$, $i = 1,...,n$; $j = 1,...,m$, *está definida por* $c_{ij} = b_{ij} - a_{ij}$.

El conjunto $x^\bullet, x \in L \cup T$ es el conjunto de todos los sucesores inmediatos de x; el conjunto ${}^\bullet x, x \in L \cup T$ es el conjunto de todos los predecesores inmediatos de x.

Definición 2.2.5 *Una transición* $t_i \in T$ *está habilitada para un marcado* M_k *si :*

$$\forall\ l_j \in L : M_k(l_j) \geqslant Pre(l_j, t_i)$$

es decir el número de marcas es mayor a los arcos de entrada a la transición.

Una transición $t_i \in T$ que está habilitada en el marcado M_k puede ser disparada, y su disparo cambia el marcado a M_{k+1}, donde:

$$M_{k+1}(l) = M_k(l) - Pre(l_i, t_j) + Post\ (l_i, t_j), \forall\ l \in L.$$

El hecho de que M_{k+1}sea obtenido a partir de M_k disparando t es representado por $M_k[t > M_{k+1}$.Una secuencia de transiciones $\sigma = t_1 t_2 ... t_n$ es una secuencia de disparos en $\langle N, M_o \rangle$ sólo sí existe una secuencia de marcados tal que $M_o[t_1 > M_1[t_2 > M_2 ... [t_n > M_n$, y se dice que M_n es alcanzable desde M_o. Extendiendo la notación este hecho se representa por $M_o\ [\sigma > M_n$.

El conjunto de los marcados alcanzables desde M_o en la red N se representa por $\mathcal{R}(\langle N, M_o \rangle)$.

El conjunto de todas las secuencias disparables en $\langle N, M_o \rangle$ es:

$$\mathcal{L}(\langle N, M_o \rangle) = \{\sigma \mid M_o\ [\sigma >; M_k, M_k \in \mathcal{R}(\langle N, M_o \rangle)\}$$

Los vectores X que hacen que se cumpla $C.X = 0$ y además $X \geq 0$ son llamados *T-semiflujos*; y los vectores Y tal que $Y^T.C = 0$, $Y \geq 0$ son llamados *P-semiflujos.*

El soporte de un t-semiflujo X se representa por $||X|| = \{t_i \mid X(t_i) \neq 0\}$.

El vector característico del soporte es $|| \vec{X} || = [x_i]$ donde:

$$x_i = \begin{cases} 1 \text{ si } t_i \in ||X|| \\ 0 \text{ otro caso.} \end{cases}$$

El soporte de un p-semiflujo se obtiene de manera similar.

Un p-semiflujo Y_i es de soporte mínimo si y sólo sí no existe otro p-semiflujo Y_i' tal que $||Y_i'|| \subset ||Y_i||$.

Un t-semiflujo X_i es de soporte mínimo si y sólo sí no existe otro t-semiflujo X_i' tal que $||X_i'|| \subset ||X_i||$.Un t-semiflujo (p-semiflujo) es *canónico* sí el máximo común divisor de sus componentes es 1. Un p-semiflujo (t-semiflujo) es *mínimo* si y sólo sí es canónico y de soporte mínimo.

Definición 2.2.6 *La ecuación de estados de RdP se define:*

$$M_{k+1}(l) = M_k(l) + C.\vec{\sigma}, \forall l \in L$$

Donde:

- C es la matriz de incidencia.
- $\vec{\sigma}$ es un vector (m×1) llamado el vector de secuencias de disparo.

Definición 2.2.7 *Un lugar $l \in L$ es k-acotado si $\forall M(l) \in \mathcal{R}(\langle N, M_o \rangle)$ el marcado de l cumple con $M(l) \leq k < \infty$.*

Definición 2.2.8 *Una red marcada $\langle N, M_o \rangle$ es k-acotada sí todos sus lugares son k-acotados. Una red es estructuralmente k-acotada sí $\forall M_o$ todas las redes marcadas $\langle N, M_o \rangle$ son k-acotadas.*

Definición 2.2.9 *Una red marcada $\langle N, M_o \rangle$ es libre de bloqueos sí $\forall M \in \mathcal{R}(\langle N, M_o \rangle) : \exists$ $t \in T$ tal que M habilita t.*

2.3 Subclases de Redes de Petri.

En esta sección se presenta una clasificación de RdP de acuerdo a las características de la topología del grafo.

Definición 2.3.1 ***Grafo Marcado Fuertemente Conexo (GMFC)*** *Es una RdP que cumple con $\mid l^\bullet \mid = \mid {}^\bullet l \mid = 1 \; \forall \; l \in L$ y $\forall \; x, y \in L \cup T \; \exists$ un camino de x a y $(x \rightarrow y)$ y un camino de y a x $(y \rightarrow x)$.*

Definición 2.3.2 ***Máquina de Estados Fuertemente Conexa (MEFC)*** *Es una RdP que cumple con* $| t^{\bullet} | = | {}^{\bullet}t | = 1 \; \forall \; t \in T$ *y* $\forall \; x, y \in L \cup T \; \exists$ *un camino de* x *a* y $(x \rightarrow y)$ *y un camino de* y *a* $x (y \rightarrow x)$.

Definición 2.3.3 ***Una RdP Libre Elección.*** *Es una red es una red tal que:* $\forall \; l \in L$, *si* $| l^{\bullet} | > 1$, *entonces* $\forall t_k \in l^{\bullet}, | {}^{\bullet}t_k | = 1$

Es decir , si dos transiciones t_i y t_j tienen un lugar de entrada l en común, se deduce que l es el único lugar de entrada de t_i y t_j.

2.4 Redes de Petri Temporizadas.

En este apartado estudiaremos una extensión de las RdP que sirven para evaluar el desempeño de un SMF: *Las Redes de Petri Temporizadas.* Las Redes de Petri Temporizadas son muy utilizadas para modelar y analizar SMF; en [6] las utiliza para evaluar el desempeño de un sistema concurrente; en [8] las aplica para el cálculo del Scheduling en un SMF; en [9] las utiliza para crear un simulador de objetos heterogéneos orientado a sistemas de manufactura.

Definición 2.4.1 ***Redes de Petri Temporizadas(RdPT).*** *Es la tripleta* $\langle N, M_o, D \rangle$, *donde* N, M_o *se definen como una RdP ordinaria y* D *es una función que asocia un número real positivo a cada transición* t_i, *representando el tiempo de disparo de la transición* t_i.

En este modelo una marca puede tener dos atributos: *disponible o no disponible.*

Para el marcado inicial todas las marcas están disponibles, y el cambio al estado no disponible se hace de la siguiente manera:

La transición a dispararse está habilitada, es decir:

$$M(l_i) \geq Pre(l_i, t_i) \; \forall l_i \in^{\bullet} t_i$$

donde $M(l_i)$ son marcas disponibles, el disparo de una transición t_i "congela" $Pre(l_i, t_i)$ marcas de cada lugar de entrada l_i (ésto es, no las remueve de los lugares, pero si las etiqueta como no disponibles; de manera que ninguna otra transición las puede utilizar en su disparo), pero no es sino hasta d_i unidades de tiempo más tarde cuando deposita $Post(l_i, t_i)$ marcas disponibles en cada lugar de salida l_j.

2.5 Subclase de RdP que modela el flujo de piezas en un SMF.

La especificación de un SMF se refiere al hecho de describir, sin ambigüedad, las características de un SMF. Por ejemplo, se debe conocer la familia de productos que serán fabricados, las operaciones que requiere cada producto, la cadencia de producción.

Un modelo para un SMF debe capturar esta información.

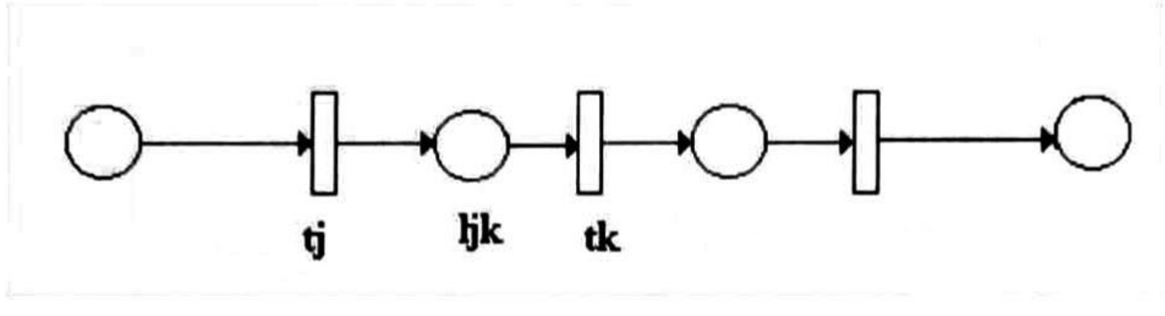


Figura 2.2:

En el presente trabajo se considera que un SMF está especificado de la siguiente forma:

1.- Una secuencia de procesado es una colección de operaciones (es decir $s_{op} = \{op_i op_j op_k ... op_l \mid op_n \in Op\}$) que forma una muestra positiva del lenguaje que describe la fabricación de un producto.

2.- Cada operación tiene asociado uno o más recursos que realizan dicha operación y el costo de realizarla.

3.- El número de productos fabricados por unidad de tiempo (throughput) es conocido.

4.- El costo de producción es conocido.

El modelo que se construya debe reflejar la información anterior. En este caso se utilizarán a las Redes de Petri porque pueden capturar la información anterior y además por ser una herramienta formal que permite realizar análisis.

En esta sección se define una subclase de redes de Petri , siguiendo la misma metodología planteada en [4] Se definen la clase de los *Sistemas de Procesos Simples con Recursos (*SPSR), un SPSR consta de un conjunto de procesos secuenciales que comparten un conjunto de recursos en común. La estructura especificada de éstos sistemas refleja el comportamiento y flexibilidad de un SMF. En la teoría de las redes de Petri, la mayoría de los trabajos se han enfocado al desarrollo de teorías completas para determinadas subclases de redes, tales como grafos marcados, máquinas de estados, redes de libre elección, etc. La clase de red con la que vamos a trabajar surge a raíz de modelar el comportamiento de determinadas clases de SMF y su estudio no está completo todavía.

2.5.1 La clase de los SPSR.

Antes de definir los SPSR definiremos dos subclases para su comprensión, los Procesos Simples y los Procesos Simples con recursos.

Para modelar una secuencia de procesado se utiliza una transición para cada operación en la secuencia. Las transiciones representan operaciones consecutivas y éstas se conectan por medio de un lugar. Es decir si op_j y op_k son dos operaciones consecutivas, entonces se utiliza un lugar $l_{jk} \exists {}^\bullet l_{jk} = t_j, l_{jk}^\bullet = \{t_k\}$.

Cada secuencia de procesado se representa como una red de la figura 2.2.

A continuación se presentará una subclase de redes de Petri que nos permite modelar posibles secuencias para el procesado de una o varias piezas; lo llamaremos *Proceso Simple.*

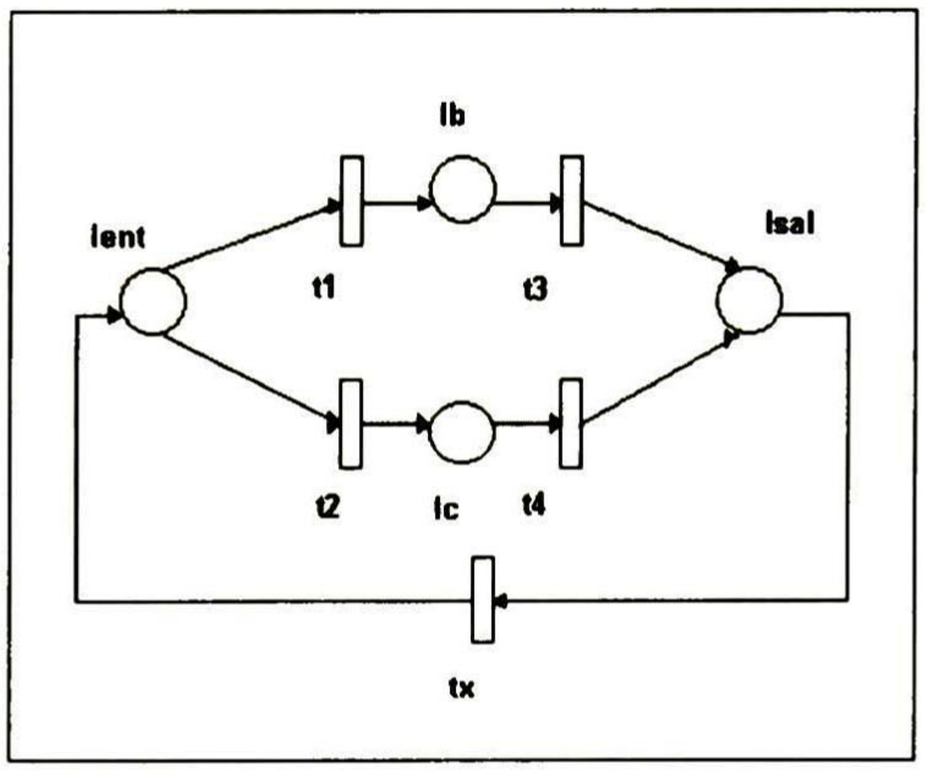


Figura 2.3: Proceso Simple

Definición 2.5.1 ***Proceso Simple(PS).*** *Es una red de Petri ordinaria* $N = (L, T, C)$, *tal que:*

i) $\lambda : Op \rightarrow T$ *es una función que asigna a cada operación del SMF una transición.*
ii) N *es una máquina de estados fuertemente conexa.*

iii) $\exists\ t_x \in T\ \forall X \mid C.X = 0,\ X \geq 0 \rightarrow t_x \in \| X \|$.

iv) $\exists\ l_{ent} \in L \mid l_{ent} \in t_x^{\bullet}$.
v) $\exists\ l_{sal} \in L \mid l_{sal} \in {}^{\bullet} t_x$.

El modelo obtenido es una máquina de estados, ésta nos sirve para representar secuencias de procesado. El modelo tiene la particularidad de permitir la especificación de un producto en una red simple de operaciones que pueden ser ejecutados por diferentes caminos (ésto representa redundancia que existe dentro de un Sistema de Manufactura, no encontrándose ésta en grafos marcados) y además se acerca más a la realidad, porque en un SMF existen varios recursos que pueden realizar la misma operación y por consiguiente existe la posibilidad de obtener un mismo producto por diferentes secuencias de procesado. El significado de la definición es el siguiente, i) λ es una función que asigna a cada operación del SMF una sola transición de la red de Petri, la condición *ii)* modela el conjunto de posibles secuencias de evolución de una pieza en el sistema durante su procesamiento, en el inciso *iii)* nos dice que la pieza está terminada, ésta es una propiedad fundamental en un SMF, es decir cada vez que se dispare la transición t_x habrá un producto terminado, en el inciso *iv)* l_{ent} es el lugar que aloja la materia prima a ser procesada, es decir marca el inicio del proceso, en el inciso *v)* l_{sal} es el lugar que aloja el producto terminado, es decir marca el fin del proceso. El modelo obtenido es mostrado en la figura 2.3.

Definición 2.5.2 *Un marcado inicial admisible para* N *es aquel que* $\forall\ l_i \in L - \{l_{ent}\}$ *se*

cumple que $M(l_i) = 0$ *y* $M(l_{ent}) > 0$.

El modelo anterior no considera cómo son asignados los recursos. Para tomar en cuenta éste hecho vamos a introducir la definición de un PSR como un PS que usa recursos en cada estado. Las interacciones de un PS con el resto de procesos concurrentemente ejecutados en el sistema se lleva a cabo a través del conjunto de recursos, es normal suponer que cuando un proceso se encuentra inactivo no realiza ninguna interacción con los demás.

El recurso R_k se representa con un lugar l_k donde $l_k^\bullet = t_j$ y ${}^\bullet l_k = t_j$ si $Op_j \in \phi(R_k)$ y t_j modela a Op_j.

Esta adición de lugares de recursos genera una nueva clase de redes de Petri que se definen a continuación.

Definición 2.5.3 ***Proceso Simple con Recursos (PSR).*** *Es una red de Petri ordinaria* $N=(L \cup L', T, C \cup C')$ *tal que:*

i) $\psi : Ef \cup M \rightarrow L'$ *es una función que asigna a cada recurso un lugar con las siguientes características:*

ii) La subred generada por $N_x = L \cup T$ *es un PS.*

iii) ${}^\bullet t_x, t_x^\bullet \notin L'$

iv) $\forall\ l_i \in L'$ *se cumple que si:*

$$l_i \in t^\bullet \rightarrow l_i \in {}^\bullet t\ \forall\ t \in T - \{t_x\}.$$

$$M(l_i) = 1.$$

v) $| l_i^\bullet | = | {}^\bullet l_i | \geq 1\ \forall\ l_i \in L'$

la función ψ asigna a cada recurso un solo lugar que $\in L'$ y C' es la matriz de incidencia asociada a dichos lugares. Cuando se escribe $C^+ \cup C^{+'}, C^- \cup C^{-'}$ significa $[\frac{C^+}{C^{+'}}], [\frac{C^-}{C^{-'}}]$, respectivamente. En lo sucesivo denominaremos a L' *conjunto de recursos del sistema.* ver figura2.4

El significado de la definición es el siguiente: como ya hemos explicado, el **PS** modela la secuencia de posibles estados en que el proceso se puede encontrar, además de las acciones del sistema (*transiciones*) que al realizarse provocan un cambio de estado. Los lugares que representan a los recurso en la red de Petri, modelarán junto con su marcado correspondiente la disponibilidad de las máquinas herramientas, los efectores; las condiciones siguientes nos ponen la restricción de que ningún lugar de recurso puede estar "conectado" a la transición t_x, ya que si hubiese alguno significa que la pieza todavía no está terminada lo cual es contrario a lo que se había planteado. En conclusión, todos los lugares $l \in L'$ estarán conectados a las transiciones que representan acciones de transformación sobre las piezas.

Como un SMF tiene la característica de poder producir una familia de productos utilizando recursos compartidos entre los diferentes productos, vamos a definir los Sistemas

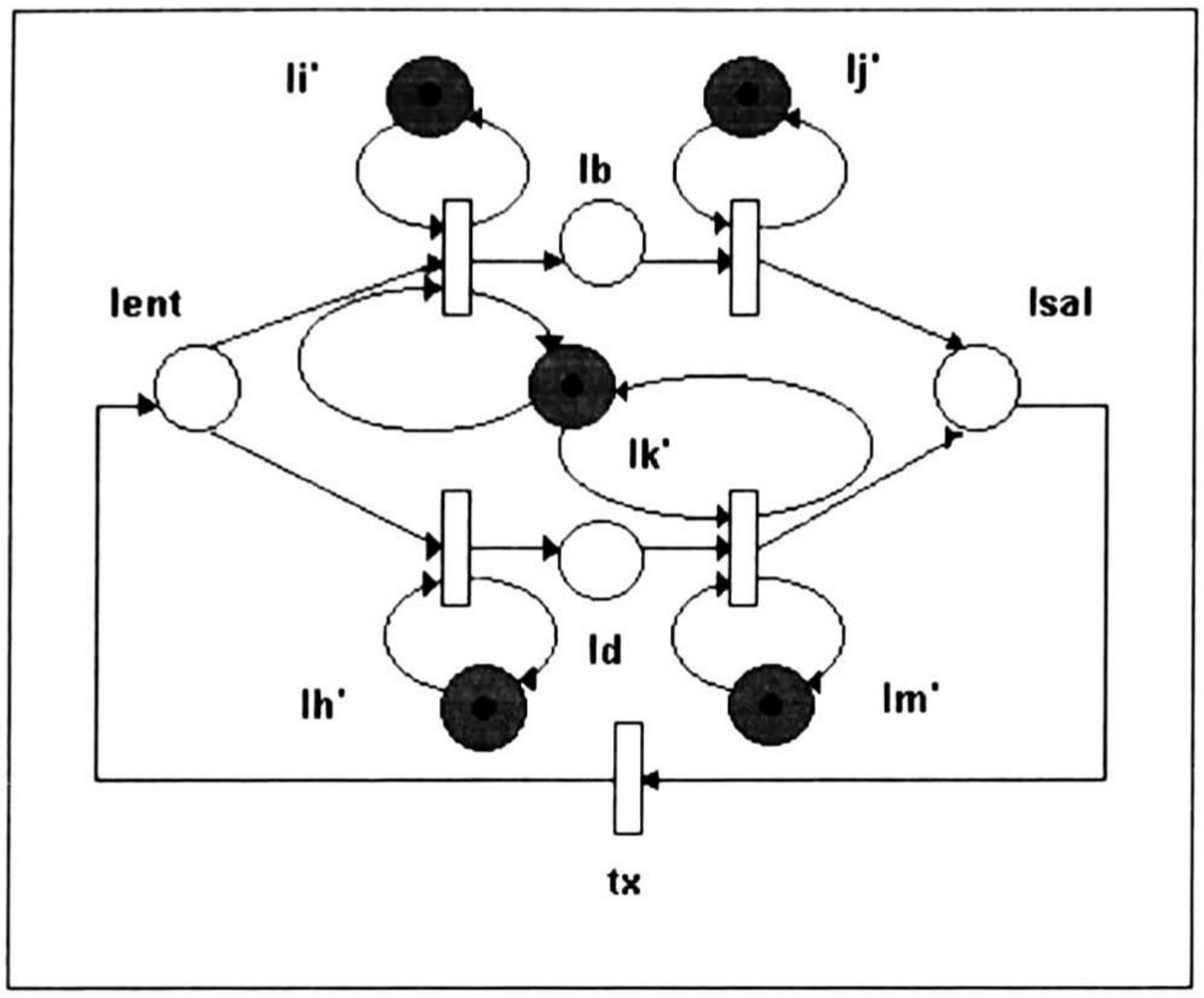


Figura 2.4: Proceso Simple con Recursos.

de Procesos Simples con Recursos que nos sirven para modelar esta característica. Para definir los sistemas de procesos simples con recursos es necesario introducir el concepto de composición de redes vía un conjunto de lugares comunes.

Definición 2.5.4 *Sean* $N_i = (L_i, T_i, C_i), i \in \{1, 2\}$ *dos redes de Petri ordinarias. Entonces:*

a) N_1 *y* N_2 *son compatibles por fusión de lugares si y sólo sí:*

$$T_1 \cap T_2 = \emptyset \text{ y } L_1 \cap L_2 \neq \emptyset$$

b) La red $N = (\ L, T, C\ )$ *tal que:*

$$L = L_1 \cup L_2, T = T_1 \cup T_2, C = C_1 \cup C_2$$

se denomina red compuesta de N_1 *y* N_2 *y se denotará como*

$$N = N_1\ o\ N_2$$

Definición 2.5.5 ***Sistema de Procesos Simples con Recursos (SPSR).****[4]Se define recursivamente como sigue:*

i) Todo PSR es un SPSR.

ii)

$$N_i = (L_i \cup L_i^{'}, T_i, C_i \cup C_i^{'}), i\ \in \{1, 2, 3..., n\}$$

ó

$$N = N_1 \; o \; N_2 \; o \; N_3 \; o ... N_n$$

tal que se cumple:

$$\forall \; i \neq j \; C_i \cap C_j = \emptyset$$

$$L_i \cap L_j \neq \emptyset = L_{ij}$$

Un SPSR es una $N(L, T, C)$ red de Petri que cumple las condiciones anteriores, donde:

$$L = \bigcup_{1}^{i} L_i \cup L_i^{'}$$

$$T = \bigcup_{1}^{i} T_i, \text{ donde } i \in \{1, 2, 3..., n\}$$

C es la matriz de incidencia del SPSR

El significado de la anterior definición es bastante intuitivo: componemos dos **SPSR** únicamente cuando son compatibles por fusión de un conjunto de lugares que modelan recursos del sistema. El modelo completo del SMF es mostrado en la figura 2.5.

Ejemplo de como modelar un SMF con RdP SPSR:

En la figura2.6 se muestra un SMF que cuenta con:

i) tres pallets de almacenamiento (a,b,c).

ii) dos robots (d,g).

iii) dos rectificadoras de superficies planas (e,f; esmeril, pulidor).

iv) dos taladros de banco(h,i).

Las operaciones del SMF son las siguientes:

1.- Si hay piezas en el pallet "a" el robot "d" puede tomar la piezas y colocarlas en cualquier de las rectificadoras "e","f"

2.- Después de realizarse la operaciones en las rectificadoras "e","f" el robot "d"debe tomar la pieza y colocarla en el pallet "b"

3.- Si hay piezas en el pallet "b" el robot "g" puede tomar la piezas y colocarlas en cualquier de los taladros "h","i"

4.-Después de realizarse la operaciones en los taladros "h","i" el robot "g"debe tomar la pieza y colocarla en el pallet "c"

NOTA: Las piezas que se encuentran en el pallet "c" son piezas terminadas.

• Las transiciones de la red representan las siguientes operaciones del SMF:

t1:Toma el robot "d" una pieza del pallet "a" y la coloca en la rectificadora "e"

t2: Toma el robot "d" una pieza del pallet "a" y la coloca en la rectificadora "f"

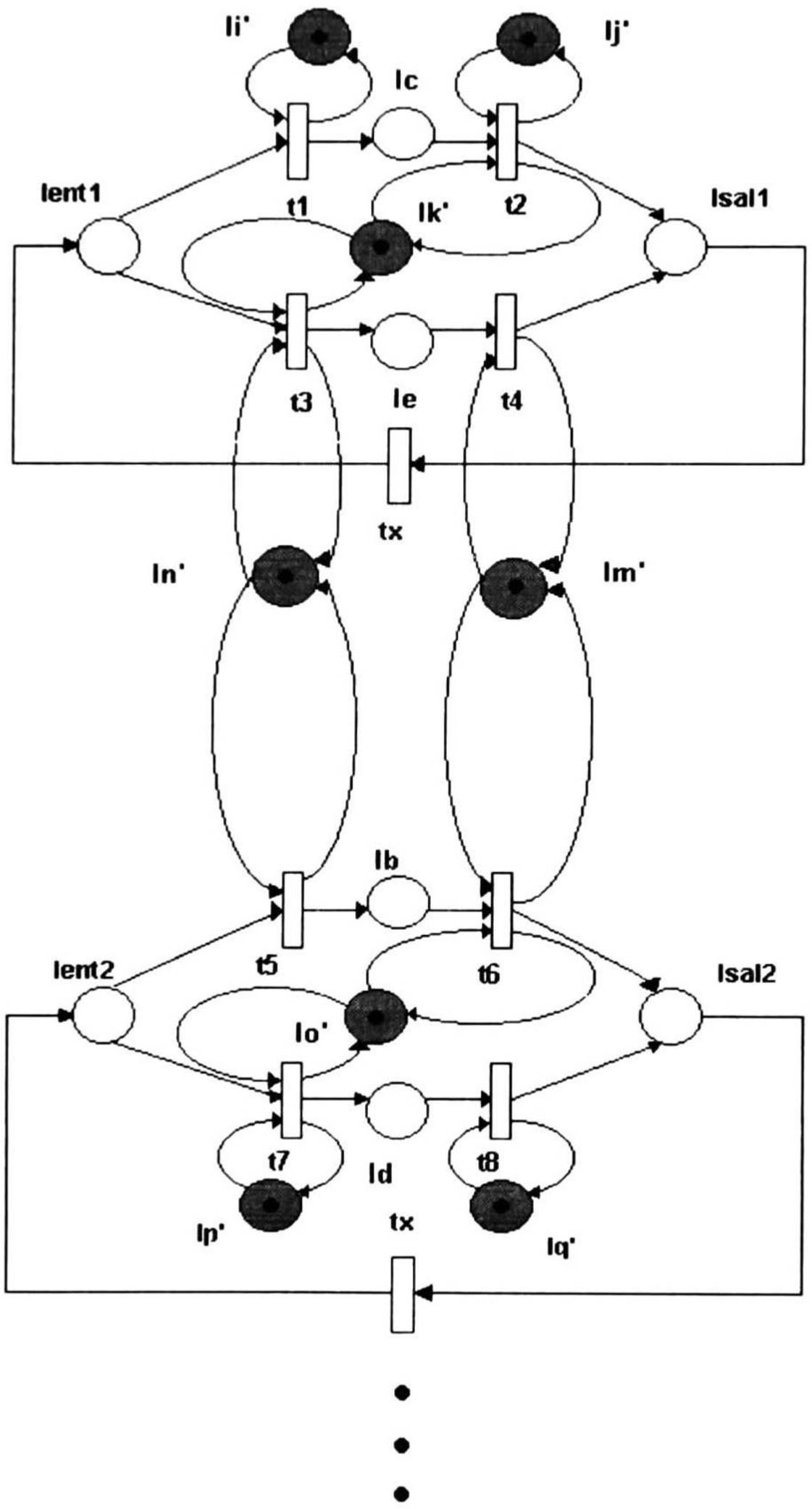


Figura 2.5: Sistema de Procesos Simples con Recursos.

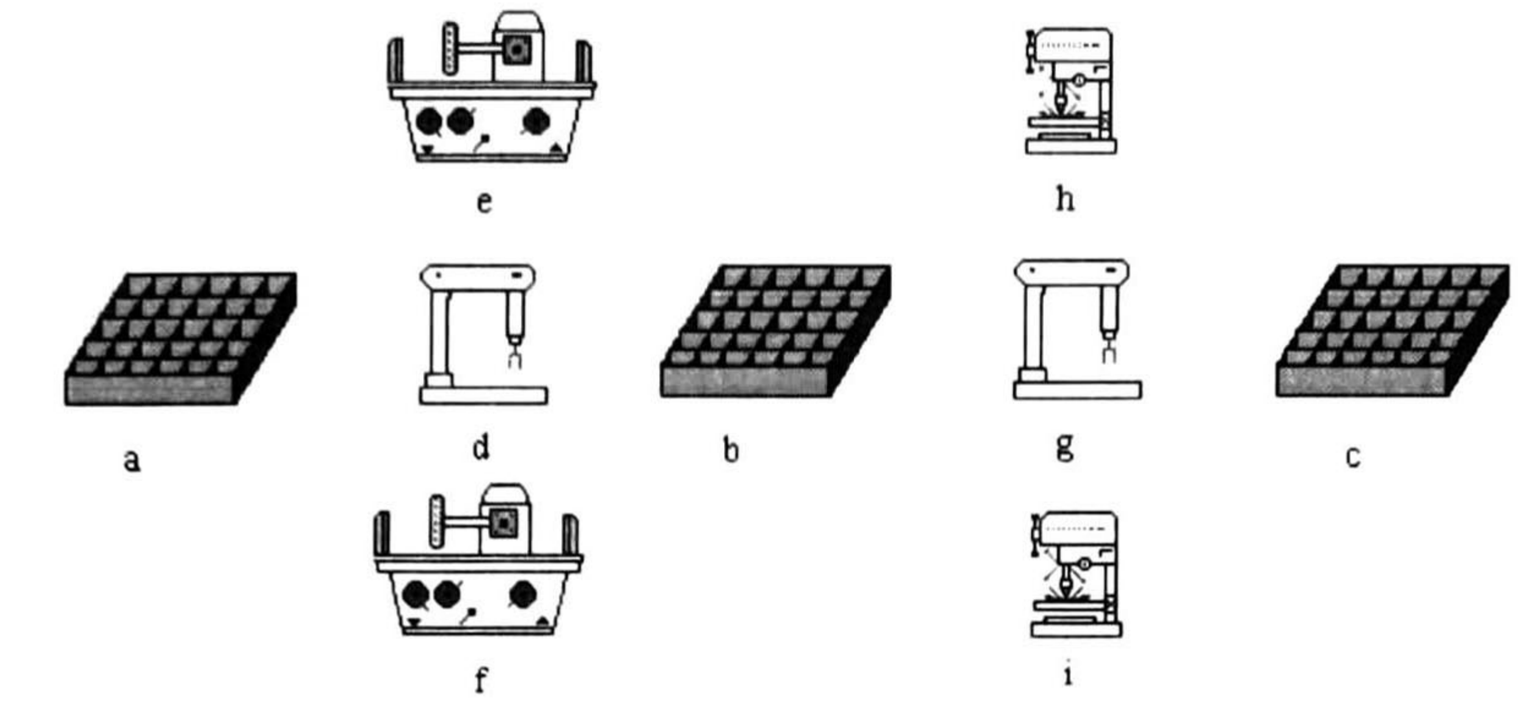


Figura 2.6: Sistema de Manufactura Flexible.

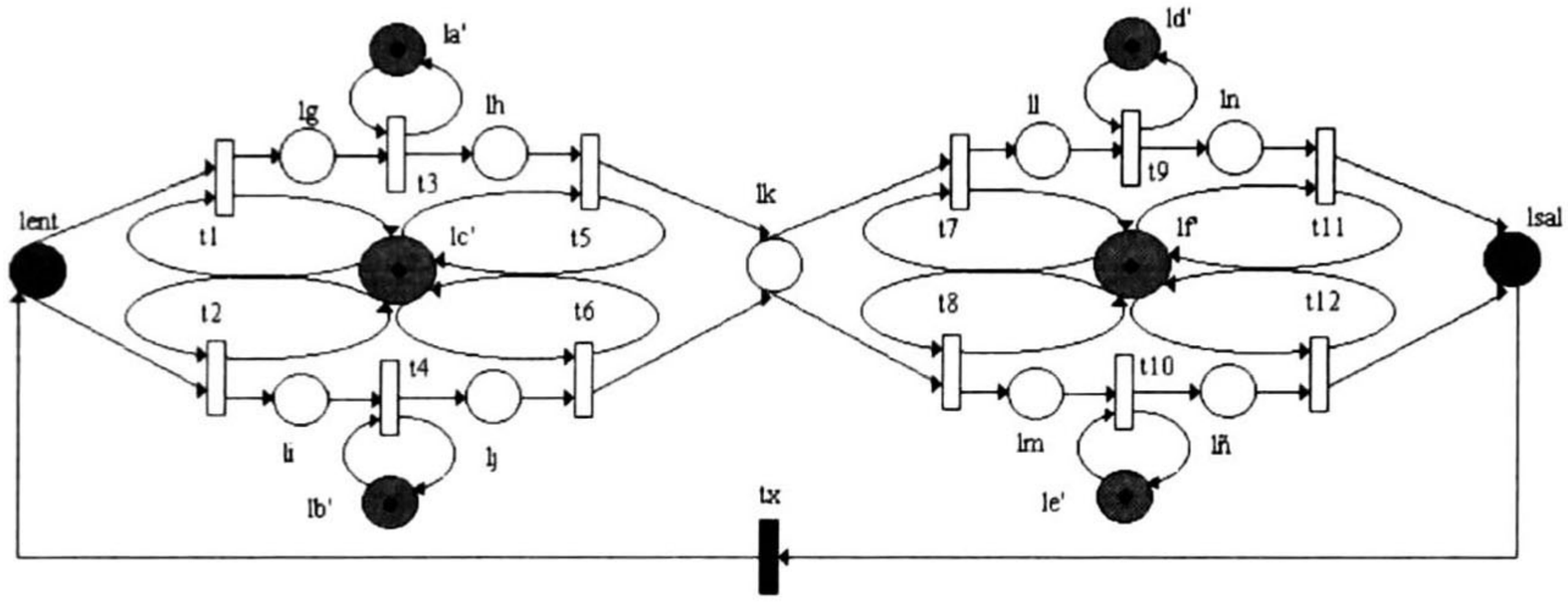


Figura 2.7: Red de Petri (SPSR) del SMF

t3: La rectificadora "e"realiza su función sobre la pieza.
t4: La rectificadora "f" realiza su función sobre la pieza.
t5: Toma el robot "d" la pieza de la rectificadora "e" y la coloca en el pallet "b".
t6: Toma el robot "d" la pieza de la rectificadora "f" y la coloca en el pallet "b".
t7: Toma el robot "g" una pieza del pallet "b" y la coloca en el taladro "h".
t8: Toma el robot "g" una pieza del pallet "b" y la coloca en el taladro "i".
t9: El taladro "h" realiza su función sobre la pieza.
t10: El taladro "i" realiza su función sobre la pieza.
t11: Toma el robot "g" la pieza del taladro "h" y la coloca en el pallet "c".
t12: Toma el robot "d" la pieza del taladro "i" y la coloca en el pallet "c"
• Los recursos del SMF son representados por los siguientes lugares y su marcado:
la': representa la rectificadora de superficies planas "e".
lb': representa la rectificadora de superficies planas "f"
lc':representa el robot "d"

ld′: representa el taladro de banco "h"
le′: representa el taladro de banco "i"
lf′: representa el robot "g"
NOTA: Las piezas que se encuentran en el pallet "c" son piezas terminadas.

En la figura 2.7 se muestra un modelo SPSR del SMF descrito.

2.6 Conclusiones

Se presentó a las Redes de Petri que se utilizan para modelar SMF, se utiliza la subclase de los Sistemas de Procesos Simples con Recursos (SPSR), para modelar las piezas, la secuencia de procesado para la fabricación de un producto, las operaciones que se realizan sobre las piezas, las máquinas o efectores que intervienen en cada operación, si la máquina o efector esta lista para realizar una operación y la familia de productos de un SMF. Los SPSR no modelan a la máquina o efector y no se pueden modelar operaciones de ensamblado o desensamblado.

En el siguiente capítulo se presentan los Sistemas Tolerantes a Fallas, cómo se clasifican y qué elementos lo componen.

Capítulo 3

Tolerancia a Fallas.

Resumen: En este capítulo se presentan algunas técnicas que nos ayudan a mejorar la confiabilidad de un sistema como son la prevención de fallas y la tolerancia a fallas. Se clasifican los tipos de fallas debido a su origen. Además se presentan los diferentes niveles de tolerancia a fallas que existen. Dentro de las técnicas de tolerancia a fallas se verá que dependen de elementos extras que se introducen al sistema es decir componentes redundantes. Se definirán las dos aproximaciones de redundancia, la redundancia estática, la redundancia dinámica y sus principales características.

3.1 Introducción.

Un SMF puede caer en diferentes estados de error cuando se lleva a cabo una operación ocasionado por fallas en algún efector o máquina. Del ejemplo de la última sección del capítulo anterior, se pueden obtener muchos estados de error en el que puede caer el SMF. Por ejemplo, en el pallet "a" se encuentra una pieza lista para ser procesada, la operación siguiente a realizar es que el robot "d" tome la pieza del pallet "a" y la coloque en la rectificadora "e" La operación se realizó, la pieza está lista para seguir su proceso, pero existe una falla en el robot "d" Puede que el origen de la falla sea el sensor del robot, es decir que el sensor este dañado, indicando que el robot "d" no está disponible.

Otro ejemplo de un estado de error, es que el robot "d" tome la pieza del pallet "a" , pero no la coloque en la rectificadora "e" La causa del error puede ser que en el trayecto a la rectificadora hubo una falla en el actuador final del robot "d" y se le cayó la pieza.

Lo ideal de cualquier sistema, sería que no ocurrierán fallas, pero como se verá en las siguientes secciones ésto es imposible. Otro punto importante es que si las fallas se presentan en algunos recursos, no se tenga que parar todo el sistema.

Las fallas debido a su origen se pueden clasificar en las siguientes [13]:

- Fallas transitorias: una falla transitoria comienza en un tiempo particular, permanece en el sistema por algún periodo de tiempo y luego desaparece. Por ejemplo, los componentes de hardware los cuales tienen reacciones adversas a algunas interferencias externas tales como campos eléctricos o radiactividad. Después que la perturbación desaparece lo hace la falla. Muchas fallas en los sistemas de comunicación son transitorias.

- Fallas permanentes: una falla permanente aparece en un tiempo particular y se queda en el sistema hasta que éste se repara. Por ejemplo, un alambre roto o determinados errores en el diseño del software.

- Fallas intermitentes: éstas son fallas transitorias que ocurren de vez en cuando. Por ejemplo, un componente del hardware que es sensitivo al calor, trabaja por un tiempo, se enfría y vuelve a trabajar.

Dos conceptos que pueden ayudar a los diseñadores a mejorar la confiabilidad de sus sistemas son distinguidos [13]:

- Prevención de Fallas, intenta eliminar cualquier posibilidad de falla que ocurra dentro de un sistema antes de que este operando.

- Tolerancia a Fallas, le permite a un sistema que esté operando aun en la presencia de fallas.

3.2 Prevención de Fallas.

Hay dos etapas para la Prevención de Fallas: Evitación de Fallas y Remoción de Fallas[13]:

La Evitación de Fallas pretende limitar la introducción de componentes potenciales de falla durante la construcción del sistema. Los componentes de un sistema son de hardware o de software

Para mejorar la confiabilidad de hardware del sistema se debe:

a) usar componentes más confiables dentro del costo dado y las restricciones de desempeño.

b) usar técnicas refinadas para interconexión de componentes y ensambles de subsistemas.

c) usar técnicas de blindaje del hardware para eliminar formas esperadas de interferencias.

Por otro lado, los componentes de software no se deterioran con el uso, por lo que los errores que se introducen se deben a mala programación, para mejorar la calidad del software se debe:

a) especificar formalmente los requerimientos.

b) usar metodologías de diseño probadas.

c) usar lenguajes con abstracción y modularidad.

d) usar un ambiente para desarrollo de proyectos, para ayudar a manipular los componentes de software y por lo tanto el manejo de la complejidad.

A pesar de las técnicas de Evitación de Fallas, se ha comprobado que las fallas inevitablemente estarán presentes en el sistema después de su construcción.

La segunda etapa de Prevención de Fallas es: *Remoción de Fallas.* Esto normalmente consiste en procedimientos de búsqueda y eliminación de las causas del error. A pesar de que técnicas como revisión del diseño, verificación del programa e inspección del código pueden ser empleadas, generalmente se pone énfasis en la prueba del sistema a través de simulación. Desafortunadamente, las pruebas de los sistemas nunca pueden ser exhaustivas y quitar todas las fallas potenciales. En particular existen los siguientes problemas:

1.- Una prueba puede usarse para demostrar la presencia de fallas, pero no la ausencia de éstas.

2.- Algunas veces es difícil probar bajo condiciones reales. Por ejemplo, una de las mayores causas que conciernen a la Iniciativa de Defensa Estratégica Americana es la imposibilidad de pruebas reales, excepto en condiciones de batalla. La mayoría de las pruebas se hacen con sistemas en modo de simulación y es difícil garantizar que la simulación sea confiable.

3.- Los errores que han sido introducidos en la etapa de requerimientos de desarrollo del sistema puede no manifestarse hasta que el sistema esté operando. Por ejemplo, en el diseño de la Aeronave F-18 se introdujo un error concerniente al tiempo de liberación de un misil montado en el ala. El problema fué descubierto cuando el misil falló al separarse del lanzador

después de la activación, causando que la aeronave perdiera violentamente el control. Un ejemplo de la Evitación de Fallas es la nave espacial viajero no tripulada Voyager.

A pesar de todas las técnicas de verificación y pruebas, los componentes en general de hardware fallarán, por lo tanto la filosofía de Tolerancia a Fallas debe usarse.

3.3 Tolerancia a Fallas.

Debido a las inevitables limitaciones del concepto de Prevención de Fallas, los diseñadores de los STR deben considerar el uso de la Tolerancia a Fallas. Por supuesto esto no significa que debemos abandonar las técnicas para la Prevención de Fallas.

Varios niveles de Tolerancia a Fallas pueden ser previstos en un sistema [13]: Tolerancia a Fallas Total, Tolerancia a Fallas de Degradación Suave y Tolerancia a Fallas de Estado Seguro. A continuación se presentan sus definiciones y se mencionan algunos ejemplos.

Definición 3.3.1 *Tolerancia a fallas Total. El sistema continua operando, en presencia de fallas aunque sea por un período de tiempo limitado, sin pérdidas significativas de funcionalidad y desempeño.*

Por ejemplo, en un SMF, ocurren fallas transitorias o intermitentes en algunos de los recursos, las fallas desaparecen en un periodo pequeño de tiempo y la estructura de nuestro SMF se conserva.

Definición 3.3.2 *Tolerancia a Fallas de Degradación Suave. El sistema continua operando en presencia de fallas, aceptando una degradación parcial de la funcionalidad o desempeño.*

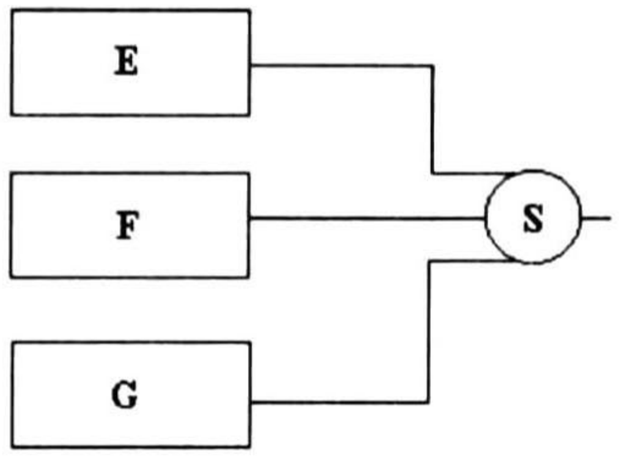
Por ejemplo, en un SMF, las fallas que se presentan, son fallas permanentes en algún recurso, el SMF va a sufrir una degradación, porque las operaciones en las que intervenía ese recurso dentro del SMF no se van a poder realizar, el SMF puede seguir produciendo, generalmente disminuyendo su cadencia de producción, hasta que el recurso este reparado.

Definición 3.3.3 *Tolerancia a Fallas de Estado Seguro. El sistema mantiene su integridad mientras acepta paros temporales en su operación.*

Por ejemplo, ahora se presentan fallas permanentes en algunos recursos, al eliminar las operaciones en las que intervienen éstos recursos, el SMF no puede seguir produciendo, se tiene que llevar al SMF a un estado seguro, para pasar a la etapa de reparación de los recursos; otro ejemplo, el control por computadoras de los alerones del Airbus A310 cuando se detecta un error en el aterrizaje se lleva el sistema a un estado seguro. En esta situación un estado seguro es tener ambas alas en la misma posición.

Todas las técnicas de Tolerancia a Fallas dependen de un elemento extra introducido en el sistema para detectar el error, diagnosticar la falla y la recuperación del error.

Estos componentes son redundantes en el sentido de que no son requeridos para la operación normal del sistema, a esto se le llama normalmente redundancia protectiva. El objetivo



E	F	G	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figura 3.1:

de la Tolerancia a Fallas es la de disminuir la redundancia mientras se maximiza la confiabilidad, sujeta a las restricciones de costo y tamaño del sistema. Se debe tener cuidado en la estructuración de los STF porque los componentes agregados inevitablemente incrementan la complejidad de todo el sistema. Esto conduciría a sistemas menos confiables. Ejemplo, el primer lanzamiento del transbordador espacial se abortó por una dificultad de sincronización con la réplica de los sistemas de computo en 1981.

Para ayudar a reducir los problemas asociados con la interacción con los componentes redundantes es importante separar los componentes tolerantes a fallas del resto del sistema. Hay varias clasificaciones de redundancia dependiendo de cuales componentes del sistema están bajo consideración y cual terminología se está usando [13].

3.3.1 Tolerancia a Fallas con redundancia estática.

En la Tolerancia a Fallas con redundancia estática es también conocida como libre de fallas, los componentes extras son empleados dentro del sistema, para esconder los efectos de las fallas. El caso más usado es la Redundancia Modular Triple (TMR) que consiste en tener tres réplicas del sistema y un circuito llamado votante; el circuito compara la salida de todas las réplicas y toma como respuesta correcta la respuesta de la mayoría. Si una salida cualquiera difiere de las otras, esta salida se enmascara es decir se anula y se toma como correcta las salidas de las otras dos, evitando así que la falla se manifieste como un error en el sistema. La hipótesis aquí es que la falla no es debida a defectos en el diseño, sino que es debida a transitorios o degradación de los componentes. En la figura 3.1 se presenta un ejemplo de redundancia triple modular donde las replicas E, F y G son idénticas con salidas binarias y S es la señal correcta que enmascara el error (circuito votante).

Algunos problemas que se presentan son los siguientes:

¿Qué pasa cuando la salida es analógica?

Como cada réplica del sistema se hace con tecnologías diferentes, las señales de salida de las tres réplicas no serán idénticas, es decir tendrán pequeñas variaciones entre ellas. Por lo tanto el circuito votante debe ser capaz de distinguir entre una variación debido a la

tecnología usada y una variación ocasionada por una falla, para tomar como correcta o no la señal en la etapa de votación.

¿Qué pasa cuando los tiempos de respuesta son diferentes?

El circuito votante debe saber en que tiempo se presenta cada señal de las réplicas, para no tomar una señal errónea, y después pasar a la etapa de votación.

Las réplicas siempre están trabajando haciendo más pesado y más lento el sistema por mencionar algunos puntos.

3.3.2 Tolerancia a Fallas con redundancia dinámica.

La redundancia dinámica tiene un componente que indica si existe un error en el sistema. (es decir se detecta el error en lugar de la Tolerancia a Fallas con redundancia estática que lo enmascara). El error es tratado para determinar qué elemento está fallando. A esta etapa se le conoce como diagnóstico. El estado de error del sistema se debe llevar a un estado válido del sistema a esta etapa se le conoce como recuperación del error. La recuperación del error se lleva a cabo por un conjunto de rutinas. Para implementar la Tolerancia a Fallas con redundancia dinámica se tiene esquemas basados en puntos de prueba y puntos de recuperación; recuperación hacia adelante o hacia atrás. En los siguientes capítulos serán tratados estos conceptos.

3.4 Arquitectura de la aproximación propuesta del STF.

En los sistemas continuos, es decir los sistemas que se pueden modelar mediante ecuaciones diferenciales, diferentes aproximaciones para la detección de fallas (detección del error) usando su modelo matemático han sido desarrolladas en los últimos años. Por ejemplo [25], [26], [20].

La detección del error o de fallas basado en un modelo consiste en obtener si existen discrepancias no aceptables entre las señales de entrada al sistema, las señales de salida con las señales generadas por el modelo.

En esta tesis se propone modelar el SMF en redes de Petri SPSR y utilizar el modelo en las etapas de diagnóstico de fallas y recuperación del error.

El sistema tolerante a fallas está compuesto de los siguientes sistemas:

- **Modelo**. Es un modelo del SMF en RdP SPSR visto como un SDED.
- **Detección del error.** Es el primer problema de tolerancia a fallas que se planteó en el capítulo uno, conocer cuándo el sistema alcanza un estado r_e que no sea el correcto, se obtendrá restando los estados del modelo de los estados del SMF, si la diferencia no es igual a cero el SMF se encuentrá en un estado de error.
- **Diagnóstico de fallas.** Es el segundo problema de tolerancia a fallas, encontrar que recursos del SMF están fallando, se llevará acabo por medio de pruebas a los recursos, a estas pruebas las llamaremos pruebas de diagnóstico.

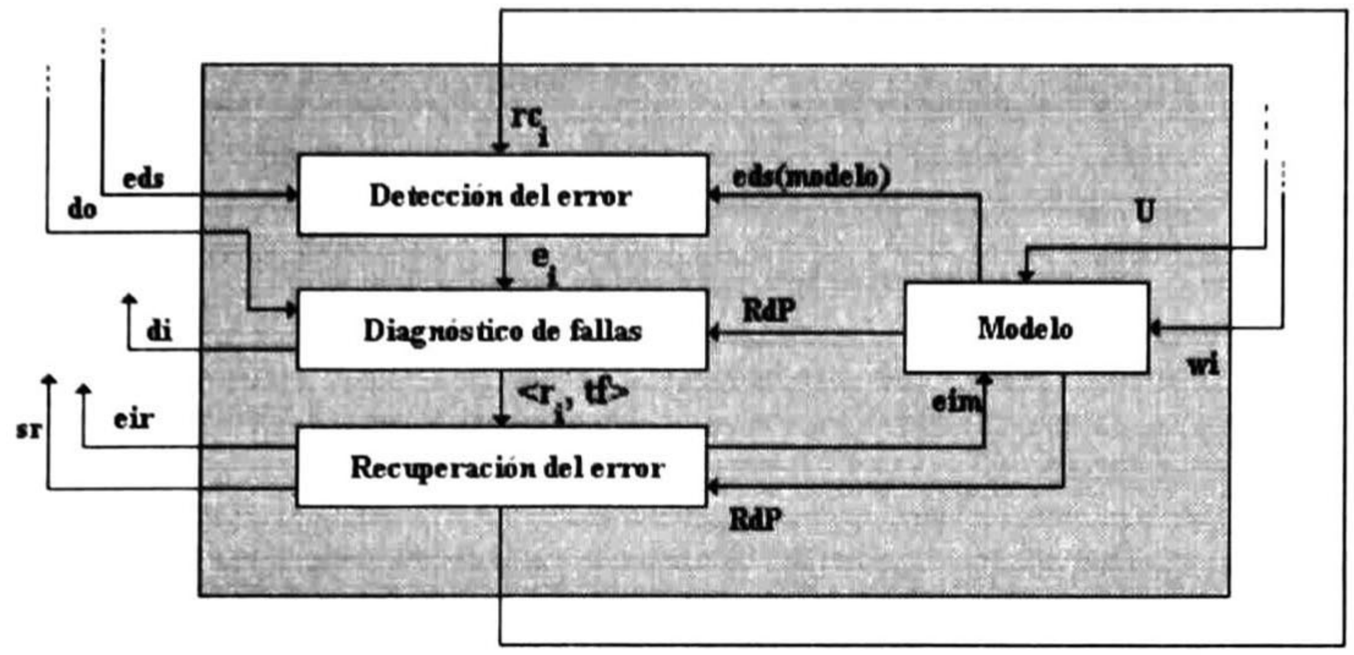


Figura 3.2: Sistema Tolerante a Fallas.

- **Recuperación del error.** Es el tercer problema de tolerancia a fallas, es encontrar si existe una secuencia de operaciones que lleve del estado de error a un estado válido. Para ello se utilizará la ecuación dinámica de la RdP.

El diagrama a bloques del sistema tolerante a fallas, que se propone en esta tesis es el mostrado en la figura 3.2.

La detección del error se lleva acabo calculando e_i y consiste en restar a los estados del sistema *eds* los estados del modelo $eds(RdP)$. Si e_i es diferente de cero es porque ocurrió una falla, y se debe activarse el diagnóstico de fallas.

En la etapa de diagnóstico de fallas dependiendo del vector e_i, y la estructura de la red de Petri RdP se reduce el número de recursos a analizar. Se aplican pruebas a los recursos *di* y dependiendo de los resultados *do*, se determina que recurso r_i está fallando y tipo de falla tf.

La etapa de recuperación del error depende del recurso y de su tipo de falla. La estructura de la red de Petri RdP ayuda a encontrar la secuencia de recuperación sr, y determina que operaciones del sistema *eir* (en el modelo *eim*) en caso de fallas permanentes nos pueden llevar a un estado de bloqueo. Para saber cuando se termina la etapa de recuperación se activa la señal rc_i. Las señales restantes fueron definidas en el primer capítulo.

3.5 Conclusiones.

En este capítulo se presentaron los conceptos de prevención de fallas y tolerancia a fallas, los dos tipos de tolerancia a fallas que existen, y la arquitectura del control tolerante a fallas que se utiliza en esta tesis.

Los conceptos de Tolerancia a Fallas con redundancia dinámica serán expuestos en los próximos capítulos desde la perspectivas de Redes de Petri. Condiciones necesarias y suficientes para la recuperación del error serán presentadas. En el próximo capítulo se presenta

la primer etapa del control tolerante a fallas, que es la detección del error, y se define lo que es el Modelo y cómo se utilizará en el **STF**.

Capítulo 4

Aproximación propuesta: Detección del error.

Resumen: En este capítulo se presenta la primer etapa problema de los **Sistemas Tolerante a Fallas (STF)**, que es la etapa de **detección del error**. Así mismo, se propone el modelo que se utilizará en el esquema de la detección del error, y los distintos elementos que lo componen.

4.1 Introducción.

La efectividad de un STF depende de la efectividad de las técnicas de la detección de sus errores.

Se pueden identificar dos clases de técnicas de detección de error [13]:

- Detección ambiental. Éstos son los errores que son detectados en el ambiente donde se ejecuta el programa. Incluyen aquellos que son detectados por el hardware tales como "instrucción ilegal ejecutada", "sobreflujo aritmético", etc.
- Detección de la aplicación. Éstos son los errores que son detectados en la aplicación misma. Las diferentes técnicas que pueden usarse por la aplicación son las siguientes:
  - Verificación de las réplicas. Se ha mostrado que la N-programación puede ser utilizada para tolerar fallas y que la técnica puede ser utilizada para detectar el error empleando la redundancia "dos modular" [25], [26], [20].
  - Verificación de tiempos. Éste consiste de temporizadores, es decir, si un proceso no inicia dentro de una ventana de tiempo predefinida, se asume que el proceso está en un error.
  - Verificación regresiva. Ésta se utiliza en componentes donde existe una función (isomórfica) uno a uno entre la salida y la entrada. A partir de la salida se calcula la que debería ser la entrada y se compara el valor con la entrada actual.
  - Verificación de códigos. Son usados para comprobar que los datos sean los correctos. Se basan en la información redundante contenida dentro del dato.
  - Verificación razonable. Son basados en el conocimiento del diseño interno y la construcción del sistema. Ellos verifican que el estado de los datos o valores de un objeto sean razonables, basándose en su uso.
  - Verificación estructural. Se utiliza para verificar la integridad de los datos de los objetos tales como una lista y una cola. Pueden consistir de conteo de los números de elementos en los objetos, apuntadores redundantes, o información del estado extra.

En este trabajo se utilizarán los verificadores de tiempo y un modelo del SMF, para la detección del error.

4.2 Detección del error.

Recordando del capítulo uno, en el primer punto de la definición del problema de tolerancia a fallas se establece que se debe conocer cuándo se ha producido un error. A éste hecho se conoce como **Detección del Error.**

El SMF alcanza un estado r_e de error si :

- Si alcanzó un estado $r_e \notin S(SMF)$.
- Si alcanzó un estado $r_e \in S(SMF)$ en un tiempo no especificado.

Para resolver el problema de Detección del error se propone un esquema basado en el modelo. Para esto el modelo debe ser capaz de reproducir los estados $S(SMF)$ en el tiempo especificado, y suponemos que el modelo no falla.

En el capítulo dos se vio que un SMF puede ser modelado con un SPSR. Además los SPSR nos reproducen el espacio de estados $S(SMF)$, pero para poderlos utilizar en la **detección del error,** se les agregarán cuatro funciones. Las primeras dos están relacionadas con el tiempo que tarda una operación, la tercera son señales provenientes del controlador y la cuarta son señales de los sensores provenientes del SMF.

4.3 Modelo.

Para utilizar los SPSR en la detección del error, se debe considerar cómo las señales provenientes del SMF y del control óptimo afectan al modelo propuesto.

A continuación se presenta cómo incorporar las señales de entrada al modelo y los verificadores de tiempo a los SPSR.

Sea $o = [o_1, o_{2..} \ o_k]$ un vector asociado a las diferentes señales booleanas o eventos provenientes del sistema; entonces existen 2^k combinaciones de estas señales y a cada combinación la llamaremos $mo_i, i \in [1, 2^k]$ (los diferentes monomios de entrada mo);

Con la consideración previa definimos la siguiente función:

$$W = [w_i];\ i \in [1...m]$$

donde

$$w_i = \sum_{h=1}^{2^k} \varphi_h . mo_h$$

W es una función vectorial definida sobre los mo y que asocia un 1 o un 0 a una transición.

En estos casos el símbolo $\sum$ significa la OR de los elementos, y $\varphi_h \in [0,1]$, donde 0 indica el mo_i que no debe ser tomado en cuenta; 1 indica que debe ser tomado en cuenta.

El modelo es un Sistema de Procesos Simples con Recursos para Tolerancia a Fallas y se define como:

Definición 4.3.1 ***Sistema de Procesos Simples con Recursos para Tolerancia a Fallas (SPSRTF).*** *Es un* ***SPSR*** *que además contiene las siguientes funciones:*

i) $\phi_1 : T \rightarrow R^+$ *es una función que asocia el tiempo mínimo en* $t \in T$.

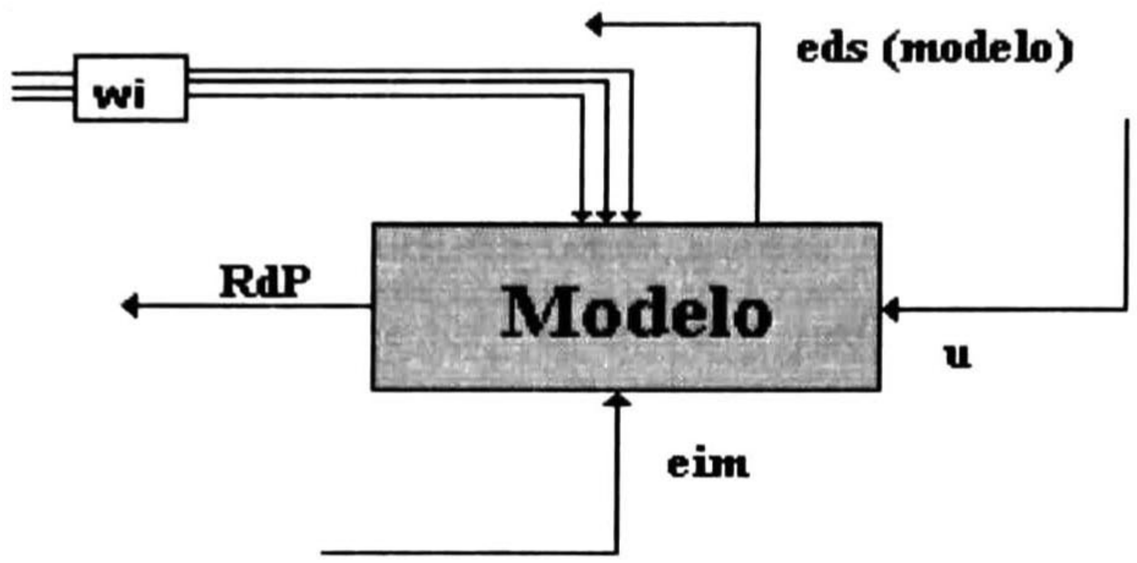


Figura 4.1: Diagrama a bloques del Modelo.

ii) $\phi_2 : T \rightarrow R^+$ *es una función que asocia el tiempo máximo en*

$t \in T$.

iii)$U = [u_i(t)]^i$ *donde* $u_i(t) \in [0,1]$ $i = 1, 2, 3, ..., n$ n *es el numero de transiciones. Es una señal proveniente del control óptimo que indican que transiciones se deben disparar.*

U son las señales generadas por el control óptimo que indican que eventos deben realizarse.

iv) $W : \{mo_i\} \rightarrow T$ *es una función de entrada a las transiciones.*

La función *i)* es el tiempo mínimo en que debe realizarse cada operación (transición) dentro del sistema; *ii)* es el tiempo máximo en que debe realizarse cada operación (transición) dentro del sistema; *iii)* son señales del controlador que indican que trasición t_i se debe habilitar; *iv)*son señales de los sensores que nos indica que la pieza ya fué procesada.

El diagrama a bloques del Modelo, las señales de entrada y las señales de salida son las mostradas en la figura 4.2. La señal *eim* indica que transiciones se deben de eliminar de la RdP.

4.3.1 Reglas de disparo y ecuación de estado.

Sea τ_j el tiempo en que $u_j(\ \tau_j\ )$ cambia de cero a uno.

Sea ξ_j el tiempo en que ocurre el cambio de $M_k(l_i) < E(l_i, t_j)$ a $M_k(l_i) \geq E(l_i, t_j)$.

Entonces la transición t_j está habilitada al tiempo Γ_j si y solo sí $\tau_j \ \wedge \xi_j \neq 0$.

Una transición t_j debe iniciar su disparo tan pronto como está habilitada $\Gamma_j = \max(\xi_j, \tau_j\ )$.

El disparo de t_j provoca los siguientes cambios.

1.-$\xi_j = 0, \tau_j\ = 0$.

2.- $M_k(l) = M_{k-1}(l) - C^-.\ \vec{v_j}$ al tiempo Γ_j

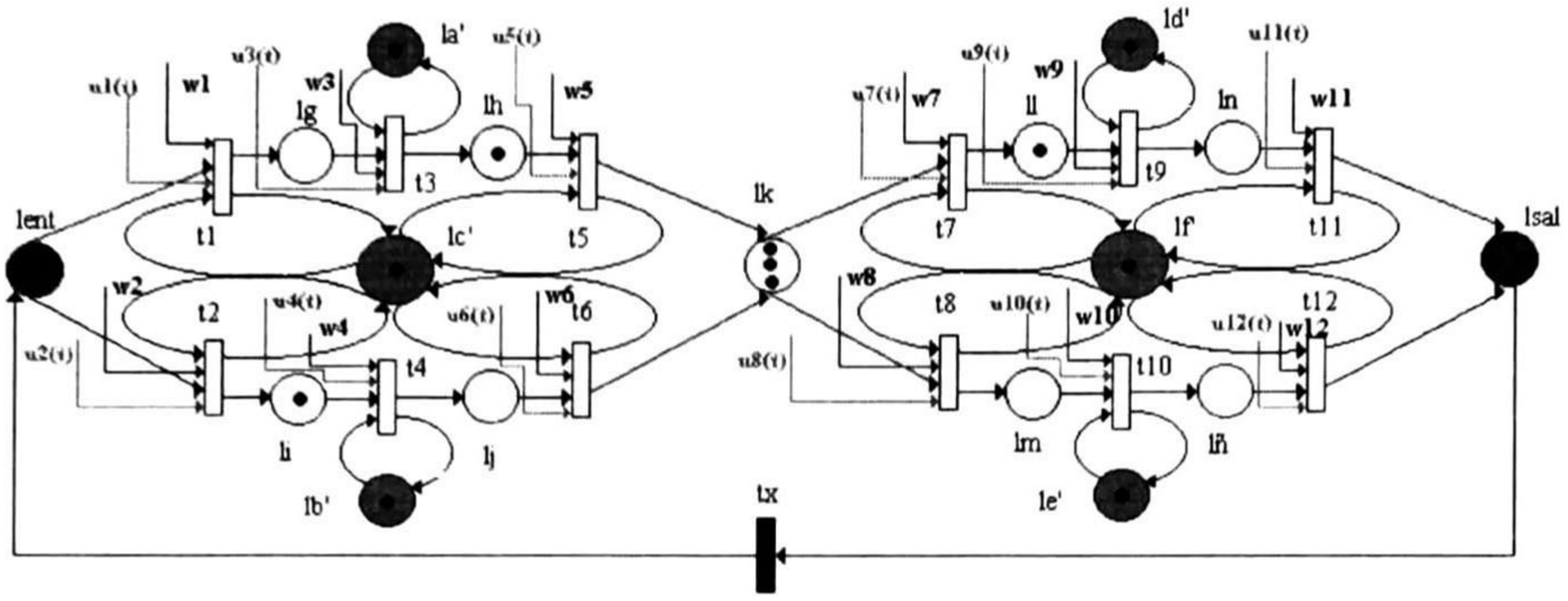


Figura 4.2:

La transición termina su disparo y el marcado evoluciona al tiempo Γ_j^n, dependiendo de las funciones de tiempo $\phi_1(t_j), \phi_2(t_j)$, y del tiempo en que se active la señal w_i. Se tienen los siguientes casos:

$$3.\text{-}M_{k+1}(l) = \begin{cases} M_k(l) + C^+ \cdot \vec{v_j}, \text{ si } ((\Gamma_j + \phi_1(t_j) \leq \Gamma_j^n \leq \Gamma_j + \phi_2(t_j)) \wedge (w_j = 1)) \\ \vee\ (\Gamma_j^n > \Gamma_j + \phi_2(t_j)) \\ M_k(l) + C^- \cdot \vec{v_j}, \text{ si } (\Gamma_j^n < \Gamma_j + \phi_1(t_j)) \wedge (w_j = 1) \end{cases}$$

Ejemplo: en la figura 4.1 se presenta el modelo al tiempo del SMF del capítulo dos con redes de Petri SPSRTF, con un marcado $[\ 7,1,1,1,1,1,1,0,1,1,0,3,1,0,0,0,4]^T$ y sus tiempos mínimos y máximos en que tarda en realizarse cada operación son los siguientes:

$\phi_1(t1) = 3.0,\ \phi_2(t1) = 3.5,\ \phi_1(t2) = 3.0,\ \phi_2(t2) = 3.5$
$\phi_1(t3) = 4.8, \phi_2(t3) = 5.0,\ \phi_1(t4) = 4.8,\ \phi_2(t4) = 5.0$
$\phi_1(t5) = 3.0, \phi_2(t5) = 3.5,\ \phi_1(t6) = 3.0, \phi_2(t6) = 3.5$
$\phi_1(t7) = 3.5,\ \phi_2(t7) = 4.0,\ \phi_1(t8) = 3.5,\ \phi_2(t8) = 4.0$
$\phi_1(t9) = 3.0,\ \phi_2(t9) = 3.5,\ \phi_1(t10) = 3.0,\ \phi_2(t10) = 3.5$
$\phi_1(t11) = 3.5,\ \phi_2(t11) = 4.0,\ \phi_1(t12) = 3.5,\ \phi_2(t12) = 4.0$

Se presenta el marcado al tiempo β, es decir $\xi_1 = \beta$ y $\tau_1 = \xi_1 + 1$ (el tiempo en que la señal de control $u_1(\ \tau_1\ )$ cambia de cero a uno) entonces $\Gamma_1 = \xi_1 + 1$ y es el tiempo en que inicia su disparo t_1. Al tiempo Γ_1 se tiene que $M_k(l_{ent}) = 6$.

La transición termina su disparo y el marcado va a evolucionar al tiempo Γ_1^n, dependiendo de las funciones de tiempo $\phi_1(t_1), \phi_2(t_1)$, del tiempo en que se active la señal w_1. Se tienen los siguientes casos:

$$M_{k+1}(l_{ent}) = \begin{cases} 6, \text{ si } ((\Gamma_1 + 3 \leq \Gamma_1^n \leq \Gamma_1 + 3.5) \wedge (w_1 = 1)) \\ \vee\ (\Gamma_1^n > \Gamma_1 + 3.5) \\ 7, \text{ si } (\Gamma_1^n < \Gamma_1 + 3) \wedge (w_1 = 1) \end{cases}$$

$$M_{k+1}(l_g) = \begin{cases} 1, \text{ si } ((\Gamma_1 + 3 \leq \Gamma_1^n \leq \Gamma_1 + 3.5) \wedge (w_1 = 1)) \\ \vee\ (\Gamma_1^n > \Gamma_1 + 3.5) \\ 0, \text{ si } (\Gamma_1^n < \Gamma_1 + 3) \wedge (w_1 = 1) \end{cases}$$

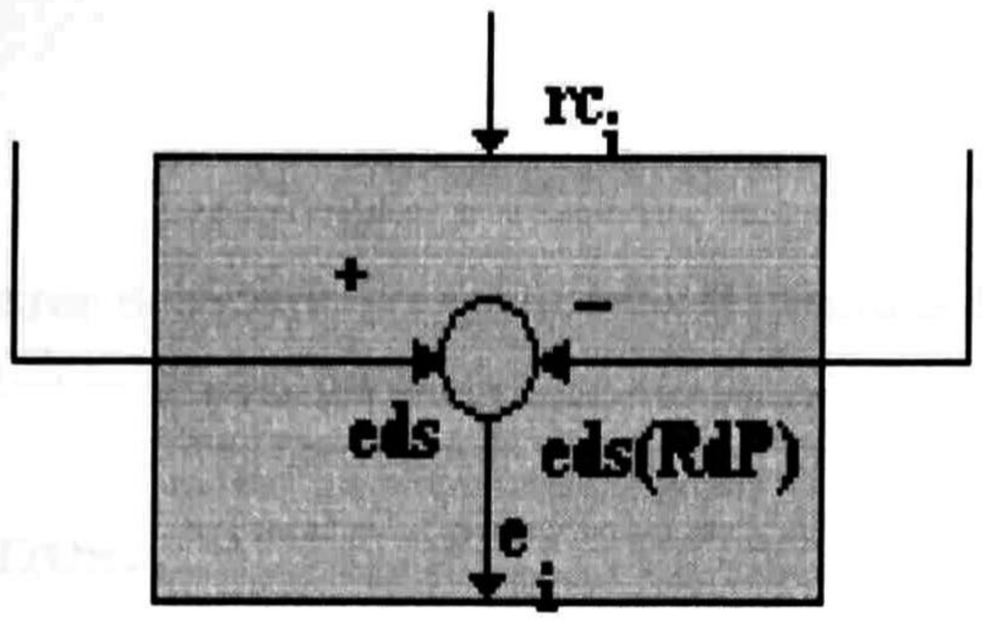


Figura 4.3:

4.4 Sistema de Detección del error.

En este trabajo el estado actual del sistema es llamado M_S, son datos obtenidos por los sensores, que nos indica cuántas piezas están en el almacén, que operación es la próxima a efectuarse sobre una pieza y cuántas están terminadas. Además nos indica si una máquina se encuentra o no disponible. El estado actual del modelo es llamado M_M, que es la función de marcado de la red de Petri.

Para detectar el error, al estado actual del sistema se le resta el estado actual del modelo, como se indica en la figura 4.3. Se obtiene el vector e_i donde i es el número de veces que se ha calculado e. Si $e_i \neq \vec{0}$ el error será tratado por las siguientes etapas del **STF.** Si el **STF** no ha determinado cómo recuperarse, la señal rc_i inhibe los estados donde $e_i \neq 0$ para nuevos cálculos del vector e, esto para evitar que se trabaje varias veces sobre el mismo vector de error.

La señal de error e_i se calculará cuando termine de dispararse una transición del modelo o cuando se termine de realizar una operación en el sistema.

Por la forma en que se construye el modelo, su espacio de estados alcanzables es igual al espacio de estados válido del SMF, es decir:

$$\mathcal{R}(Modelo, M_0) = S(SMF)$$

Si la señal de error $e_i \neq \vec{0}$, se tienen los siguientes casos:

i) Se alcanzó un estado no válido.

$$M_S \notin \mathcal{R}(Modelo, M_0)$$

ii) Se alcanzó un estado válido pero no en el tiempo especificado.

$$M_S \in \mathcal{R}(Modelo, M_0)$$

Dependiendo el caso de error detectado se ejecutarán las rutinas de recuperación adecuada, esto se muestra en los siguientes capítulos.

4.5 Conclusiones.

En este capítulo se propuso una extensión a los SPSR para utilizar este tipo de redes en STF y se mostró como la regla de evolución permite capturar cuándo ocurre un error en el SMF. Se estableció el **SDE** que se utilizará en esta tesis, cómo trabaja y los elementos que lo constituyen. En el próximo capítulo se presenta la segunda etapa del **STF** que es el diagnóstico de fallas.

Capítulo 5

Aproximación propuesta: Diagnóstico de fallas.

Resumen: En este capítulo se definirá la etapa de Diagnóstico de Fallas, las aproximaciones usadas para crear un Sistema de Diagnóstico de Fallas (**SDF**), los problemas de los SDF de tiempo real. Por último se presenta el SDF propuesto en este trabajo el cual consta de dos etapas, la extracción de los recursos con posibles fallas y localización de los recursos con fallas.

5.1 Introducción.

La segunda etapa del STF es el diagnóstico de fallas (también conocida como localización del error o localización de la falla [23]). Por muchos años el diagnóstico automático ha sido una área activa en investigación tanto en aplicaciones técnicas como en aplicaciones médicas [19].

5.1.1 Diagnóstico basado en síntomas-fallas.

Generalmente el diagnóstico de fallas automático puede ser visto como un proceso secuencial que involucra dos pasos: la generación del síntoma y la tarea de diagnóstico. A continuación se presenta el primer paso del diagnóstico de fallas que es la generación del síntoma [20]:

- La generación del síntoma es encontrar cuándo cambia una variable observable, que se encontraba en un comportamiento normal a un comportamiento anormal. Principalmente es requerido para la reducción de los datos a analizar en el diagnóstico de fallas. Se distinguen dos clases:

  - Generación de síntomas analíticos. Son las discrepancias que resultan al comparar las señales medidas directamente del sistema, con las señales de los modelos o modelos de los procesos. Para obtener esas discrepancias se mencionan a continuación algunas técnicas:

    * Verificación del valor limite de señales medibles. Es verificar que las señales medidas del sistema se encuentren entre las tolerancias de las señales en operación normal (es decir, sin fallas).
    * El análisis de señales. Las señales medidas directamente del sistema son comparadas con modelos de señales como espectro de frecuencia o valores característicos por ejemplo varianza, amplitud, frecuencia o parámetros del modelo.
    * Análisis del proceso. Empleando modelos matemáticos del proceso, junto con estimación de parámetros, estimación de estados, para compararlos con las señales medidas directamente del sistema.

  - Generación de síntomas heurísticos. Además de la generación de síntomas usando información cuantificable, los síntomas heurísticos pueden ser producidos usando información cualitativa de los operadores humanos. La información cualitativa, como valores característicos de ruidos especiales, de colores, de olores, vibraciones, etc. Los datos estadísticos (probabilidades de falla), analizados de la experiencia con procesos similares pueden ser agregados. De está manera los síntomas heurísticos son generados los cuales pueden ser representados por variables lingüisticas.

- Tarea de diagnóstico consiste en interpretar los síntomas e identificar qué elemento del sistema tiene una falla. Para esta tarea se pueden distinguir los siguientes algorítmos:

- métodos de clasificación.
- métodos de inferencia.

Los métodos de clasificación incluyen clasificadores geométricos, estadísticos y difusos. También se pueden usar redes neuronales como un método de clasificación para el diagnóstico de fallas automático o técnicas híbridas neurodifusas [24]. Por el contrario, los métodos de inferencia son basados en reglas, a menudo se les llama métodos basados en reglas. Los métodos más usados de inferencia son los Sistemas Expertos y los de lógica difusa. Las reglas pueden ser formuladas en un modo expresivo (If-Then-Else). Se pueden ver aplicaciones de este método en [3].

5.1.2 Diagnóstico basado en el modelo.

El razonamiento basado en el modelo utiliza un modelo del dominio del conocimiento estructural y funcional acerca de determinado sistema [?].

Sea C un conjunto de componentes que interactuan para producir el comportamiento del sistema. Se asume que SC es el conjunto de especificaciones del comportamiento local de cada componente. El proceso de diagnóstico se divide en tres etapas.

Primero, utilizando el conjunto de especificaciones y los datos de entrada, el proceso de diagnóstico genera el comportamiento local de cada componente. En la segunda etapa las suposiciones que incluyen en las especificaciones y que subyacen este comportamiento local son contradictorias con el comportamiento local observado. En la tercer etapa el proceso de diagnóstico utiliza una o más técnicas, tales como la relajación de las restricciones de Davis, la poda del HS-tree de Reiter para encontrar los componentes defectuosos.

La adquisición de los datos puede ser más sencilla. Durante el diseño del sistema a ser modelado, se hace un examen detallado de la estructura del sitema y mucha de la información para crear el modelo del sistema está disponible desde que el sistema fue diseñado.

5.2 Diagnóstico de fallas.

Definiremos a continuación el concepto de diagnóstico de fallas.

Definición 5.2.1 *Diagnóstico de fallas [3]. Es identificar los componentes físicos que causan un comportamiento incorrecto de la planta.*

5.2.1 Tipos de problemas del diagnóstico de fallas.

Los problemas típicos encarados por los Sistemas de Diagnóstico de Fallas en Tiempo Real (**SDF**) son resumidos a continuación:

• Operación Continua: Los **SDF** deben ser capaces de operar continuamente incluso en diferentes fases de operación de la planta (una fase es la puesta en marcha del sistema, el sistema en operación normal es decir produciendo, etc.) La funcionalidad de secciones de

la planta así como de los componentes involucrados, pueden cambiar en diferentes fases. El problema que se presenta es que cada fase de operación tiene diferente modelo matemático, el **SDF** debe tomar en cuenta los diferentes modelos matemáticos de las fases y saber en que fase de operación se encuentra el sistema, para llevar acabo su objetivo.

• Eventos Asíncronos: Las fallas en una planta no se presentan conforme a un plan predefinido. Por lo tanto los **SDF** deben ser capaces de aceptar entradas de un evento asíncrono no planeado (es decir una falla) en cualquier momento.

• Razonamiento Temporal: En los ambientes de tiempo real, la relación del tiempo con los eventos detectados son esenciales para hacer deducciones correctas y significativas considerando sus fuentes.

• Tiempo de diagnóstico predecible: Debido a que las hipótesis de fallas generadas por los **SDF** son típicamente usadas para iniciar reacciones de recuperación o intervenciones para prevenir fallas catastróficas, el diagnóstico debe ser completado bajo limitaciones de tiempo.

• Fallas Sencillas o Múltiples: En el caso de una falla simple, sólo un componente físico es la causa de que la planta falle, mientras que en el caso de una falla múltiple más de un componente físico son la causa. El **SDF** debe ser capaz de detectar ambos casos.

• Lazos de Retroalimentación: Plantas dinámicas de gran escala contienen componentes que son parte de un lazo de retroalimentación. Una falla en uno de estos componentes puede ser aumentada o compensada por la retroalimentación.

• Compromiso de tiempo y respuesta: Los **SDF** deben ser capaces de hacer un diagnóstico no muy detallado cuando dispone de poco tiempo. El resultado puede ser refinado con el paso de más tiempo y proporciona más información.

• Alarmas: EL **SDF** debe de advertir al operador o a un programa de recuperación de errores acerca de los componentes fallidos de la planta.

5.3 Sistema de Diagnóstico de Fallas.

Para la etapa de diagnóstico de fallas se utiliza una aproximación que es un proceso dinámico basado en un modelo del sistema en RdP. El Diagnóstico de fallas sólo se activará si se detectó alguna anormalidad en el sistema, es decir la señal del error es diferente de cero, o por una señal externa del operador del sistema. El resultado del diagnóstico de fallas se mandará a la última etapa del STF que es la Recuperación del error. El proceso dinámico del diagnóstico de fallas automático cuenta con dos pasos (ver figura 5.1).

- Extracción de los recursos con posibles fallas.

- Localizar los recursos con fallas.

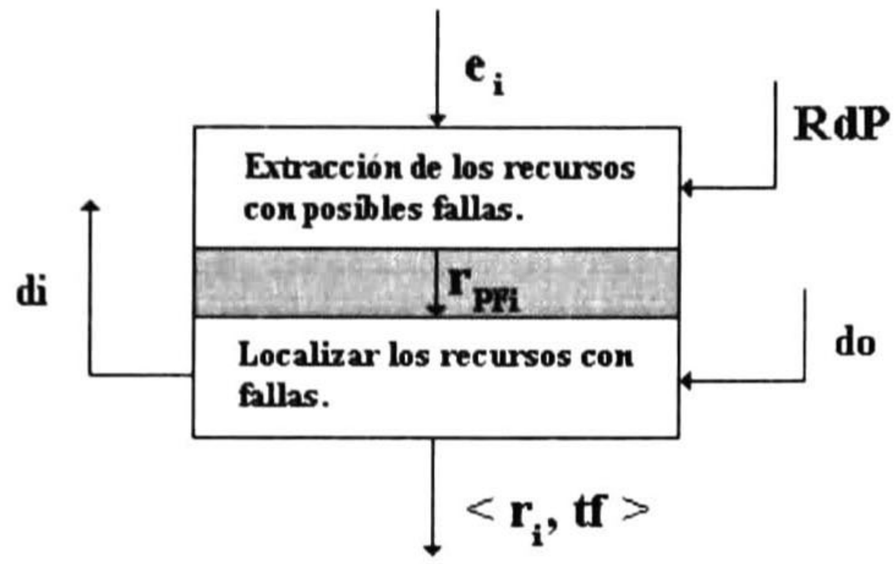


Figura 5.1:

5.3.1 Extracción de los recursos con posibles fallas.

Como se sabe un SMF cuenta con un número de recursos finitos (taladro, torno, fresadora, robots), estos recursos sirven para realizar diferentes procesos (taladrado, torneado, fresado, traslado de piezas etc.). Así cuando un recurso tiene una falla puede producir una degradación en el SMF. El SDF localiza los recursos con fallas, para que posteriormente la etapa de recuperación del error encuentre formas de que la falla de un elemento no cause un error posterior al sistema. Cuando existe una falla en algún componente físico de un recurso, se manifiesta como un error dentro del **SMF**. El **SDE** indica que existe un error, pero el diagnosticar qué componente físico está dañado y a qué recurso pertenece, se puede volver complicado debido al tamaño y el número de recursos con los que cuenta un SMF. Para resolver este problema el primer paso del **SDF** es reducir el número recursos dónde se puede localizar la falla. A este paso se le llama "extracción de los recursos con posibles fallas r_{PFi}", y consiste en encontrar una función tal que:

$$\mu : \mathcal{E} \rightarrow 2^{|Ef \cup M|}$$

Encontrar una expresión analítica para μ resulta difícil. En la mayoría de los casos se utilizan sistemas de diagnóstico basado en síntomas y razonamiento abductivo [20], [3]. Sin embargo con la ayuda de la estructura de la RdP éste problema se facilita.

El siguiente procedimiento muestra cómo encontrar r_{PFi}. Cuando ocurre un error $e_i \in \mathcal{E}$, y $r_{PFi} \in 2^{|Ef \cup M|}$, entonces

$$\mu(e_i) = r_{PFi}$$

donde r_{PFi} se calcula como se describe a continuación:

$$r_{PFi} = r_{PFi1} \cup r_{PFi2}$$

Donde r_{PFi1} es el conjunto de recursos que según el modelo deben estar disponibles en un tiempo definido "t", pero el SMF nos indica que no están disponibles, o que en el modelo

nos indica que no deben estar disponibles en un tiempo "t", pero en el SMF nos indica que están disponibles. Es decir:

$$L_r = \{l \in L_i^{'} \mid e_i \neq 0\}$$

$$r_{PFi1} = \{r_k \mid r_k = \psi^{-1}(l_i) \forall l_i \in L_r\}$$

Y r_{PFi2} es conjunto de recursos que deberían de haber realizado una operación dentro del SMF en un tiempo definido "t" Es decir:

$$\begin{aligned} Sea\ L_h &= \{l \in L_i \mid e_i \neq 0\} \\ Si &\mid L_h \mid = 1 \\ T_D &= ({}^\bullet l_j \cup l_j^\bullet) \\ Si &\mid L_h \mid > 1 \end{aligned}$$

$$T_D = \cup({}^\bullet l_j \cup l_j^\bullet \cap^\bullet l_k \cup l_k^\bullet) \text{ donde } l_j, l_k \in L_h\ ,\ j \neq k$$

$$L_p =^\bullet t_i \cap t_i^\bullet, \forall t_i \in T_D$$

$$r_{PFi2} = \{r_k \mid r_k = \psi^{-1}(l_i) \forall l_i \in L_p\}$$

Los recursos $r_k \in r_{PFi}$ se les etiqueta como no disponibles, si están realizando alguna operación se detiene y se etiqueta como no disponible porque serán diagnosticados.. En el modelo $M(l_i) = 0\ \forall\ l_i \in L_r \cup L_p$

5.3.2 Localizar los recursos con fallas.

Después que se obtuvo el conjunto de los recursos con posibles fallas r_{PFi} se debe determinar qué recursos tienen fallas. Localizar los recursos con fallas $r_{fi} \in 2^{|r_{PFi}|}$, es encontrar la función:

$$\vartheta : Ef \cup M \rightarrow 2^{|r_{PFi}|},\ \text{donde } \vartheta(r_{PFi}) = r_{fi}$$

Encontrar una expresión analítica para ϑ resulta difícil y en este trabajo se calcula utilizando un Sistema Experto.

Sistema Experto

El sistema experto cuenta con bases de conocimientos llamadas librerías de pruebas que nos ayudarán a calcular r_{fi}, esta idea fue tomada de [3]. Los pasos a seguir para encontrar r_{fi} son los siguientes:

- Evaluación de la situación. Se consulta el conjunto de librerías de pruebas para encontrar las librerías de pruebas adecuadas a los recursos con posibles fallas r_{PFi}

- Cargar las librerías de pruebas. Las librerías de pruebas de los recursos r_{PFi} se ejecutan, para buscar el recurso que presenta la falla.

  - Resultados de las librerías de prueba.

    * Si los resultados al aplicar las librería de pruebas nos indican que no existe ninguna falla en r_{PFi}, se trató sólo de una falla transitoria, es decir se presentó en período de tiempo muy pequeño, ocasionó el error y después desapareció. Se denota como:

$$< r_i, Transitoria >$$

$$\forall r_i \in r_{PFi}$$

Se les quita la etiqueta de no disponibles y se etiqueta como disponibles, en el modelo $M(l_i) = 1 \forall l_i \in L_r \cup L_p$.

* Si los resultados al aplicar las librerías de pruebas, nos indican que existe una falla en un recurso $r_k \in r_{PFi}$, entonces se trata de una falla permanente. Se denota como:

$$< r_k, Permanente >$$

El recurso con falla r_k se queda con la etiqueta de no disponible, $r_i \in r_{PFi} - \{r_k\}$ se etiquetan como disponibles, en el modelo $M(l_i) = 1 \forall l_i \in \{L_r \cup L_p - \psi(r_k)\}$

- Resultados del Diagnóstico. Una falla transitoria puede ocasionar muchos errores, si se presenta cada cierto tiempo (falla intermitente). Por lo tanto se debe de llevar una contabilidad de cuántas veces se presenta una falla transitoria en un recurso. Además se debe de aceptar un número máximo de fallas transitorias Υ en un periodo de tiempo "t", si se excede de ese número se tomará como una falla permanente. Lo anterior define que el sistema de diagnóstico es dinámico es decir:

Si $< r_i, Tipo\ de\ falla,\ n >$ representa el recurso r_i, el tipo de falla que tiene y el número n indica las veces que el recurso ha tenido fallas transitorias, entonces los resultados del sistema de diagnóstico se modela como en la figura **??** y el diagnóstico actual DA se calcula como:

Si $rl(r_i) =< r_i, Transitoria >$, $da =< r_i, Transitoria,\ q >$

$$DA(r_i) = \begin{cases} < r_i, Transitoria,\ q+1 > Si\ q+1 < \Upsilon \\ < r_i, Permanente,\ 0 > Si\ q+1 = \Upsilon \end{cases}$$

Si $rl(r_i) =< r_i, Permanente >$, $da =< r_i, Transitoria,\ q >$

$$DA(r_i) = < r_i, Permanente,\ 0 >$$

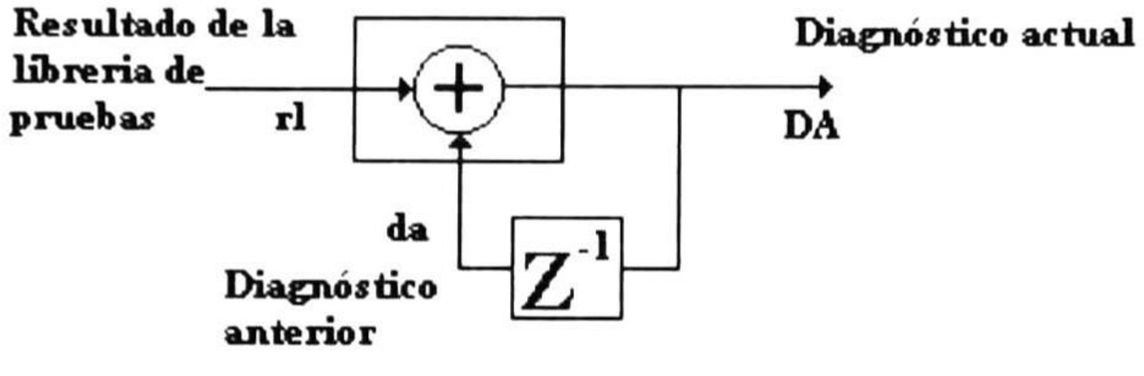


Figura 5.2:

Librerías de prueba.

Es una base de conocimientos dentro del Sistema Experto, que contiene un conjunto de reglas llamadas pruebas de diagnóstico. Una vez que la prueba de diagnóstico ha sido ejecutada se carga la siguiente prueba de diagnóstico de la librería de prueba.

Para elaborar una librería de prueba, se debe tener amplio conocimiento del recurso (máquina o efector); como son los modelos matemáticos; las fallas más frecuentes y cómo detectar dichas fallas, etc.

- La prueba de diagnóstico.

Las pruebas de diagnóstico son reglas del tipo IF-THEN-ELSE que contienen información acerca de los elementos (sensores, actuadores, controladores) del recurso que se desea diagnosticar o de los procesos en el que interviene el recurso en operación normal, es decir en la ausencia de fallas. Además contiene información (di) sobre las mediciones o datos que se deben realizar al recurso que se desea diagnosticar. La información de las mediciones obtenidas a través de sensores (do) , la información del sistema experto junto con las reglas del sistema experto determinan el resultado de cada prueba. Las pruebas de diagnóstico son muy especificas, destinadas a causas de diagnóstico de fallas. Las pruebas de diagnóstico de fallas pueden ser de cualquier tipo: cuantitativas, cualitativas, heurísticas y basadas en modelos, nuevas pruebas pueden ser desarrolladas y agregadas a las diferentes librerías de prueba en cualquier momento.

A continuación se presenta información sobre cómo se han diagnosticado fallas de 1991 a 1996 [21] en algunas máquinas, robots, con esa información se pueden realizar pruebas de diagnóstico.

- **Robot industrial.**

  - *Para diagnosticar fallas en el proceso*. Estimación de parámetros con un modelo detallado, un observador con un modelo no lineal, evaluación estadística, y la historia del proceso.

- *Para diagnosticar fallas en los sensores, y en el proceso.* Estimación de parámetros con un modelo no lineal, supervisión en tiempo, análisis de frecuencia.

- **Taladro.**

  - *Para diagnosticar fallas en el proceso.* Análisis espectral de la frecuencia usando reconocimiento de patrones con redes neuronales, estimación de parámetros con un modelo lineal reducido.

- **Esmeril.**

  - *Para diagnosticar fallas en el proceso.* Estimación de parámetros con una aproximación del modelo de la señal.

- **Torno.**

  - *Para diagnosticar fallas en el proceso.* Clasificación de las señales de multisensores.

- **Fresa.**

  - *Para diagnosticar fallas en el proceso.* Análisis espectral de la frecuencia usando reconocimiento de patrones con lógica difusa.

Por ejemplo de como hacer una prueba de diagnostico para encontrar una falla en sensor de un robot industrial podría ser:

IF (Efectuar di, do = a los parámetros obtenidos del modelo lineal) Then
("No existe falla en sensor").
Else
("Existe falla en el sensor").

5.4 Conclusiones.

En este capítulo se propuso el uso de las RdP para localizar los recursos a analizar (por-que pueden tener fallas), esto es una ventaja sobre los esquemas que utilizan razonamiento abductivo [3], ya que pueden localizar un número de recursos o ser demasiado lentos en su razonamiento. En la etapa de análisis de los recursos se úso un sistema experto. Finalmente, un sistema dinámico determina cuál es el tipo de falla que tiene el recurso. Éstos resultados del diagnóstico se utilizarán en el sistema de recuperación del error que se verá en el siguiente capítulo.

Capítulo 6

Aproximación propuesta: Recuperación del error.

Resumen: En este capítulo presenta la etapa de la recuperación del error, se analiza diferentes aspectos como son el tipo de falla que diagnóstico el SDF, el SMF en caso de una degradación de su funcionalidad, los posibles bloqueos del sistema debido a fallas en algunas máquinas y por último a partir de éstos análisis se presenta el algoritmo del sistema de recuperación del error que se propone en esta tesis.

6.1 Introducción.

Una vez que el error ha sido detectado y se han diagnosticado las fallas, entonces el procedimiento de recuperación del error debe iniciar.

Definición 6.1.1 ***Recuperación del error*** *[13]. Es un conjunto de rutinas que se deben realizar para llevar al sistema, de un estado de error a un estado válido aunque quizá, con un sistema degradado.*

Antes de iniciar las rutinas de recuperación es muy importante analizar las clases de fallas que encontró el **SDF**. Si se encontraron fallas transitorias el plan de proceso de nuestro sistema sigue siendo el mismo, es decir existen pequeñas perdidas en la cadencia de producción pero no son significativas (nivel de tolerancia a fallas total). Entonces surge el siguiente problema:

- ¿Existirá la secuencia de operaciones que lleve el estado de error a un estado válido al sistema?

Si existen fallas permanentes en algunos recursos, éstos no se podrán utilizar hasta que sean reparados. Por lo tanto las operaciones en las que intervienen los recursos con fallas permanentes se deben eliminar del plan de proceso del controlador óptimo (nivel de tolerancia a fallas de degradación suave) y del modelo. Debido a lo anterior surgen los siguientes problemas:

- ¿El **SMF** podrá seguir produciendo la misma variedad de productos, cuando existen fallas permanentes?
- ¿ Qué productos no podrá seguir produciendo el SMF?
- ¿Cómo garantizar que el **SMF** degradado no se bloquea después de realizar las rutinas de recuperación?
- ¿Existirá la secuencia de operaciones que lleve el estado de error a un estado válido al sistema?

En las próximas secciones se analizan los problemas anteriores.

6.1.1 Análisis de clases de fallas.

El SDF calculó el vector DA.

I. Si las fallas en los recursos son fallas transitorias la funcionalidad del sistema se conserva y se debe de buscar la secuencia de recuperación del error.

II. Si las fallas en los recursos son fallas permanentes la funcionalidad del sistema se degrada ya que se tienen que eliminar las operaciones en las que intervienen esos recursos.

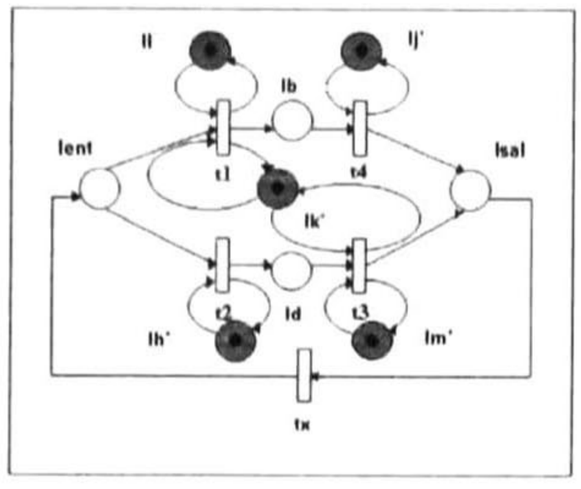


Figura 6.1:

Sea R_{fp}el conjunto de recursos con fallas permanentes y los lugares que los representan en el modelo son:

$$l_{fp} = \{l_i \mid l_i = \psi(r_{fp}) \forall r_{fp} \in R_{fp}\}$$

entonces el conjunto de transiciones a eliminar del modelo se calcula:

$$T_{fp} = \{t_i \mid t_i \in \cup^{\bullet} l_i \text{ donde } l_i \in l_{fp}\}$$

y las operaciones a eliminar son:

$$Op_{fp} = \{Op_i \mid Op_i = \lambda^{-1}(t_i) \forall t_i \in T_{fp}\}$$

Las operaciones $Op_i \in Op_{fp}$ ya no podrán ser realizadas (a menos de que reparen los recursos), por esta razón el control óptimo ya no debe tomarlas en cuenta. Dado que el control óptimo ya no debe permitir el disparo de $t_i \in T_{fp}$,esta transición puede ser eliminada o desconectada del modelo, y este nuevo modelo sirve para representar al sistema degradado. Se prefiere eliminar T_{fp} del modelo, porque los algorítmos de recuperación del error y de control óptimo hacen uso de esta información, de esta manera se les indicaría automáticamente que dichas operaciones ya no están disponibles.

Para eliminar una t_i basta con quitar la columna correspondiente a t_i de las matrices C, C^+, C^-

Por ejemplo, se tiene un **SMF**, cuyo se representa en RdP como en la figura 6.1 entonces su matriz de incidencia es:

$$C = \begin{array}{c} \\ l_{ent} \\ l_b \\ l_d \\ l_i' \\ l_j' \\ l_k' \\ l_h' \\ l_m' \\ l_{sal} \end{array} \begin{bmatrix} t_1 & t_2 & t_3 & t_4 & t_x \\ -1 & -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

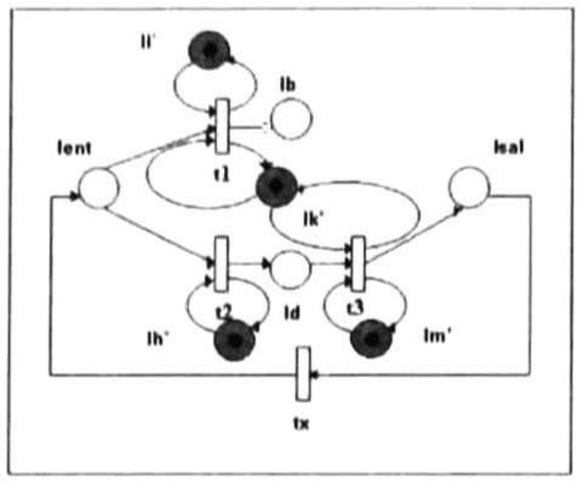


Figura 6.2:

Si existe una falla permanente en el recurso $\psi^{-1}(l'_j)$ por lo tanto $T_{fp} = \{t_4\}$al quitarla de la matriz de incidencia nos queda:

$$C = \begin{bmatrix} & t_1 & t_2 & t_3 & t_x \\ l_{ent} & -1 & -1 & 0 & 1 \\ l_b & 1 & 0 & 0 & 0 \\ l_d & 0 & 1 & -1 & 0 \\ l'_h & 0 & 0 & 0 & 0 \\ l'_i & 0 & 0 & 0 & 0 \\ l'_j & 0 & 0 & 0 & 0 \\ l'_k & 0 & 0 & 0 & 0 \\ l'_m & 0 & 0 & 0 & 0 \\ l_{sal} & 0 & 0 & 1 & -1 \end{bmatrix}$$

Al construir la red de Petri, a partir de la nueva matriz de incidencia nos queda una nueva red de Petri como se muestra en la figura 6.2, a esta la llamaremos modelo degradado.

En la siguiente sección se analiza al SMF que ha sufrido una degradación física (recursos con fallas permanentes). El análisis se realiza para determinar si el SMF sigue produciendo la misma familia de productos y para saber si algún producto se dejará de producir. Generalmente, cuando existe un recurso con falla permanente disminuye significativamente la cadencia de producción del SMF; el estudio del desempeño y nuevo scheduling no son abordados en este trabajo.

6.1.2 Análisis de la familia de productos en un SMF degradado.

Para el análisis de la familia de productos en un SMF degradado solo se tomará en cuenta la red de Petri formada por las secuencias de procesado es decir las redes PS del modelo.

A continuación se definen algunos conceptos que se utilizarán para saber cuando es posible que un SMF degradado pueda seguir produciendo la misma familia de productos.

Definición 6.1.2 *Un lugar l_i es llamado de selección sí,*

$$l_i^{\bullet} > 1.$$

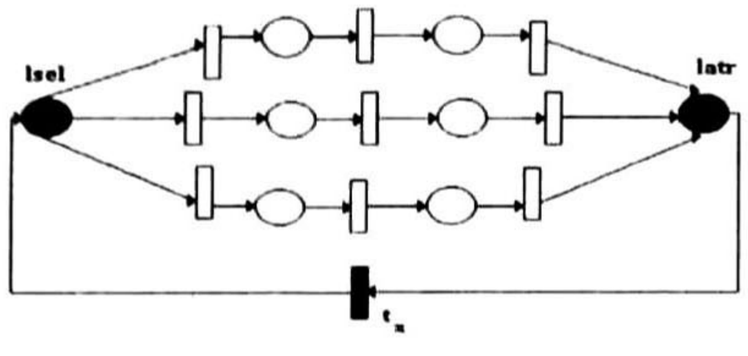


Figura 6.3: Un par de selección-atribución formado por lsel y latr.

Definición 6.1.3 *Un lugar l_k es llamado de atribución sí,*

$$^\bullet l_k > 1.$$

Definición 6.1.4 *En una MEFC si $C_1 = \{l_i, t_i, ...l_k\}$ y $C_2 = \{l_i, t_j, ...l_k\}$ son dos caminos dirigidos de l_i a l_k que no cruzan por t_x⊣*

$$\{C_1 - \{l_i, l_k\}\} \cap \{C_2 - \{l_i, l_k\}\} = \emptyset$$

y el ciclo no dirigido $\mathfrak{C}_{l_j l_k}$ que se forma con C_1 y C_2
Un (l_i, l_k) es llamado un par de selección-atribución sí

$$l_i^\bullet > 1 \wedge^\bullet l_k > 1$$

y entre l_i , l_k se forma por lo menos un $\mathfrak{C}_{l_j l_k}$

Ejemplos, en la figura 6.3 $< l_{sel}, l_{atr} >$ forman un par de selección-atribución.

Definición 6.1.5 *Sea $\theta = \{\vec{\gamma_i} | \vec{\gamma_i}$ un vector de disparos que representan las secuencias de procesado$\}$.*

Lema 6.1.1 *Sea el conjunto de secuencias de procesado a eliminar θ_k por fallas permanentes. Es posible eliminar θ_k y que el sistema puede seguir produciendo la misma familia de productos si y solo sí:*

$$C\,\vec{\gamma_k} = C \sum_{\vec{\gamma_i} \in \{\theta - \theta_k\}} \alpha_i\,\vec{\gamma_i} \quad \forall \gamma_k \in \theta_k \text{ y } \vec{\gamma_k} \text{ es el vector de disparos de } \gamma_k.$$

Demostración:
→
1)Tenemos que $M_o\,[\vec{\gamma_k} > M_f$ entonces en la RdP ocurre $M_f = M_o + C.\,\vec{\gamma_k}$
2)Si $\vec{\gamma_k}$ se elimina y se sigue produciendo la misma familia de productos

$$\rightarrow \exists \gamma_l \in \{\theta - \theta_k\} \dashv M_o[\gamma_l > M_f \rightarrow \vec{\gamma_l} = \sum_{\vec{\gamma_i} \in \{\theta - \theta_k\}} \alpha_i\,\vec{\gamma_i}$$

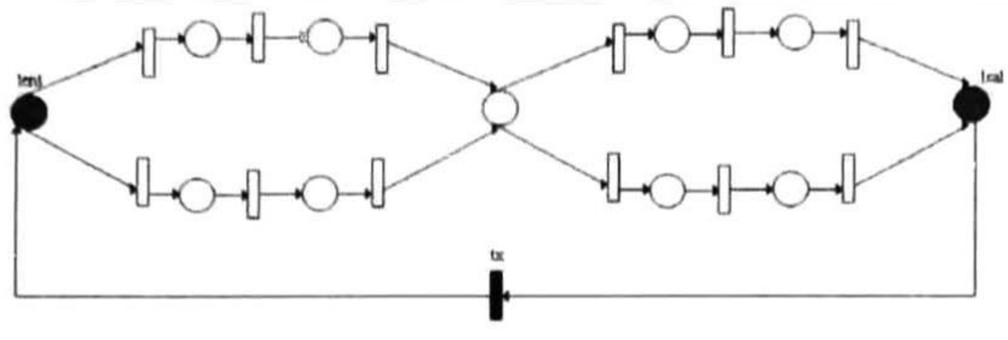


Figura 6.4: Red de Petri con 4 secuencias de procesado.

entonces en la RdP ocurre

$$M_f = M_o + C \sum_{\vec{\gamma_i} \in \{\theta - \theta_k\}} \alpha_i \vec{\gamma_i}$$

de 1) y 2) igualamos los M_f

$$M_o + C \vec{\gamma_k} = M_o + C \sum_{\vec{\gamma_i} \in \{\theta - \theta_k\}} \alpha_i \vec{\gamma_i}$$

eliminando M_o concluimos que

$$C \vec{\gamma_k} = C \sum_{\vec{\gamma_i} \in \{\theta - \theta_k\}} \alpha_i \vec{\gamma_i}$$

$\leftarrow$

$$C \vec{\gamma_k} = C \sum_{\vec{\gamma_i} \in \{\theta - \theta_k\}} \alpha_i \vec{\gamma_i} \text{ si } \vec{\gamma_l} = \sum_{\vec{\gamma_i} \in \{\theta - \theta_k\}} \alpha_i \vec{\gamma_i}$$

entonces

$$C \vec{\gamma_k} = C \vec{\gamma_l}$$

si sumamos el marcado M_o a ambos lados de la ecuación tenemos

$$M_o + C \vec{\gamma_k} = M_o + C \vec{\gamma_l}$$

de la ecuación de la RdP $M_f = M_o + C \vec{\gamma_k} \wedge M_f = M_o + C \vec{\gamma_l}$

por lo tanto se puede seguir produciendo la misma familia de productos eliminando las secuencias $\gamma_k \in \theta_k$, y utilizando las secuencias $\gamma_l \in \{\theta - \theta_k\}$.

Sin embargo el problema del cálculo de las secuencias de procesado es NP

Tomar la RdP de la figura 6.4 El número de secuencias de procesado de la red es cuatro.

Ahora si aumentamos una trayectoria en cada par selección-atribución tendremos la figura 6.5, veremos como las secuencias de procesado aumentan de una manera no polinomial. El número de secuencias de procesado aumentan a 9.

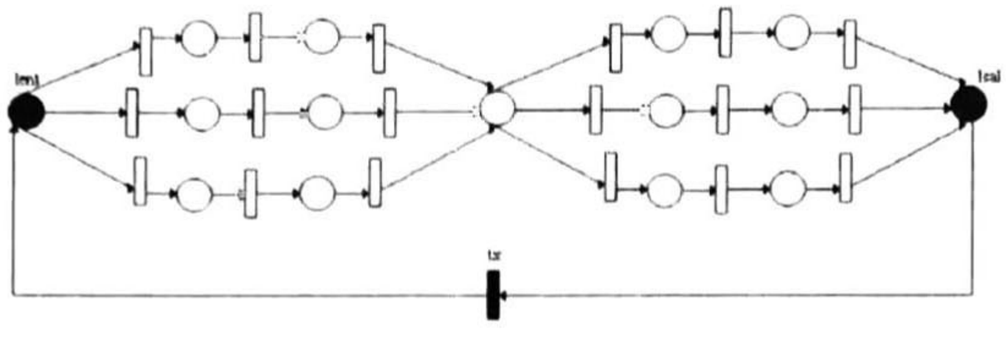


Figura 6.5: Red de Petri con 9 secuencias de procesado.

Para no tener que calcular todas las secuencias de procesado, necesitamos de las siguientes definiciones:

Definición 6.1.6 *Los predecesores de primer, segundo, y n orden de t_i se representan como sigue:*

$$Sea\ {}^\bullet t_i\ =^1 t_i,\ {}^\bullet({}^\bullet t_i)\ =^2 t_i,\ {}^\bullet({}^\bullet({}^\bullet t_i))\ =^3 t_i,\ ...y\ así\ sucesivamente$$

Definición 6.1.7 *Se dice que t_i viene de un lugar fuente sí,*

$$^{2q+1}t_i = \{\emptyset\}\ donde\ q = 0, 1, 2, 3, ...n$$

Definición 6.1.8 *Se dice que t_i se encuentra entre el par selección-atribución (l_i, l_j) si, $\exists$ un camino C_1 de l_i a t_i $\exists t_x \notin C_1$ y un camino C_2 de t_i a l_j $\exists t_x \notin C_2$.*

Definición 6.1.9 *Sea $T_{SA} = \{t_i \in T$ |es un conjunto de transiciones que se encuentran al menos entre un par de selección-atribución$\}$.*

Teorema 6.1.1 *Sea t_i la transición que representa la operación donde se localizan fallas permanentes. Se puede eliminar t_i y seguir produciendo la misma variedad de productos si y sólo sí t_i se encuentra al menos entre un par selección-atribución y existe una camino de l_{ent} a t_i o t_i viene de un lugar fuente.*

Demostración:

$\rightarrow$

Se puede eliminar $t_i \wedge$ seguir produciendo la misma variedad de productos $\rightarrow (t_i \in T_{SA} \wedge \exists$ un camino de l_{ent} a $t_i) \vee t_i$ viene de un lugar fuente.

Sea $\Omega = \{\varpi_i \mid \varpi_i$ es un camino que lleva de l_{ent} a $l_{sal}\}$

Si se elimina $t_i \in \varpi_k \wedge$ se siguen produciendo la misma variedad de productos $\rightarrow \exists$ otro camino ϖ_l que no contiene a t_i.

Si ϖ_k y ϖ_l se pueden representar como $\varpi_k = \alpha S_1 \beta$ y $\varpi_l = \alpha S_2 \beta$ donde $t_i \in S_1$ y $S_1 \cap S_2 = \emptyset$ y además S_1, S_2 son de longitud máxima; el lugar previo l_k a S_1y a S_2 es un lugar de selección; el lugar posterior l_r a S_1y a S_2 es un lugar de atribución.. $l_k S_1 l_r$ y $l_k S_2 l_r$

forman un ciclo no dirigido $\mathfrak{C}_{l_k l_r}$. Por definición t_i pertenece a un par selección-atribución que forman l_k y l_r.

En el caso de que no exista un camino de l_{ent} a l_{sal} que contenga a t_i y aún eliminando t_i se sigue produciendo la misma variedad de productos, entonces t_i no está en una secuencia de procesado y por lo tanto viene de un lugar fuente, en el otro caso, cuando cuando $t_i \notin T_{SA}$ y aún eliminando t_i se sigue produciendo y por tanto t_i viene de un lugar fuente.

$\leftarrow$

Se puede eliminar $t_i \wedge$ seguir produciendo la misma variedad de productos $\leftarrow (t_i \in T_{SA} \wedge \exists$ un camino de l_{ent} a $t_i) \vee t_i$ viene de un lugar fuente.

Si $t_i \in T_{SA} \rightarrow \exists\ l_s^\bullet > 1 \wedge^\bullet l_a > 1$ que se unen con dos caminos C_1y C_2 tal que $\{C_1 - \{l_i, l_k\}\} \cap \{C_2 - \{l_i, l_k\}\} = \emptyset$ y $t_i \in C_1$ entonces $t_i \notin C_2$. Por lo tanto todas las secuencias de procesado que contienen todas las transiciones de C_2 alcanzan el mismo estado final que las secuencias de procesado que contienen todas las transiciones de C_1, por lo anterior el sistema seguirá produciendo la misma familia de productos si se elimina t_i.

Si t_i viene de un lugar fuente, entonces al eliminar t_i no se eliminan secuencias de procesado por lo tanto se sigue produciendo la misma variedad de productos.

Corolario 6.1.1 *Si la transición a eliminar* $(t_i \in T_{SA} \wedge \exists$ *un camino de* l_{ent} *a* $t_i) \vee t_i$ *viene de un lugar fuente,* $t_i \in T_j$*, donde* j *es una clase de producto del* ***SMF****. Entonces si se podrá seguir produciendo el producto* j

Corolario 6.1.2 *Si la transición a eliminar(* $t_i \notin T_{SA} \vee \nexists$ *un camino de* l_{ent} *a* $t_i) \wedge t_i$ *no viene de un lugar fuente,* $t_i \in T_j$*, donde* j *es una clase de producto del* ***SMF****. Entonces no se podrá seguir produciendo el producto* j

Nota: Si no se puede producir ninguna clase de productos en el SMF, éste se lleva al último nivel de la Tolerancia a Fallas que es la **Tolerancia a Fallas de Estado Seguro**.

6.1.3 Análisis para evitar bloqueos en sistemas degradados.

En los casos donde ocurren fallas permanentes, como se mencionó anteriormente, la estructura física del SMF se degradada y por consiguiente la del modelo ver figura 6.2. Supongamos que se tiene la red de Petri de la figura 6.2, con un marcado inicial $[1,0,0,0,0,0,0,0,0,0]^T$ y se termina de disparar t_1, el nuevo marcado es $[0,1,0,0,0,0,0,0,0,0\ ]^T$, si se observa en la red ninguna transición va a estar habilitada es decir la red se bloqueo. Para garantizar la ausencia de bloqueos se deben eliminar las operaciones que lleven a un estado de bloqueo. Analizando el modelo se pueden obtener el conjunto de transiciones T_b que llevan el sistema a un estado de bloqueo. Con el modelo y la transición que se eliminó se obtienen las transiciones que se deben eliminar del modelo y las operaciones que se deben de eliminar del controlador del SMF.

- Si se da el caso del corolario 6.1.1, entonces las transiciones a eliminar del modelo para evitar bloqueos son:

$$T_b = \{^{2r}t_i \mid r = 1, 2, ..., q \wedge \ (^{2q+1}t_i = \{l_{sel}\} \vee^{2q+1} t_i = \{\emptyset\})\}$$

T_b contiene a todas las transiciones que están entre el lugar de selección y la transición eliminada.

- Si se da el caso del corolario 6.1.2, entonces las trasiciones a eliminar del modelo para evitar bloqueos son:

$$T_b = \{^{2r}t_i \mid r = 1, 2, ..., q \wedge \ ^{2q+1}t_i = \{l_{ent}\}\}$$

Las operaciones a eliminar del controlador se calculan de la siguiente manera:

$$Op_b = \{Op_i \mid Op_i = \lambda^{-1}(t_i) \forall t_i \in T_b\}$$

No se deben tomar en cuenta en el cálculo de e los estados $M(l_i)$ donde $l_i \in T^{\bullet}_{bi}$, estados, ésto para evitar que al momento de eliminar las operaciones, alguna operación se este ejecutando y después se presenté una falla.

6.1.4 Búsqueda de la secuencia de recuperación del error.

Para la recuperación del error existen dos enfoques: recuperación hacia adelante y recuperación hacia atrás [13].

Definición 6.1.10 ***Recuperación hacia adelante.*** *Intenta continuar desde un estado de error del sistema, haciendo correcciones para llevarlo a un estado válido que está adelante.*

Definición 6.1.11 ***Recuperación hacia atrás.*** *Consiste en el restablecimiento del sistema a partir de un estado seguro previo al estado donde ocurrió el error.*

Definición 6.1.12 ***Punto de recuperación.*** *Es el punto donde se restaura un sistema, es decir es el estado válido al que se lleva el sistema que se encontraba en un estado de error.*

Definición 6.1.13 ***La secuencia de recuperación.*** *Es una secuencia de operaciones del sistema que se deben realizar para llevar el estado de error del sistema a un punto de recuperación y la denotaremos como ρ.*

Para llevar acabo la búsqueda de secuencia de recuperación del error se propone en esta sección una recuperación del error hacia adelante, utilizando la ecuación de estados de RdP:

$$M_{k+1} = M_k + C.\,\vec{\sigma} \tag{6.1}$$

Cuando el sistema cae en un estado de error r_e, se puede calcular la distancia más corta entre r_e y algún estado válido del sistema (punto de recuperación). Esta tarea no resulta

fácil. Otra forma es llevar al sistema de r_e a el $M_M(RdP)$ (estado válido y que debía alcanzar el SMF). El cálculo de una secuencia ρ que lleve r_e a el $M_M(RdP)$ se apoya en la estructura de la RdP, sin embargo esta secuencia no siempre existe y entonces se tendría que buscar otro estado válido y ver si existe una nueva secuencia.

En vez de seguir una aproximación como la anterior, se prefiere usar como estado válido el estado final del sistema $M_f(RdP)$ nuevamente la secuencia ρ para llevar el sistema de r_e al $M_f(RdP)$ puede no existir. Sin embargo este caso resulta muy interesante porque si éste no tiene solución podemos garantizar que ningún otro lo tendrá, y si éste tiene solución, puede que otros estados válidos la tengan, pero ya se encontró uno y en tiempo polinomial.

Propiedad 6.1.1 *Sea r_e el estado de error y $M_f(RdP)$ el estado final del SMF. Si $\nexists \rho \exists$ $r_e[\rho > M_f(RdP)$ entonces $\nexists$ $\rho' \exists r_e[\rho' > M_V(RdP)$ donde $M_V(RdP)$ es un estado válido del sistema.*

Demostración:

Suponer que $\exists$ $\rho' \exists$ $r_e[\rho' > M_V(RdP)$, entonces como la red es una MEFC $\exists$ $\rho_j \exists$ $M_V(RdP)[\rho_j > M_f(RdP)$ y la secuencia ρ' ρ_j sería tal que $r_e[\rho'\rho_j > M_f(RdP)$ por lo tanto se demuestra que la propiedad es cierta.

A continuación se presenta como se implemento el algoritmo de la búsqueda de la secuencia de recuperación ρ, para esto se utiliza las siguientes definiciones:

Definición 6.1.14 *El conjunto de lugares que se encuentran en un estado de error es:*

$$L_{iH} = \{l \mid e_i(l) \neq 0 \text{ donde } l \in L_i,\ i = 1, 2, 3...n\}$$

Definición 6.1.15 *$M_f(l)$ es el punto de recuperación y es el estado final del sistema $M_f(RdP)$, éste se calcula como:*

$$M_f(l_{isal}) = \sum M_M(l_j) \text{ donde } l_j \in L_{iH}$$

los marcados restantes de la red se igualan a cero para cuestión de cálculo.

Como el punto de recuperación es el estado final del sistema es decir $M_f(l)$ es el estado próximo de una RdP, se pone en cero la columna que corresponda a las transiciones t_x en la matriz de incidencia, y el estado actual de la RdP es el estado de error r_e. Por medio de la ecuación de estados de una RdP se tratará de alcanzar un estado válido a partir del estado de error, es decir:

$$M_k = M_S(l) \text{ (es el estado de error } r_e)$$
$$\text{y } M_{k+1}(l) = M_f(l) \text{ donde } l \in L_{iH}$$

Encontrar $\sigma \exists M_k[\sigma > M_{k+1}$ es equivalente a encontrar la secuencia de recuperación ρ.

Tomando en cuenta la estructura de la RdP tenemos la siguiente propiedad.

Propiedad 6.1.2 *Una condición necesaria para que exista una secuencia de recuperación ρ es que el siguiente problema tenga solución*

$$min \sum_i \vec{\rho(i)}$$

$$M_V(l) = M_S(l) + C.\ \vec{\rho}$$
$$\rho \geqslant 0$$

Demostración: se obtiene viendo que la ecuación de estados de la RdP es condición necesaria para que un marcado sea alcanzable [5].

A continuación mostraremos algunas tablas de resultados para solucionar el problema de la búsqueda de la secuencia de recuperación del error, dependiendo del espacio de estados al cual pertenece el estado de error del sistema y del tipo de falla que se trate:

Condición:	*Si* $M_S(l) \in \mathcal{R}(Modelo, M_o) \wedge$ *las fallas que se presentan en los recursos son fallas transitorias.*
Punto de recuperación.	*El estado final.*
Existe la secuencia de recuperación ρ.	*Siempre existe* ρ.

Condición:	*Si* $M_S(l) \in \mathcal{R}(Modelo, M_o) \wedge$ *las fallas que se presentan en los recursos son fallas permanentes.*
Punto de recuperación.	*El estado final.*
Existe la secuencia de recuperación ρ.	*Puede o no existir* ρ.

Condición:	*Si* $M_S(l) \notin \mathcal{R}(Modelo, M_o) \wedge$ *las fallas que se presentan en los recursos son fallas transitorias.*
Punto de recuperación.	*El estado final.*
Existe la secuencia de recuperación ρ.	*Nunca existe* ρ.

Condición:	*Si* $M_S(l) \notin \mathcal{R}(Modelo, M_o) \wedge$ *las fallas que se presentan en los recursos son fallas permanentes.*
Punto de recuperación.	*El estado final.*
Existe la secuencia de recuperación ρ.	*Nunca existe* ρ.

Las rutinas de recuperación dependen del tipo de falla y del espacio de estados al cual pertenece el estado de error del sistema.

En los casos I, II, si **existe** ρ el marcado del modelo se iguala al marcado de error del SMF. Si **no existe** ρ, no existe un punto de recuperación para el estado de error. Entonces se deben de sacar las piezas que se encuentran en estado de error y el marcado del modelo donde $e_i \neq 0$ se iguala a cero.

En los casos III, IV, si $\sum M_M(l_j) > \sum M_S(l_j)$ donde $l_j \in L_{iH}$, como **no existe** ρ el marcado del modelo se iguala al marcado de error del sistema.

En los casos III, IV, si $\sum M_M(l_j) = \sum M_S(l_j)$ donde $l_j \in L_{iH}$ como **no existe** ρ se deben de sacar las piezas que se encuentran en estado de error y el marcado del modelo donde $e_i \neq 0$ se iguala a cero.

Para indicar que la etapa de recuperación de e_i finalizó se activa la señal rc_i.

A continuación se presenta el algoritmo del sistema de recuperación del error:

Algoritmo del sistema de recuperación del error.

General

IF($R_{fp} \neq \emptyset$) *Then* (calcular T_{fp})

While ($T_{fp} \neq \emptyset$)

$t_i \in T_{fp}, t_i \in T_k$

IF (No se cumple el teorema 6.1.1) *Then* (" El producto k no se puede seguir produciendo.")

Calcular T_{bi}.

$T_{fp} := T_{fp} - T_{bi} - \{t_i\}$.

No tomar en cuenta en el cálculo de e los estados $M(l_i)$ donde $l_i \in T_{bi}^{\bullet}$

Eliminar $t_i \cup T_{bi}$.

End (While).

Eliminar $Op_{fp} \cup Op_b$

Else

buscar, si existe ρ.

IF (existe ρ) *Then* (el marcado del modelo donde $e_i \neq 0$,se iguala al marcado del sistema).

Else (El nuevo marcado del modelo y del sistema dependen del caso que se presente).

Activar rc_i.

End.(General).

Ejemplo: Se tiene el marcado del modelo es $[7,1,1,1,1,1,1,0,1,1,0,2,1,1,0,0,4]^T$ y el marcado del sistema es $[\ 7,1,1,1,1,1,1,0,1,0,1,2,1,1,0,0,4]^T$ El modelo esta representado por la red

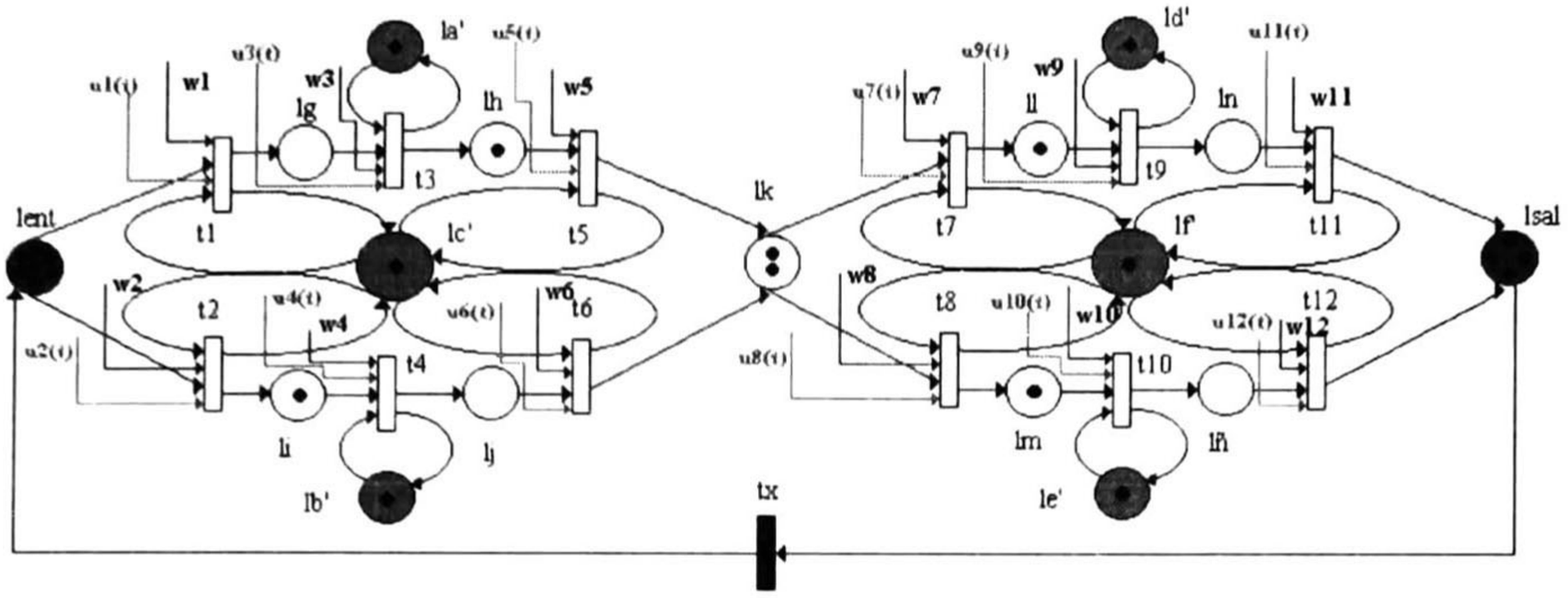


Figura 6.6:

de Petri SPSRTF de la figura 6.6. El marcado del sistema pertenece al conjunto de estados alcanzables por lo tanto no existe la secuencia de recuperación.

Se termina de disparar la transición t_4 por lo tanto se calcula e_i=[0,0,0,0,0,0,0,0,0,0,-1,1,0,0,0,0,0,0] el sistema de diagnóstico al extrar los recursos con posibles fallas tenemos que:

$$\begin{aligned} L_r &= \{\emptyset\} \\ L_h &= \{l_i, l_j\} \\ T_D &= \{t_4\} \\ L_p &= \{l_b^{'}\} \end{aligned}$$

El recurso al que se le van a aplicar las pruebas de diagnóstico es la rectificadora "f".

Al aplicar la librería de pruebas de rectificadora "f" los resultados del diagnóstico indican una falla permanente.

El algoritmo de recuperación trabaja de la siguiente manera:

$R_{pf} = \{$ la rectificadora "f"$\}$

$T_{fp} = \{t_4\}$

$t_4 \in T_{fp}, t_4 \in T_1$

$T_{bi} = \{t_2\}$

$T_{fp} := \{t_4\} - \{t_2\} - \{t_4\}.$

$T_{fp} := \{\emptyset\}$

No tomar en cuenta en el cálculo de e el estado l_i.

Eliminar $t_4 \cup t_2$

Eliminar $Op_4 \cup Op_2$

si, existe la secuencia ρ y es t_6, t_7, t_9, t_{11} o t_6, t_8, t_{10}, t_{12}

(el marcado del modelo donde $e_i \neq 0$,se iguala al marcado del sistema).

Activar rc_i.

6.2 Conclusiones.

En este capítulo se presentó un método para determinar cuando un SMF puede recuperarse del error. Claramente la recuperación depende del tipo de sistema y dónde ocurrió el error. Adicionalmente se presentó un algoritmo para evitar bloqueos en un SMF con fallas. Por último se determinó cómo calcular la secuencia de recuperación. Un algoritmo, al final del capítulo, permite establecer las recuperación automáticamente.

Conclusiones.

El trabajo desarrollado en la presente memoria se centra en cómo las redes de Petri ayudan a construir un STF para sistemas de manufactura.

El estudio de las propiedades de las RdP y el uso del esquema de control jerárquico presentado en la figura 1.5, las hipótesis de que los controladores no fallan, hacen que el sistema sea de fácil diseño y construcción, ya que sólo deben contemplar los estados válidos del sistema. El planteamiento del problema es el clásico que divide el STF en detección del error, diagnóstico y recuperación.

Se presenta una introducción al modelado de los SMF. Para el modelado se presenta un tipo de red de Petri propuesto por Ezpeleta y anteriormente por De Cindio porque han demostrado ser adecuadas para representar las operaciones y asignación de recursos de SMF. De los STF se mencionan sus principales características y cómo han sido tratados.

En la aproximación que se propone en esta tesis, la detección se hace en base al modelo realizado en RdP, para esto se le agrega a las RdP cuatro funciones extras, una que comunica al SMF con el modelo, otra que comunica al controlador con el modelo y dos para tomar en cuenta los tiempos mínimos y máximos para realizar una operación. Con esto se logra tener un modelo bastante bueno y manejable que captura la información deseada (información acerca de evitación de cualisiones o niveles de inventarios debe ser mantenida por el control supervisor). El estado del modelo y el estado del SMF son restados, si la resta es diferente del vector cero, entonces existe un error en el SMF

A continuación el mismo modelo en RdP sirve para localizar el conjunto de recursos que debe ser analizados en caso de un error. Esto es una ventaja sobre otros métodos que utilizan sistemas expertos para localizar este conjunto (en este trabajo se utilizan propiedades estructurales de las RdP). Para diagnosticar qué elemento está fallando del conjunto previamente establecido, se utiliza una librería de pruebas y un sistema experto. Dependiendo de la historia de la falla y la falla encontrada, se determina cuál es la nueva falla.

Por último, en el capítulo seis se hace un análisis exhaustivo de los tipos de falla para establecer las posibilidades de que un sistema degradado pueda producir la misma familia de productos. El lema 6.1.1 indica que para saber si un sistema degrado pueda seguir produciendo la misma familia de productos, se deben estudiar todas las posibles secuencias de procesado, asignación de recursos y recursos que presentan fallas. Esta tarea resulta NP-completa. Sin embargo, con el teorema 6.1.1 basta observar los pares selección-atribución (aquí propuestos), la asignación de recursos y los recursos con fallas, lo cual facilita el problema.

También se establece cómo degradar mínimamente el SMF en caso de que sea necesario (eliminando sólo los componentes dañados y los que nos pueden llevar a un estado de bloqueo), para crear el algoritmo nos basamos en los conceptos de evitación bloqueos de Ezpeleta.

Finalmente la secuencia de recuperación del error se lleva acabo encontrando la secuencia de que lleva el SMF de un estado de error a un estado válido. Aquí se encuentra un vector de disparos utilizando un problema de programación lineal, utilizando las ideas propuestas por Colom.

Como trabajo futuro se propone precisar más cada una de las ideas, ya que debido a la amplitud del trabajo y por ser el primero en esta área, se prefirió tener una panorama general a un buen nivel de conocimientos que profundizar demasiado en un sólo tema. Además los algorítmos propuestos deben ser puestos en práctica. Actualmente la arquitectura propuesta en la figura 1.5 ha sido probada por Rosa María Córdoba uniendo las partes de control supervisor y control óptimo en un prototipo hecho en Kappa, falta agregarle el esquema de tolerancia a falla.

Bibliografía

[1] T. Anderson and P.A. Lee. Fault tolerance principles and practice. Englewood Cliffs NJ: Prentice-Hall International. 1981.

[2] A. Desrochers and R. Al-Jaar. Applications of Petri nets in manufacturing systems IEEE Press, 1995.

[3] S. Pudalkar, G. Karsai, C. Biegl, and Janos Sztipanovits K. Okuda and N. Miyasaka Osaka gas company. Real-time fault diagnostics. Systems Expert IEEE, 1991.

[4] J.Ezpeleta. Análisis y síntesis de modelos libres de bloqueos para sistemas Concurrentes Tesis Doctoral, Universidad de Zaragoza, María de Luna 3 E-50015 Zaragoza, España, Junio 1993.

[5] M. Silva. Las redes de Petri: en la automática y la informática AC, Madrid, España 1985.

[6] J. Magott New NP-Complete problems in performance evaluation of concurrent systems using Petri Nets. IEEE Transactions on Software Engineering, May 1987.

[7] K.P. Valvanis. On the hierarchical modeling, analysis and simulation of flexible manufacturing systems with extended Petri nets. IEEE Transactions on systems, man and cybernetics, vol 20 n.1 January/February 1990 pp 94-100.

[8] G. Sánchez O. Scheduling en sistemas de manufactura flexible. Tesis de maestría. Departamento de ingeniería eléctrica. Sección control automático. Centro de Investigación y de Estudios Avanzados del I.P.N., México D.F, 1997.

[9] J.F. Sánchez. Simulador de objetos heterogéneos(orientado a sistemas de manufactura). Tesis de maestría. Instituto Nacional de Astrofisica, Optica y Electrónica 1997.

[10] M. Chu Zhou and F. Dicesare adaptive design of Petri net controllers for error recovery in automated manufacturing systems. IEEE Transactions on systems, man and cybernetics vol. 19 No.5 September/October 1989.

[11] V.S. Srinivasan and M. A. s Jafari. Fault detection / monitoring using time Petri Nets. IEEE Transactions on systems, man, and cybernetics, pp. 1115-1162 vol. 23, No. 4, July / August 1993.

[12] R. Valette and J.Cardoso,D. Dubois. Monitoring manufacturing systems by means of Petri nets with imprecise markings. IEEE Transactions on systems, man, and cybernetics, pp. 233-238 vol. 22, No.5, September 1989.

[13] A. Burns and A. Wellings. Real systems and their programming languages. Addison-Wesley publishing company 1990.

[14] B.Randell. Reliable computer systems. In opering systems, and advanced course. Bayer R., Graham R.M. and Seegmuller G., eds. pp.282-391. Berlin: Springer-Verlag 1978.

[15] Marco A. López Enríquez. Modelado de un sistema de manufactura flexible. Tesis de maestría. Departamento de ingeniería eléctrica. Sección control automático. Centro de Investigación y de Estudios Avanzados del I.P.N., México D.F.,1997.

[16] A. Ramírez. Scheduling en Redes de Petri. Tesis doctoral, Universidad de Zaragoza, María de Luna 3E-50015 Zaragoza, España, 1993.

[17] A. Ramírez. Modelado de tareas y especificación de controladores en céldas robotizadas. Tesis de maestría. Departamento de Ingeniería Electrica. Sección control automático. Centro de Investigación y de Estudios Avanzados del I.P.N., México D.F.,1990.

[18] C. Ramamoorthy y G. H. Performance evaluation of asynchonous concurrent systems using Petri nets. IEEE Transactions on Software Engineering, pp.234-249 1980.

[19] S. Leonhardt y M. Ayoubi. Methods of fault diagnosis. Control engineering practice, pp. 683-692 vol. 5 Number 5 May 1997 Pergamon press.

[20] R. Isermann. Supervision, fault-detection, and fault-diagnosis methods- and introduction. Control engineering practice, vol. 5 pp. 639-652 Number 5 May 1997 Pergamon press.

[21] R. Isermann y P. Ballé. Trends in the application of model-based fault detection and diagnosis of technical processes. Control engineering practice, vol. 5 pp. 709-719 Number 5 May 1997 Pergamon press.

[22] M. Blanke, R.Izadi-Zamanabadi, S.A.Bogh and C.P. Lunau. Fault- tolerant control systems. Control engineering practice, vol. 5 pp. 693-702 Number 5 May 1997 Pergamon press.

[23] Dhraj K. Pradhan. Fault-tolerant computer system design. Prentice Hall 1996.

[24] M. Ayoubi. Fussy systems design based on a hybrid neural structure and application to the fault diagnosis of technical processes. Control engineering practice, vol 4, N.1, pp.35-42 1996.

[25] R. Isermann. Process fault detection based on modeling and estimation methods. Automatica 20, pp.387-404 1984.

[26] A. Ye. Shumsky. Failure detection filter for diagnosis of nonlinear dynamic systems. In: Proc. IFAC Symp. SAFEPROCESS 94, Espoo Finland pp. 335-340 1994.

[27] R. Valette. "Control of felxible production systems and Petri nets " 3th European workshop on applications and theory of Petri nets. Varena, september 1982.

[28] J. Pujol Mesalles. Aplicación de SBC para diagnosis. Curso de Doctorado.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD GUADALAJARA

El Jurado designado por el Laboratorio de Ingeniería Eléctrica y Ciencias de la Computación del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: "Sistemas Tolerantes a Fallas en Sistemas de Manufactura Flexible", el día 3 de abril de 1998.

Dr. Arturo del Sagrado Corazón Sánchez Carmona

Dr. José Javier Ruiz León

Dr. Deni Librado Torres Román

Dr. Antonio Ramírez Treviño

M.en C. Alejandro Malo Tamayo

PROL. LOPEZ MATEOS SUR 590 APARTADO POSTAL 31-438 GUADALAJARA, JAL. 45090 TEL. (3) 684-1580 FAX (3) 684-1708