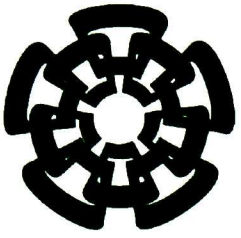
BC-655
Don 2011

xx (179128.1)



Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Unidad Guadalajara

Simulación de sistemas de eventos discretos temporizados modelados mediante redes de Petri con cronómetros

Tesis que presenta:
Janeth Gabriela Rivera Aguilar

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis
Dr. Luis Ernesto López Mellado

CINVESTAV del IPN Unidad Guadalajara, Guadalajara, Jalisco, Febrero de 2011.

Simulación de sistemas de eventos discretos temporizados modelados mediante redes de Petri con cronómetros

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Janeth Gabriela Rivera Aguilar
Ingeniero en Computación
Universidad de Guadalajara 2000-2004

Becario de Conacyt, expediente no. 219313

Director de Tesis
Dr. Luis Ernesto López Mellado

CINVESTAV del IPN Unidad Guadalajara, Febrero de 2011

Dedico esta tesis

A mis padres por su amor y apoyo incondicional, a mis hermanos Carlos, Bety, Laura, Lucia, Angeles y Guillermo por estar conmigo, ayudarme y alentarme, y a mi sobrina Adry por alegrarme con su sonrisa

Agradecimientos

Al CONACYT por otorgarme la beca que me permitió estudiar la maestría

A Sergio por caminar a mi lado durante esta etapa

A Corina la mejor amiga, que me enseño a sonreír aun en los momentos difíciles.

A Miriam la amiga más infantil, por sus observaciones sobre mi trabajo y por recordarme que no hay edad para hacer travesuras

A Martha la amiga más sensata, por ser quien mantuvo la cordura en esta generación

A Erick el amigo más inteligente, por su disposición para explicar y resolver mis dudas

A Molus el amigo más extrovertido, por las largas horas de conversación y todos sus consejos

A Octavio el amigo más dedicado, por su paciencia para explicar y escuchar

A Dan el mejor amigo, por mantener la calma y tranquilizarme en momentos de estrés

A Guadalupe por romper la rutina agregando horas de baile

A Oscar, Elias, Gustavo y Paquito por todos sus consejos en la redacción de la tesis

A todos los compañeros y amigos que siempre tuvieron palabras de aliento para mi

A mi asesor por su dedicación, apoyo y sobre todo paciencia

A los doctores Felix, Raúl y Siller por sus consejos y disposición para transmitir sus conocimientos

Simulación de sistemas de eventos discretos temporizados modelados mediante redes de Petri con cronómetros

Resumen

Esta tesis trata sobre la simulación de sistemas de eventos discretos temporizados, en particular aquellos que incluyen tareas interrumpibles. Se presenta un simulador de modelos expresados con redes de Petri con cronómetros globales (RPCG). Para este fin se propone un algoritmo de ejecución de modelos expresados mediante RPCG. El simulador es desarrollado en base a este algoritmo y es incorporado a una herramienta de edición y análisis de redes de Petri. Los resultados de una simulación pueden ser visualizados a través de gráficas que muestran la evolución de los cronómetros.

Simulation of timed discrete event systems modeled with stopwatches Petri nets

Abstract

This thesis deals with simulation of timed discrete event systems, in particular those including interruptible tasks. A simulator of global stop watch Petri nets (GSWPN) models is presented. For this purpose, an algorithm for the execution of GSWPN models is proposed. The simulator is developed based on this algorithm and it is embedded within a Petri net edition and analysis tool. Simulation results can be displayed through graphics that show the evolution of stopwatches.

Índice general

Introducción

Contexto

El análisis de sistemas de eventos discretos (SED) temporizados ha sido abordado mediante la simulación de modelos cuando la complejidad de éstos no permite obtener soluciones analíticas; en el caso de modelos expresados con redes de Petri (RP) temporizadas, este tipo de métodos están orientados a subclases de RP y los resultados son en muchos casos aproximaciones.

La simulación del sistema a través de la ejecución de su modelo en RP temporizadas permite el análisis del comportamiento transitorio y en estado permanente de sistema; en análisis del tipo "qué pasa si" la prueba de escenarios manipulando marcados iniciales y temporizaciones puede ser llevada a cabo de manera interactiva fácilmente [2].

Un cierto tipo de SED donde las tareas son interrumpidas y luego reanudadas no puede ser modelados con RP temporizadas. Para ello se han propuesto las RP con cronómetros (StopWatch Petri Nets) o relojes manipulables, donde los relojes se pueden detener, reanudar y reiniciar a cero.

Trabajos relacionados

Existen varios trabajos donde se aborda el estudio de las redes de Petri con cronómetros. Entre ellos tenemos el tabajo de Lime y Roux [3] que proponen una extensión de las redes de Petri temporizadas llamada Schedulling Extended Time Petri Nets que consiste en una calendarización de tareas que pueden ser suspendidas y activadas basandose en la utilización de recursos y pólizas de prioridad.

En el 2006 Magnin y Roux presentan un método eficiente para computar el espacio de estados de redes con cronómetros en [14] [15], este método funciona de manera eficiente y exacta basado en el uso de DBM (Difference Bounds Matrices).

En otro de sus trabajos [18] abordan el problema haciendo la conversión de una red de Petri con cronómetros (SWPN) a una RP clásica bisimilar con hiperarcos inhibidores y arcos

de flujo. De manera que el cómputo del espacio de estados en tiempo discreto es computado de forma directa utilizando herramientas para redes de Petri clásicas.

Otro trabajo en el que también se aborda el problema del análisis del espacio de estados de las RP con cronómetros es el de Allahham y Alla [7] quienes proponen un método de análisis basado en la conversión de una RP con cronómetros a un autómata con cronómetros equivalente. Además en este mismo trabajo se introduce la pre y post inicialización de los cronómetros.

Algunos otros trabajos se refieren a la utilización de alguna herramienta que facilite y/o automatice el análisis de éste tipo de modelos. Entre ellos está Romeo [12] que permite el cómputo de estados de RP temporizadas haciendo una conversión a autómatas temporizados.

Otra herramienta es el PHAVer (Polyhedral Hybrid Automaton Verifier) [8] que permite la verificación de las propiedades de la red y también permite computar el espacio de estados de un modelo, esta basado en un algoritmo de alcanzabilidad que computa aproximaciones para un autómata híbrido.

En la actualidad no existen herramientas que permitan simular las RP con cronómetros de forma directa, sin hacer una conversión a un modelo equivalente. Una herramienta que permita la simulación facilitaría el análisis y le proporcionaría información a diferentes grados de detalle al usuario.

El trabajo realizado

El objetivo de esta tesis ha sido la realización de un simulador que ejecute modelos expresados en RP con cronómetros, en particular una extensión llamada RP con cronómetros globales (RPCG), propuesta en [13] donde los cronómetros son manipulados en cualquier lugar del modelo. Para este fin se propone un algoritmo de ejecución de modelos expresados mediante RPCG el cual permite el máximo paralelismo en el disparo de las transiciones, garantizando que los resultados de la simulación cumplan con las restricciones temporales y que su ejecución se pueda realizar sobre una arquitectura multiplataforma.

Este algoritmo procesa de manera directa la red con cronómetros, es decir, no hace conversiones a otro modelo equivalente, como resultado de la simulación se proporciona una secuencia de disparos que cumpla con las restricciones de tiempo y las condiciones iniciales.

La funcionalidad del algoritmo se basa en la manera en que son elegidas las transiciones que serán disparadas; hay dos formas de elegir dichas transiciones: al primer instante en el que son habilitadas o en el último instante en el que están habilitadas.

El simulador es desarrollado en base a este algoritmo y se incorpora a una herramienta de edición y análisis de redes de Petri llamada SPADES. Los resultados de una simulación

pueden ser visualizados a través de gráficas que muestran la evolución de los cronómetros. El simulador se probó con diferentes casos de estudio sobre diferentes plataformas.

Estructura de la tesis

La tesis está organizada como sigue. En el capítulo 1 se presentan algunos conceptos generales sobre redes de Petri temporizadas y se presenta la definición de las RPCG. En el capítulo 2 se presenta el algoritmo de simulación para RPCG. En el capítulo 3 se presenta el proceso de Ingeniería de software que se siguió durante el desarrollo del simulador. En el capítulo 4 se muestra la herramienta que se obtuvo, se presentan algunos casos de estudio y se detalla cada uno de los módulos de la herramienta. Finalmente, en las conclusiones se mencionan las principales características del algoritmo propuesto y su implementación, así como trabajo futuro.

Capítulo 1

Análisis temporizado de sistemas

Resumen

En este capítulo se presentan los conceptos básicos de las redes de Petri, las redes de Petri temporizadas como una extensión de las PN y las redes de Petri con cronómetros como una extensión de las TPN. Finalmente se presenta una revisión de la literatura sobre herramientas y algoritmos para la ejecución de odelos en redes de Petri con cronómetros.

1.1. Redes de Petri

Una red de Petri es un grafo bipartita que se compone de dos tipos de nodos: lugares (que se representan como círculos) y transiciones (que se representan como rectángulos); éstos nodos se enlazan a través de arcos (que se representan como líneas dirigidas), un arco solo puede unir dos nodos si son de diferentes tipos [1]. La Figura 1.1 muestra una red de Petri.

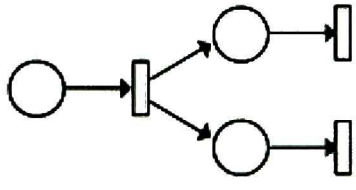


Figura 1.1: Red de Petri

Definición 1.1 (Definición formal de PN). Una red de Petri es una cuádrupla $G = (P, T, Pre, Post)$ $P = \{p_1, p_2, p_3, \ldots p_n\}$ *es un conjunto no vacío de lugares,* $T = \{t_1, t_2, t_3, \ldots t_m\}$ *es un conjunto no vacío de transiciones y* $P \cap T = \emptyset$, $Pre : P \times T \rightarrow \mathbb{N}$ *es la función de incidencia previa,* $Post : P \times T \rightarrow \mathbb{N}$ *es la función de incidencia posterior,* $Pre(p_i, t_j) > 0$ *indica el peso del arco que va del lugar* p_i *a la transición* t_j, $Post(p_i, t_j) > 0$ *indica el peso del arco que va de la transición* t_j *al lugar* p_i.

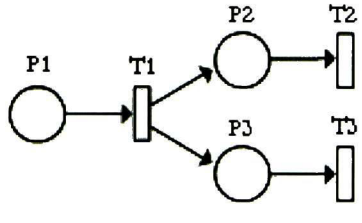


Figura 1.2: Red de Petri

Definición 1.2 (Marcado). Cada lugar puede contener una cantidad entera positiva de marcas (que se representan como puntos dentro del lugar). Sea $M : P \rightarrow \mathbb{N} \cup 0$ la función de marcado donde $m(p_i)$ es la cantidad de marcas que tiene el lugar p_i. El marcado de una red de Petri puede representarse por un vector A.

Definición 1.3 (Red de Petri marcada). Una red de Petri marcada es una pareja $R = (G, M_o)$ donde G es la estructura de una red de Petri y M_o es un vector de marcado llamado marcado inicial.

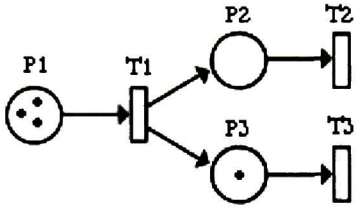


Figura 1.3: Red de Petri marcada

Definición 1.4 (Transición Habilitada). Se dice que la transición $t_j \in T$ está habilitada por el marcado M si para cada lugar de entrada se cumple:

$$\forall p_i \in P, \quad M(p_i) \geq Pre(p_i, t_j)$$

1.1.1. Evolución de marcado

Una red de Petri marcada cambia su marcado después de disparar las transiciones habilitadas. Su nuevo marcado se puede calcular con la siguiente ecuación de estado:

$$M_k = M_{k+1} + Av$$

Donde: A es la matriz de incidencia, v es el vector de disparo que indica la cantidad de veces que se dispara cada una de las transiciones habilitadas, M_{k-1} es el marcado anterior de la red, M_k es el nuevo marcado alcanzado después de disparar las transiciones habilitadas.

Definición 1.5 (Redes de Petri a salvo). Son redes donde $\forall p_i \in P$ el marcado que puede tener cada lugar es a lo mas 1, es decir, $M(p_i) \leq 1$.

1.2. Redes de Petri temporizadas

Las redes de Petri temporizadas (TPN por sus siglas en inglés) son un extensión de las RP que permiten modelar sistemas con especificaciones temporales, es decir, sistemas donde el tiempo desempeña un papel importante en la evolución de la red.

Definición 1.6 (Definición formal de TPN). Es una pareja $< R, Z >$ donde R es la red y Z una función que establece el tiempo de habilitación que se representa como un número real positivo que es asignado a los nodos de R.

Se han estudiado dos variantes de las TPN, las TPPNs donde el tiempo está asociado a los lugares de la red y las TTPNs donde el tiempo se asocia a las transiciones de la red. En [2] se presenta un análisis por simulación para las TPN.

1.2.1. TPPNs Redes temporizadas en lugares

En este tipo de redes se le asocia una cantidad de tiempo a cada lugar, de manera que cada marca del lugar p_j se vuelve disponible solo hasta que ha transcurrido el tiempo asociado a este lugar, a partir del instante en que la marca llega al lugar. Por consiguiente las transiciones pueden ser habilitadas para su disparo solo por marcas disponibles.

1.2.2. TTPNs Redes temporizadas en transiciones

En este tipo de redes se asocia una cantidad de tiempo a las transiciones, de manera que la transición t_j se pueden disparar solo si está habilitada por marcado, es decir, se cumple la función de incidencia previa y además ha transcurrido la cuota de tiempo asociado a la transición a partir del instante en que la transición fue habilitada por marcado.

1.2.3. Características de las TPNs

Las redes temporizadas permiten modelar la dinámica que tienen las tareas de un sistema, donde a cada tarea se le asigna un tiempo de duración. Una de las limitantes que tienen las redes temporizadas es que no permiten modelar la suspensión de las tareas que están en ejecución y después de un lapso de tiempo ser reanudadas siempre y cuando no hayan completado su tiempo de ejecución.

1.3. Redes de Petri con Cronómetros

Las redes de Petri con cronómetros (Stopwatch Petri Nets) son una extensión de las TPN. Se caracterizan porque permiten modelar especificaciones temporales de diferentes tareas acotadas por intervalos de tiempo, dichas tareas pueden ser detenidas y reanudadas más adelante en el mismo punto donde fueron detenidas, manteniendo un control sobre su tiempo de ejecución.

Este tipo de redes permiten modelar aspectos importantes de los sistemas temporizados como: la sincronización, el paralelismo, la exclusión mutua y la calendarización de tareas. La evolución temporal de la red se realiza mediante la inclusión de cronómetros en la red de Petri. De manera que cada tarea tiene asignado un conjunto de cronómetros y los intervalos de tiempo válidos para que dicha tarea pueda ser ejecutada; así, la tarea solo puede ser ejecutada cuando los cronómetros que tiene relacionados marcan tiempos correspondientes a los intervalos establecidos para que se realice la tarea. Diferentes definiciones para el modelado de redes de Petri con cronómetros han sido propuestas, a continuación se presenta algunas de estas definiciones.

1.3.1. Scheduling TPNs

Este formailismo [3] se basa en la idea de calendarizar procesos, con esta finalidad en se define de que manera varios planificadores (schedulers) del sistema pueden activar o suspender una serie de tareas que se tienen por realizar. Para lograrlo se introducen dos conceptos: prioridades y recursos compartidos. El planificador de tareas utiliza estas dos políticas para determinar las secuencias de disparo, tomando en consideración las prioridades y recursos compartidos (procesadores) que han sido asignadas a los lugares, mientras que para las transiciones se definen intervalos de tiempo sobre los que está permitido ejecutar la tarea. En base a estos elementos se determina la secuencia en la que serán disparadas las transiciones de la red de Petri. Además en [4] se presenta un método exacto para el cómputo del espacio de estados de una Scheduling extended time Petri net (SETPN).

1.3.2. Preemptive-TPNs

En [5] se propone un modelo similar al anterior, donde se definen políticas de planificación para determinar cuales transiciones de las que están habilitadas serán disparadas; esta elección se hace en base a dos tipos de políticas: prioridades y recursos compartidos. Las políticas de planificación son asignadas a las transiciones.

1.3.3. IHTPN Inhibitor Hyperarc TPNs

Calcular el espacio de estados de una SWPN se vuelve un trabajo complejo aun cuando se trate de una SWPN acotada en tiempo discreto, ya que al intentar delimitar su espacio de estados se genera un poliedro que contiene el conjunto de todos los posibles estados alcanzables. Por ello en algunos trabajos se decidió abordar el problema haciendo una conversión de una red de Petri con cronómetros(SWPN) a un modelo equivalente, ya sea un red de Petri clásica, un autómata con cronómetros u otro modelo que simplifique el análisis. En [6] se presenta la definición de los hiperarcos inhibidores y se propone un método para hacer la conversión de una SWPN a una red de Petri clásica que incluye: arcos de flujo, hiperarcos y arcos clásicos; el tiempo de los cronómetros está asociado a las transiciones. Los hiperarcos inhibidores son utilizados para controlar las secuencias de disparo de las transiciones, y tanto los arcos clásicos como los hiperarcos inhibidores son utilizados para representar que los cronómetros se inician, se detienen o reanudan.

1.3.4. Post and Pre−initialized stopwatch Petri Nets

En [7] se definen dos tipos de posibles inicializaciones sobre los valores de los cronómetros, pre y post inicialización. La pre-inicialización permite inicializar el valor del cronómetro en cuanto está recién habilitada una transición y la post-inicialización: permite inicializar el valor del cronómetro justo después de que fue disparada la transición.
El análisis no está hecho directamente en una red de Petri con cronómetros sino que se basa en la conversión de la red de Petri con cronómetros a un autómata con cronómetros (SWA) y presenta un método que permite el análisis y cómputo del SWA resultante. Además incluye los resultados obtenidos durante las pruebas de simulación que fueron realizadas con la herramienta PHAver [8].

1.4. Redes de Petri con cronómetros globales RPCG

Son una extensión de las redes de Petri con cronómetros; en éstas se definen cronómetros globales, cuyo valor puede ser utilizado y modificado desde cualquier lugar o transición de la red. Inicialmente todos los cronómetros están detenidos y su estado solo cambia a activo si algún lugar marcado lo indica [9] [10]. A partir de este momento en este texto se les llamará **RPCG** a las redes de Petri con cronómetros globales.

Definición 1.7 (Definición formal). Una red de Petri con cronómetros globales es una tupla $(P, T, Pre(.), Post(.), Mo, X, \Phi, G, INI)$, *donde* :

- $P = \{p_1, p_2, p_3, \ldots p_n\}$ *conjunto no vacío de lugares*
- $T = \{t_1, t_2, t_3, \ldots t_n\}$ *conjunto no vacío de transiciones, tal que* $P \cap T = \emptyset$
- $Pre : P \times T \rightarrow \mathbb{N}$ *es la función que representa el peso de los arcos que van de los lugares hacia las transiciones*

  $Post : P \times T \rightarrow \mathbb{N}$ *es la función que representa el peso de los arcos que van de las transiciones hacia los lugares*
- $M_0 \in (\mathbb{Z}^+)^{|P|}$ *el marcado inicial de la red para cada lugar, donde* $M(P_i)$ *es el número de marcas que contiene inicialmente* p_i
- $X = \{x_i \mid i \in \mathbb{N}\}$ *el conjunto finito de cronómetros a los que se les atribuyen las siguientes funciones:*
  - *Un conjunto de derivadas* $\dot{X} = \{(\dot{x}_i = c) \mid c \in \{0, 1\}\}$ *donde 0 significa que el cronómetro esta detenido y 1 que el tiempo esta avanzando*
  - *Un conjunto de invariantes* $C(X) = \{(x_i \; op \; v_j) \mid op \in \{=, <, >, \leq, \geq\}, \; v_j \in \mathbb{R}^{\geq 0}\}$ *descritas como expresiones booleanas que representan las condiciones de tiempo en que las marcas se vuelven disponibles*

  $\Phi : P \rightarrow 2^{\dot{X} \cup C(X)}$ *es la función que asocia a los lugares, un conjunto de derivadas e invariantes (condiciones de tiempo)*
- $G : T \rightarrow ExpG(C(X))$ *como la función que asocia a las transiciones un conjunto de expresiones booleanas llamadas "guardas", que definen las condiciones de tiempo que deben ser cubiertas para disparar la transición*
- $INI : T \rightarrow ExpINI(X)$ *como la función que asocia a las transiciones un conjunto de expresiones de la forma* $(x_i := vj)$. *para reiniciar al cronómetro* x_i *con un valor entero positivo.*

La Figura 1.4 muestra un ejemplo de una red de Petri con cronómetros globales (RPCG), donde se pueden observar todos los elementos que permiten detener y reanudar el progreso de las tareas. Las las derivadas e invariantes están asignadas a los lugares, mientras que las inicializaciones y guardas están relacionadas con las transiciones.
Las derivadas representan si un cronómetro esta activo o detenido, sus valores se representan con (0 y 1) respectivamente.
Las invariantes son expresiones booleanas que representan una condición de tiempo para las marcas que contiene el lugar al que corresponden.
Las inicializaciones representan el nuevo valor, un número real que será asignado a un cronómetro.
Las guardas son expresiones booleanas que representan una condición de tiempo que indica cuando la transición se convertirá en disponible.

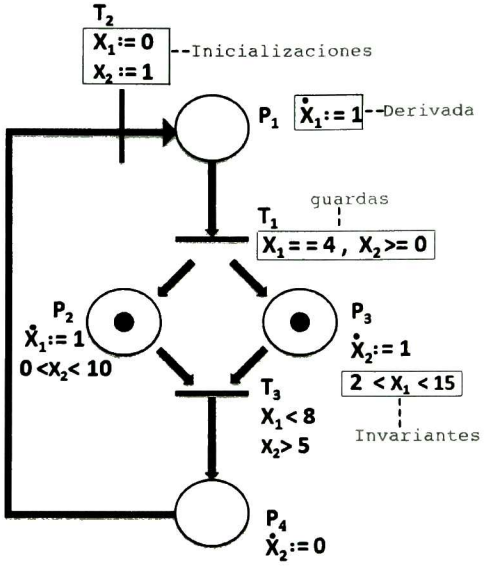


Figura 1.4: Red de Petri con cronómetros globales RPCG

- $P = \{P_1, P_2, P_3, P_4\}$
- $T = \{t_1, t_2, t_3\}$
- $M_0 = [0, 1, 1, 0]$
- $X = \{x_1, x_2\}$
- Φ
  - $\Phi(P_1) = \{(\dot{x_1} := 1)\}$
  - $\Phi(P_2) = \{(\dot{x_1} := 1), (x_2 > 0), (x_2 < 10)\}$
  - $\Phi(P_3) = \{(\dot{x_2} := 1), (x_1 > 2), (x_1 < 15)\}$
  - $\Phi(P_4) = \{(\dot{x_2} := 0)\}$

  G
  - $G(t_1) = \{(x_1 = 4), (x_2 \geq 0)\}$
  - $G(t_3) = \{(x_1 < 8), (x_2 > 5)\}$

  INI
  - $INI(t_2) = \{(x_1 := 0), (x_2 := 1)\}$

1.4.1. Habilitación de disparo de las transiciones

Sea Tm el conjunto de las transiciones habilitadas por marcado. Se dice que la transición $t_j \in Tm$ y está habilitada por el marcado $MsiM(p_i) \geq Pre(p_i, t_j) \forall p_i \in P$ y los invariantes $\Phi(p_i)$ son verdaderos.
Sea Tt el conjunto de las transiciones habilitadas por tiempo. Se dice que una transición $t_j \in Tt$ si todas las guardas $G(t_i)$ son verdaderas.
El conjunto de las transiciones disponibles para ser disparadas (Td) lo forman todas aquellas transiciones que están habilitadas por marcado y por tiempo: $Td = Tm \cap Tt$

1.4.2. Disparo de transiciones

Cuando una transición es disparada se modifica el marcado y además si la transición tiene asignados elementos de inicialización INI es necesario modificar los cronómetros.

1.4.3. Restricciones del modelo

Este formalismo es para redes de Petri a salvo o binarias. Dado que se utilizan cronómetros globales es importante verificar que no hay derivadas para un mismo cronómetro que se puede definir contradictoriamente. Esto sucede cuando dos o mas lugares haciendo referencia a un mismo cronómetro se marcan simultaneamente.

1.5. Simulación

La simulación es el proceso de elaborar un modelo matemático computarizado de un sistema o proceso y conducir experimentos con este modelo con el propósito de entender el comportamiento del sistema o evaluar varias estrategias con las cuales se puede operar el sistema [11].

La simulación es una técnica que se utiliza en el análisis de sistemas ya que permite experimentar con un modelo del sistema que se intenta estudiar. En muchas ocasiones la simulación puede ser la única posibilidad, debido a la dificultad para realizar experimentos y observar fenómenos en su entorno real, por ejemplo, un simulador de vuelo.
Durante la simulación generalmente conviene acelerar el tiempo que requieren para realizarse las tareas o procesos en ejecución, por consiguiente una simulación permite controlar el tiempo, debido a que un fenómeno se puede acelerar o retardar según convenga.

Un sistema de eventos discretos es un sistema que está determinado por una secuencia de eventos que ocurren en momentos aleatorios de tiempo $t_1, t_2 \ldots$ y el cambio de estado del sistema tiene lugar en esos instantes. En el análisis de este tipo de sistemas mediante simulación se siguen los cambios de estado consecuencia de una sucesión de eventos, y el avance del tiempo se hace hasta el evento siguiente más inminente.

Un simulador es un programa o software que reproduce el comportamiento de un modelo durante la ejecución de un programa. En general el simulador es un sistema que genera solo una trayectoria de estados para una condición inicial particular y un conjunto de entradas definidas.

1.6. Herramientas y Algoritmos de simulación para redes de Petri con cronómetros

1.6.1. Herramientas de software

Las herramientas de software utilizadas para el análisis y simulación de los modelos que se construyen, permiten observar de forma dinámica su comportamiento y evolución durante el tiempo de ejecución.
Algunas herramientas determinan el espacio de estados que genera el modelo simulado, y presentan algunas propiedades sobre el modelo. A continuación se presentan algunas herramientas de software que permiten hacer simulaciones de modelos con cronómetros.

Romeo

Este software tiene como propósito el análisis y simulación de sistemas reactivos modelados por TPNs o Schedulling-TPNs permitiendo una simulación gráfica. Está basado en métodos que permiten hacer una conversión de una TPN a un Autómata temporizado (Timed Automata TA) que preserve el comportamiento de la TPN. Soporta además el manejo de una extensión de las TPNs que incorpora la utilización de cronómetros (Scheduling-TPNs) representados en un Autómata con cronómetros (Stopwatch Autómata SWA). Las simulaciones se realizan a partir de los autómatas. La última versión soporta el uso de arcos inhibidores para el manejo de cronómetros [12].

PHAver model checking tool

Es una herramienta para verificar las propiedades de sistemas híbridos también llamados autómatas híbridos lineales. PHAver determina el espacio de estados de una computación del modelo, a partir de un autómata híbrido y su estado inicial [8].

TINA Time Petri Net Analyzer

Es una herramienta que permite el análisis de redes de Petri temporizadas TPNs y con cronómetros SwTPNs, la cual soporta el uso de arcos inhibidores e intervalos de tiempo abiertos. El software permite obtener un aproximación del espacio de estados cuando es posible utilizando aproximaciones geométricas [13].

1.6.2. Algoritmos para el análisis dinámico de redes de Petri con cronómetros

Algunos trabajos presentan métodos eficientes para calcular el espacio de estados de modelos con cronómetros; en éstos se definen claramente los pasos a seguir para obtener el cómputo de un modelo de entrada con restricciones de tiempo. Entre estos se encuentra un algoritmo que calcula de manera exacta el espacio de estados de un poliedro general, que bajo ciertas condiciones suficientes y necesarias, representa las restricciones de tiempo de la red de Petri con cronómetros, para después calcular una aproximación del espacio de estados que genera el poliedro al ser representado con matrices (DBM) [14] [15] [16]

También el problema de alcanzabilidad de estados ha sido estudiados en [17] donde se prueba que la alcanzabilidad de estados es indecidible para TPNs con cronómetros (SwTPNs), aun cuando éstas están acotadas; por lo tanto solo se pueden obtener aproximaciones sobre el espacio de estados y las condiciones suficientes para su verificación. Se presenta también un semi-algoritmo para el cálculo exacto del espacios de estado de (SwTPNs).

En [18] se presenta un algoritmo para analizar redes de Petri temporizadas con hiperarcos inhibidores. En este trabajo se presenta la semántica formal para un método que obtiene el espacio de estados de una red de Petri temporizada con hiperarcos inhibidores, donde los cronómetros se encuentran asociados a las transiciones, las cuales pueden ser detenidas y reanudadas en cualquier momento mediante el uso de arcos clásicos o hiperarcos inhibidores. Este método se basa en el cómputo exacto de un poliedro general que representa las restricciones de tiempo, para después calcular una aproximación del espacio de estados alcanzable cuando se representa al poliedro en una DBM (Difference Bounds Matrix).

Capítulo 2

Algoritmo de simulación de RPCG

Resumen

En este capítulo se describe el planteamiento del problema y el análisis de requerimientos para definir la funcionalidad y restricciones de operación del sistema, y las tareas que puede realizar el usuario.

Se presenta la especificación completa como una lista de requerimientos de usuario, sistema, funcionales y no funcionales.

En la última sección del capítulo se presenta en detalle el algoritmo propuesto para la ejecución de modelos RPCG.

2.1. Requerimientos

Los requerimientos de un sistema son la descripción de los servicios que este ofrece junto con las limitaciones operacionales a las que está sujeto. Los requerimientos se pueden clasificar como de usuario y de sistema, como se definen a continuación.

Los **requerimientos de usuario** describen los servicios que se espera que el sistema proporcione y las restricciones bajo las cuales debe funcionar.

Los **requerimientos de sistema** detallan las funciones, servicios y restricciones operativas del sistema; estos mismos a su vez se dividen en funcionales y no funcionales.

Los **requerimientos funcionales** describen a detalle las funciones, entradas, salidas y recursos del sistema.

Los **requerimientos no funcionales** definen las propiedades emergentes del sistema [19].

En esta tesis se propone un algoritmo para la ejecución de modelos expresados con **redes de Petri con cronómetros globales**. El algoritmo fue diseñado y desarrollado para satisfacer las restricciones de tiempo descritas en el modelo de cronómetros globales. Los datos de entrada son la estructura de una red de Petri (lugares y transiciones), el marcado inicial y los elementos que indican tanto variaciones como restricciones de tiempo, a los que a partir de este momento en este texto se llamarán **elementos temporizados que usan cronómetros** (**Derivadas e Invariantes** para los lugares y **Guardas e Inicializaciones** para las transiciones).

La ejecución del algoritmo encontrará una secuencia de estados válida para los datos de entrada y las condiciones iniciales proporcionadas. Finalmente como datos de salida se obtienen para cada cronómetro todos los valores que registro durante la simulación. La implementación del algoritmo es un módulo que se debe integrar al SPADES [20], una herramienta que permite el diseño y simulación de redes de Petri. A continuación se presentan los requerimientos de usuario y de sistema que fueron considerados durante el proceso de obtención de los requerimientos. Se utiliza el prefijo (U) para los requerimientos de usuario y el prefijo (S) para los requerimientos de sistema.

2.1.1. Requerimientos funcionales de usuario

Editar la red de Petri (U1).

(U1-1) El usuario puede dibujar una red de Petri o abrir un archivo donde esté almacenada, haciendo uso de la herramienta "SPADES" para después utilizar el simulador de

redes de Petri con cronómetros globales.

(U1-2) El usuario puede abrir un archivo donde tiene almacenada una red de Petri temporizada, haciendo uso de la interfaz del simulador de redes de Petri con cronómetros globales, pero en este caso no verá la red de Petri dibujada en el "SPADES"

(U1-3) El usuario puede agregar, modificar o eliminar de la red de Petri actual los elementos temporizados que usan cronómetros.

Almacenamiento de los elementos temporizados que usan cronómetros (U2).

(U2-1) El usuario puede guardar la red de Petri y elementos de la parte temporizada en un archivo de texto plano.

(U2-2) El usuario puede abrir un archivo previamente guardado con la configuración de una red de Petri temporizada haciendo uso de la interfaz que provee el simulador.

- ***Simulación (U3).***
  (U3-1) El usuario puede simular el modelo con el que trabaja actualmente o puede simular cualquier otro modelo que esté almacenado en un archivo.
  (U3-2) El usuario puede configurar los parámetros de la simulación que prefiere.

- ***Resultados de la simulación (U4).***

  (U4-1) El usuario puede observar los resultados al final de la simulación, en una tabla de resultados.

  (U4-2) El usuario puede almacenar los resultados obtenidos de la simulación en un archivo de texto plano.

- ***Graficador (U5).***

  (U5-1) El usuario puede generar gráficas de resultados de dos tipos: simples y dobles. Las gráficas simples incluyen los resultados de una simulación, mientras que las gráficas dobles incluyen los resultados de dos simulaciones.

  (U5-2) El usuario puede graficar los resultados de la simulación más reciente.

  (U5-3) El usuario puede graficar los resultados de una simulación almacenados en un archivo

  (U5-4) El usuario puede graficar una comparativa de los resultados almacenados en archivos para dos simulaciones.

2.1.2. Requerimientos funcionales de sistema

- ***Editar la red de Petri (S1).***

  (S1-1) El sistema debe ser capaz de tomar los datos de entrada del "SPADES" y procesarlos. Estos datos de entrada son: lugares, transiciones, matriz de incidencia, matrices de PRE y POST y el marcado inicial de la red.

  (S1-2) El sistema es capaz de leer los datos de entrada de un archivo de texto donde fueron previamente almacenados por el simulador y le permite al usuario editarlos.

  (S1-3) Se debe proveer una interfaz que facilite la tarea de editar: las derivadas, invariantes, inicializaciones y guardas relacionadas con una red de Petri.

- ***Almacenamiento de los elementos temporizados con cronómetros (S2).***

  (S2-1) El sistema debe proveer una interfaz que permita almacenar la red de Petri y los elementos temporizados con cronómetros.

  (S2-2) El sistema debe proveer una interfaz que permita abrir un archivo donde está almacenada la configuración de una red de Petri, además de los elementos temporizados con cronómetros.

- ***Algoritmo de Simulación (S3).***

  (S3-1) El algoritmo realizará la ejecución de un modelo RPCG de acuerdo a la semántica definida en el capítulo 1 y las condiciones iniciales configuradas por el usuario.

  (S3-2) El algoritmo debe incluir varias estrategias para determinar el conjunto de las transiciones habilitadas.

  (S3-3) El algoritmo debe incluir mecanismos que le permiten seleccionar las transiciones que se van a disparar de un conjunto de transiciones habilitadas que se encuentran en conflicto.

  (S3-4) El algoritmo debe incluir mecanismos para decidir cuando y cuanto se incrementa el reloj de simulación y todos los cronómetros de la red de Petri.

  (S3-5) Se debe incluir un control sintáctico que permita verificar que 2 o más cronómetros no se modifiquen en diferentes T-componentes, para mantener la coherencia entre las derivadas de los lugares.

Resultados de simulación (S4).

(S4-1) Al finalizar una simulación se mostrará como resultado todas las actualizaciones de tiempo que registró cada cronómetro durante la simulación.

(S4-2) El sistema dispondrá de un mecanismo para almacenar en un archivo los resultados de la simulación.

(S4-3) El algoritmo encuentra una secuencia de estados alcanzables a partir de la especificación inicial.

2.1.3. Requerimientos no funcionales de sistema:

- ***Tiempo de respuesta (S5).***

  (S5-1) Antes de iniciar la simulación se establece el tiempo total de simulación.

  (S5-2) En caso de que la simulación se vuelva infinita sin alcanzar el tiempo total de simulación, debe existir una forma de terminar la simulación.

  Portabilidad (S6).

  (S6-1) Se requiere de una herramienta portable, es decir, capaz de ejecutarse en diferentes plataformas.

  Flexibilidad (S7).

  (S7-1) El sistema se debe empaquetar como un componente para que sea fácil integrarlo a la herramienta "SPADES"

2.2. Algoritmo de simulación

2.2.1. Estrategia general

Se propone un algoritmo que permita simular la secuencia de disparos de una red de Petri temporizada con cronómetros globales partiendo de un marcado inicial. En cada instante de tiempo se disparan de manera máxima paralela todas las transiciones que cumplan sus condiciones de disparo. En caso de que el conjunto de las transiciones disponibles sea vacío, se incrementa el tiempo del reloj de simulación y de todos los cronómetros activos hasta el siguiente instante de tiempo donde ocurra algún evento.
Antes de iniciar la simulación se deben definir algunos criterios que pueden ser útiles para el análisis de resultados; estos criterios son la estrategia para elegir las transiciones disponibles y el tiempo total de simulación.

El algoritmo se ejecutará mientras haya al menos un lugar marcado y no se cumpla alguna de las condiciones para terminar la simulación. Los criterios de terminación que se definieron son los siguientes:

- Bloqueo de la red (deadlock) se detecta que no hay incremento de tiempo posible donde se den las condiciones de disparo de una transición.
- Cuando se alcanza el tiempo de simulación una vez que ha transcurrido el tiempo que se estableció para la simulación.

Durante toda la ejecución el algoritmo aplicará únicamente la estrategia de selección de transiciones disponibles que haya sido elegida antes de iniciar la simulación. El uso de estas estrategias permite que la secuencia de disparos que se alcance durante una simulación sea diferente a otra simulación, aun cuando se trate del mismo modelo. Las dos estrategias que se definieron son:

- *El primer instante.* Elige las transiciones inmediatamente cuando se vuelven disponibles.
- *El último instante.* Elige las transiciones en el último instante antes de que se vuelvan NO disponibles para todas las restricciones de tiempo con cota superior. Para el caso de las restricciones sin cota superior se pondrán en la lista de disponibles utilizando la estrategia del primer instante (debido a que no es posible determinar un último instante).

2.2.2. Definición de las estructuras de datos

Para agilizar el manejo de los datos se definieron una serie de estructuras que son utilizadas para clasificar los elementos (lugares y transiciones) de acuerdo con el tipo de características que cumplen. Se hace una breve descripción de las características que cumplen los elementos que son almacenados en cada una de las estructuras.

Evento Un evento es todo aquello que puede ocurrir durante la simulación que implica un cambio de estado. Los eventos pueden suceder de forma casi simultánea en cualquier orden.

En este trabajo se consideran cinco tipos de eventos:

Una marca se vuelve disponible.

Una marca deja de ser disponible.

- Una transición se vuelve disponible.

Una transición deja de ser disponible.

El disparo de una transición.

Simulación por eventos discretos La simulación por eventos discretos es un tipo de simulación dinámica donde el avance del reloj de simulación ocurre solo cuando se presenta un nuevo evento.

Reloj de simulación Es la variable de referencia global durante la evolución temporal del sistema, la cual registra el tiempo transcurrido desde que se inició y hasta que termina la

simulación. Permite conocer el instante en el que ocurre cada evento. La variable es modificada en instantes discretos, cada vez que se presenta alguno de los cinco eventos antes descritos.

Lugares marcados: Es la lista que contiene todos los lugares cuyo marcado es mayor que cero.

Lugares disponibles: Es la lista que contiene los lugares marcados que cumplen con todas sus restricciones de tiempo expresadas por el invariante.

Transiciones habilitadas: Es la lista que contiene las transiciones que están habilitadas únicamente por lugares disponibles.

Transiciones disponibles: Es la lista que contiene las transiciones habilitadas que cumplen todas sus restricciones de tiempo.

Transiciones disparables: Es la lista que contiene las transiciones disponibles libres de conflicto, agregando las transiciones correspondientes a la solución de los conflictos. Por lo tanto esta lista no contiene ninguna transición en conflicto.

Cronómetro: Es la variable que se utiliza para registrar el tiempo que transcurre. Un cronómetro puede tener dos estados: avanzando o detenido, pero no ambas a la vez.

Reloj de simulación: Es un cronómetro, que tiene la característica de que nunca es detenido, es decir, siempre está avanzando.

2.2.3. Descripción del algoritmo

En este apartado se describe de manera precisa la secuencia de pasos para lograr el algoritmo de simulación pueda encontrar una secuencia de disparos válida de acuerdo a las condiciones iniciales de la redes de Petri temporizadas con cronómetros globales.

Paso 1:- Se establecen los criterios de la simulación. Se define el valor para el tiempo total de la simulación y la estrategia del algoritmo.

Paso 2:- Inicialización de las variables que registran el tiempo. Se inicializan todas las variables. El reloj de simulación y los cronómetros serán inicalizados en cero y su estado será detenido. Además se obtiene el marcado inicial (m_0) y las matrices *pre* y *post* a través del SPADES.

Paso 3:- Generar la lista de lugares marcados (LLM). Localizar todos los lugares donde se cumple que $m(p_i) > 0$ y agregarlos a la lista de lugares marcados.

Paso 4:- Utilizar las derivadas correspondientes a cada uno de los lugares marcados. Por cada lugar que esté en la lista de **lugares marcados** utilizar todas sus derivadas, para actualizar el estado de los cronómetros.

Paso 5:- Generar la lista de lugares disponibles (LLD). Para todo lugar $p_i \in$

LLM si se cumplen todas las restricciones de tiempo (invariantes) o en su defecto el lugar no tiene establecidas restricciones de tiempo, entonces el lugar se agrega a la lista de lugares disponibles.

Paso 6:- Generar la lista de las transiciones habilitadas (LTH). Para toda transición $t_j \in T$, Si todos los lugares de entrada $\bullet t_j$ pertenecen **LLD**, entonces t_j se agrega a la lista **LTH**.

Paso 7:- Generar la lista de transiciones disponibles (LTD). Hay dos formas de generar esta lista, considerando que el algoritmo utiliza dos estrategias para seleccionar las transiciones disponibles.

`Primer instante`: *Para toda transición $t_j \in$ **LTH** si se cumplen todas sus restricciones de tiempo (guardas) o en su defecto la transición no tiene establecidas restricciones de tiempo, entonces la transición se agrega a la lista **LTD***

`Último instante`: *Para utilizar esta estrategia es necesario que antes de iniciar la simulación el usuario seleccione una transición $t_j \in T$ Ésta transición es la única que se disparará en el último instante; el resto de las transiciones se siguen disparando con la estrategia del primer instante. Para t_j si es disponible, primero se calcula la cantidad de tiempo que le falta a la transición t_j antes de dejar de ser disponible. Si el tiempo restante es mayor que cero o si la transición no tiene establecida una cota superior entonces se agrega la transición a la lista **LTD**.*

Paso 8:- Si LTD $= \emptyset$ el tamaño de la lista de transiciones disponibles es cero. Entonces **Buscar un incremento de tiempo** *(ir al paso 13).*
Sino *continuar.*

Paso 9:- Generar la lista de transiciones disparables (LTF). Para toda transición $t_j \in$ **transiciones disponibles** verificar si están en conflicto con alguna otra transición disponible, de ser así es necesario resolver el conflicto. Se agregan a LTF todas aquellas transiciones que se encuentran libres de conflicto.

Paso 10:- Disparar las transiciones. Para toda transición $t_j \in$ **LTF** aplicar la ecuación de estado y determinar el nuevo marcado de la red.

Paso 11:- Inicializar cronómetros. Para toda transición $t_j \in$ **LTF** inicializar los cronómetros relacionados con las transiciones disparadas que incluyan expresiones de post-inicialización.

Paso 12: Evaluar la condición de fin de la simulación. Si la red no tiene marcas, esta bloqueada o si el reloj de simulación registra un tiempo mayor o igual al tiempo total de simulación, entonces **terminar simulación.**
Sino entonces repetir algoritmo *(volver al paso 3)*

Paso 13:- Buscar un incremento de tiempo. Este incremento es el tiempo que

necesita transcurrir para que suceda el siguiente evento.
Si el incremento > 0 entonces continuar *(ir al paso 14).*
Sino entonces la red se ha bloqueado *(deadlock)* y la simulación se dará por terminada.

Paso 14:- Avanzar el tiempo. Sumar al reloj de simulación y a todos los cronómetros cuyo estado sea activo el incremento encontrado. *(ir al paso 12).*

Ejemplo

El siguiente ejemplo servirá para ilustrar de manera gráfica los pasos que sigue algoritmo. En la Figura 2.1 (a) se presenta la estructura de una red de Petri temporizada con cronómetros globales.

Una vez que se agrega el marcado inicial $m_0 = [1, 0, 0]$ como se muestra en la Figura 2.1 (b) entonces se inicia la simulación. En este ejemplo se utilizan marcas y transiciones blancas para representar lugares y transiciones que no cumplen sus restricciones de tiempo, mientras que las marcas y transiciones negras representan elementos disponibles.

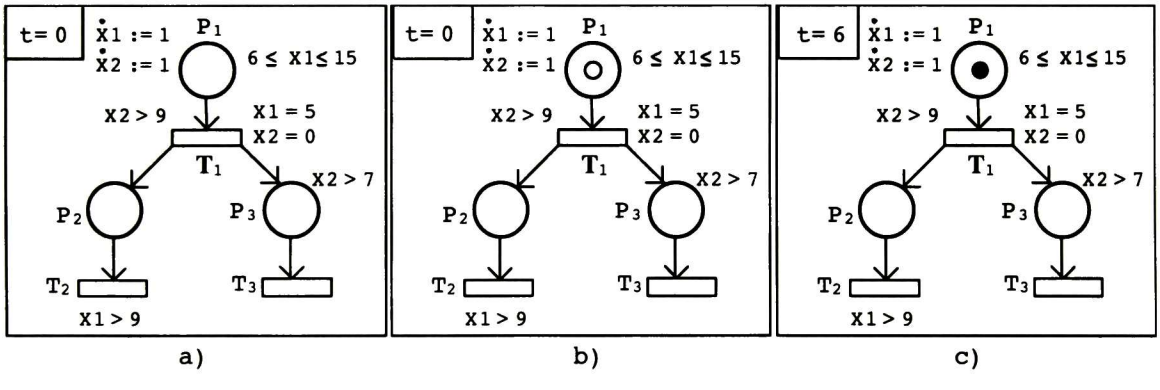


Figura 2.1: Ejemplo de ejecución de una RPCG

Se inicializan el tiempo de simulación y todos los cronómetros en cero t=0, $x_1 = 0$, $x_2 = 0$. Y entonces se crea la lista de los lugares marcados:

$LugaresMarcados = \{P_1\}$

Una vez que se conocen los lugares marcados se verifica si en el tiempo actual t=0, $x_1 = 0$ y $x_2 = 0$, se cumplen las restricciones de tiempo del lugar, en este caso $5 < x_1 < 15$. Por lo tanto:

$LugaresMarcados = \{P_1\}$

Como no hay lugares disponibles, entonces es necesario que transcurra el tiempo hasta $x_1 = 6$, como se muestra en la Figura 2.1 (c) para que se cumpla la restricción de tiempo de P_1.

$LugaresDisponibles = \{P_1\}$

Cuando $LugaresDisponibles \neq \emptyset$ entonces se puede crear la lista de las transiciones habilitadas.

$TransicionesHabilitadas = \{T_1\}$

Observe que T_1 está habilitada pero no disponible, ya que no cumple sus restricciones de tiempo, T_1: $x_2 > 9$

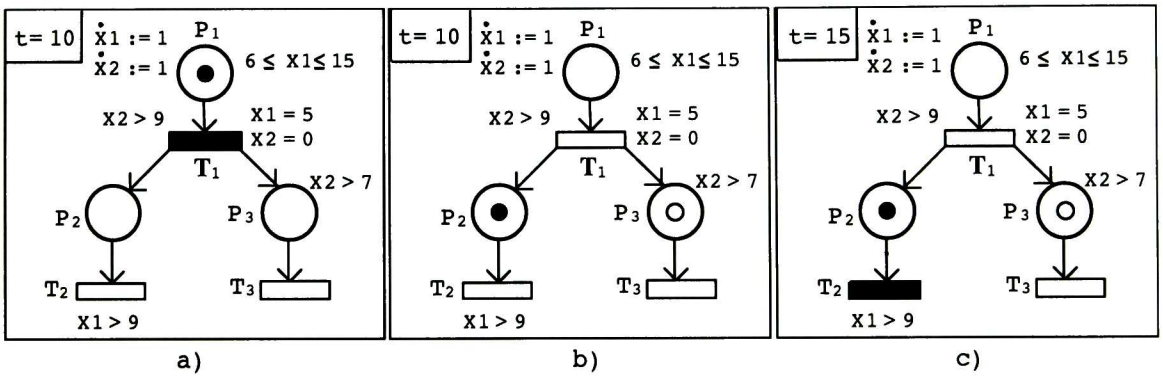


Figura 2.2: Ejemplo de ejecución de una RPCG

Para que T_1 se convierta en disponible es necesario que transcurra el tiempo hasta cuando $x_2 = 10$, el procedimiento **"calcular incremento"** debe proporcionar el valor 4, entonces todos los cronómetros activos incrementarán en 4 unidades su valor. Por lo tanto T_1 se vuelve disponible en el tiempo global *t=10* como se observa en la Figura 2.2 (a).

$TransicionesDisponibles = \{T_1\}$

Y dado que $\mid TransicionesDisponibles \mid = 1$ no hay conflicto que resolver, por lo tanto:

$TransicionesDisparables = \{T_1\}$

En el momento *t=10* se dispara la transición, se cambia el marcado y se inicializan los cronómetros $x_1 = 5$ y $x_2 = 0$

La Figura 2.2 (b) muestra el nuevo marcado, además puede observar que el lugar P_2 es disponible inmediatamente después del disparo, teniendo en cuenta que no tiene restricciones de tiempo, mientras que P_3 es un lugar que aun no es disponible.

$LugaresMarcados = \{P_2, P_3\}$

$LugaresDisponibles = \{P_2\}$

$TransicionesDisponibles = \emptyset$

Como en el tiempo *t=10* no hay transiciones disponibles, es necesario calcular un incremento de tiempo hasta $x_1 > 9$ o $x_2 > 7$ lo que suceda primero.

En el tiempo *t=15*, $x_1 = 10$ hace que se cumpla la restricción de tiempo y entonces se convierte en disponible T_2 como se muestra en la Figura 2.2 (c).

$TransicionesDisponibles = \{T_2\}$

T_2 no está en conflicto con ninguna otra transición, así que:

$TransicionesDisparables = \{T_2\}$

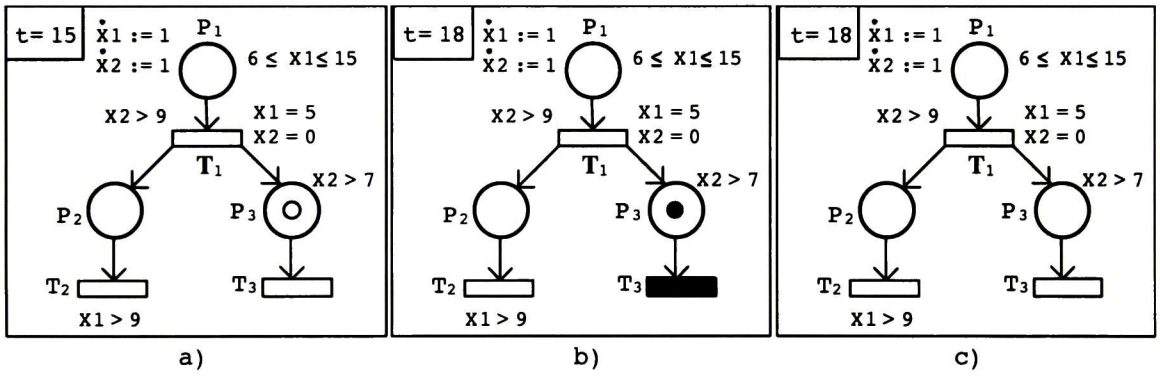


Figura 2.3: Ejemplo de ejecución de una RPCG

Después de disparar T_2 no hay inicializaciones por realizar, así que el nuevo marcado de la red es el que se muestra en la Figura 2.3 (a).

$LugaresMarcados = \{P_3\}$

Cuando *t=18* y $x_2 = 8$, se cumple la restricción de tiempo de P_3, y como se muestra en la Figura 2.3 (b) P_3 se vuelve disponible. Y de inmediato T_3 también se convierte en disponible, ya que no tiene restricciones de tiempo asignadas.

$LugaresDisponibles = \{P_3\}$

$TransicionesDisponibles = \{T_3\}$

En ese mismo instante se dispara T_3, y se modifica el marcado de la red, como se muestra en la Figura 2.3 (c). Después del disparo la red se queda sin marcado, por consiguiente en este instante se termina la simulación.

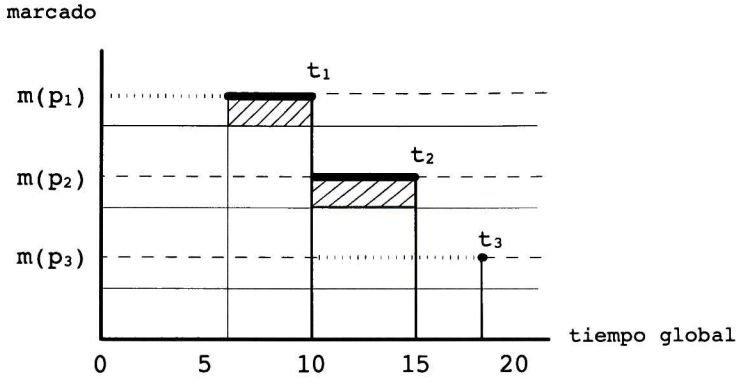


Figura 2.4: Diagrama de tiempo

El diagrama de la Figura 2.4 muestra la evolución del marcado con el transcurso del tiempo, donde las líneas discontinuas representan lugares sin marcas, las líneas punteadas representan lugares marcados pero cuyo marcado no está disponible y las líneas continuas representan marcados disponibles.

Como puede observar aunque el lugar p_1 está marcado desde $t = 0$, es hasta $t = 6$ que su marcado está disponible y se mantiene así hasta $t = 10$ cuando es utilizado para habilitar la transición t_1, la cual después de ser disparada deja al lugar p_1 sin marcas. Ahora observe que p_3 es marcado en $t = 10$ pero su marcado no está disponible sino hasta $t = 18$, cuando es utilizado para disparar la transición t_3. Por lo tanto en $t = 18$ el lugar p_3 ya no tiene marcas.

2.2.4. Algoritmo de simulación

A continuación se presenta en forma estructurada la estrategia anteriormente descrita. Se trata de un procedimiento principal y varios auxiliares que son llamados desde el primero.

Algoritmo 2.1 Ejecución de una RPCG

Entrada: *tiempoSimulacion*, *estrategiaDisparo*, m_o, *pre*, *post*, *derivadas*, *invariantes*, *inicializaciones*, *guardas*
Salida: *LTF*, *valor(reloj)*

1: *reloj* $\leftarrow 0$, *detener(reloj)*, *tam* $\leftarrow 0$, *deadlock* $\leftarrow$ *falso*
2: **para todo** $x_i \in X$ **hacer**
3: $x_i \leftarrow 0$, *detener*(x_i)
4: **fin para**
5: **repetir**
6: **para todo** $p_i \in P$ **hacer**
7: **si** $m(pi) > 0$ **entonces**
8: $p_i \in$ **LLM**
9: **fin si**
10: **fin para**
11: **para todo** $p_i \in$ **LLM** **hacer**
12: *Activar/detener cronómetros según* $DER(p_i)$
13: **fin para**
14: **Crear LLD**
15: **Crear LTH**
16: **Crear LTD**
17: *tam* $\leftarrow$ tamaño de LTD
18: **si** *tam* $= 0$ **entonces**
19: **Calcular incremento de tiempo**
20: **si** *incremento* > 0 **entonces**
21: reloj $\leftarrow$ reloj + incremento
22: **para todo** $x_i \in X$ **hacer**
23: **si** *activo*(x_i) **entonces**
24: $x_i \leftarrow x_i +$ *incremento*
25: **fin si**
26: **fin para**
27: **si no**
28: Deadlock
29: **fin si**
30: **si no**
31: **LTF** $\leftarrow$ LTD - conflictos
32: **para todo** $t_j \in$ **LTF** **hacer**
33: disparar t_j
34: **fin para**
35: Sea $INI(t_j)$ el conjunto de inicializaciones de t_j
36: **para todo** $t_j \in$ **LTF** **hacer**
37: **para todo** *valor*$(x_i) \in INI(t_j)$ **hacer**
38: **inicializar**
39: **fin para**
40: **fin para**
41: **fin si**
42: **hasta que** $(\exists p_i \mid m(p_i) > 0)$ **and** $(deadlock = falso)$ **and** $(reloj < tiempoSimulacion)$

Algoritmo 2.2 Crear LLD

Entrada: *LLM*
Salida: *LLD*

1: **para todo** $p_i \in LLM$ **hacer**
2: Sea $INV(p_i)$ la lista de invariantes de p_i.
3: $N \leftarrow$ tamaño de $INV(p_i)$
4: **si** N = 0 **entonces**
5: $p_i \in$ **LLD**
6: **si no**
7: **para todo** $inv \in INV(p_i)$ **hacer**
8: **si** $evaluacion(inv) =$ **cierto entonces**
9: $p_i \in$ **LLD**
10: **fin si**
11: **fin para**
12: **fin si**
13: **fin para**
14: **devolver LLD**

Algoritmo 2.3 Crear LTH

Entrada: *LLD*
Salida: *LTH*

1: **para todo** $t_j \in T$ **hacer**
2: **si** $\bullet t_j \in$ **LLD entonces**
3: $t_j \in$ **LTH**
4: **fin si**
5: **fin para**
6: **devolver LTH**

Algoritmo 2.4 Crear LTD

Entrada: LTH, *tiempoActual*, *estrategia*
Salida: LTD

```
1: para todo t_j ∈ LTH hacer
2:    Sea G(t_j) la lista de guardas de t_j
3:    N ← tamaño de G(t_j)
4:    si estrategia = 1 entonces
5:       Primer instante
6:       si N = 0 entonces
7:          t_j ∈ LTD
8:       si no
9:          para todo g ∈ G(t_j) hacer
10:            si evaluacion(g) = cierto entonces
11:               t_j ∈ LTD
12:            fin si
13:         fin para
14:      fin si
15:   fin si
16:   si estrategia = 2 entonces
17:      Último instante
18:      faltante ← cotaSuperior − tiempoActual
19:      si N = 0 entonces
20:         t_j ∈ LTD
21:      si no
22:         para todo g ∈ G(t_j) hacer
23:            si evaluacion(g) = cierto and (faltante = 0 or faltante = ∞) entonces
24:               t_j ∈ LTD
25:            fin si
26:         fin para
27:      fin si
28:   fin si
29: fin para
30: devolver LTD
```

Algoritmo 2.5 Calcular incremento de tiempo de simulación

Entrada:
Salida: *incremento*

1: **para todo** p_i **hacer**
2: **si** $p_i \in LLM$ y $p_i \notin LLD$ **entonces**
3: $p_i \in lugaresEnEspera$
4: **fin si**
5: **fin para**
6: **para todo** $t_j \in T$ **hacer**
7: **si** $\bullet t_j \in LLD$ **entonces**
8: $t_j \in transicionesEnEspera$
9: **fin si**
10: **fin para**
11: **para todo** $p_i \in lugaresEnEspera$ **hacer**
12: $N \leftarrow$ cantidad de invariantes de p_i
13: $faltante_{pi} \leftarrow MAX\{timeleft(inv_{1i}), timeleft(inv_{2i}), timeleft(inv_{3i}), \ldots, timeleft(inv_{Ni})\}$
14: **si** $faltante_{pi} \leq 0$ **entonces**
15: $lugaresEnEspera \leftarrow lugaresEnEspera - p_i$
16: **fin si**
17: **fin para**
18: **para todo** $p_i \in lugaresEnEspera$ **hacer**
19: $M \leftarrow$ cantidad de guardas de t_j
20: $faltante_{tj} \leftarrow MAX\{timeleft(gua_{1j}), timeleft(gua_{2j}), timeleft(gua_{3j}) \ldots, timeleft(gua_{Mj})\}$
21: **si** $faltante_{tj} \leq 0$ **entonces**
22: $transicionesEnEspera \leftarrow transicionesEnEspera - t_j$
23: **fin si**
24: **fin para**
25: $incremento \leftarrow MIN \left\{ \begin{matrix} faltante_{pi} & \forall p_i \in lugaresEnEspera \\ faltante_{tj} & \forall t_j \in transicionesEnEspera \end{matrix} \right\}$
26: **si** $incremento > 0$ **entonces**
27: **devolver** *incremento*
28: **si no**
29: $deadlock \leftarrow$ **falso**
30: **devolver** *deadlock*
31: **fin si**

Otros procedimientos auxiliares:

evaluacion (exp-bool): Proporciona el valor (V, F) resultados de la evaluación de una expresión booleana (exp-bool).

timeLeft (exp-bool): Proporciona la cantidad de tiempo que falta por transcurrir para que la expresión booleana (exp-bool) pueda ser evaluada como (V). En caso de que se trate de un retroceso en el tiempo, proporciona un número negativo.

2.2.5. Conclusiones

En este capítulo se describe el algoritmo de simulación propuesto, el cual define paso a paso la estrategia general a seguir para realizar la simulación acompañado de un ejemplo donde se ilustra. Además se describen los requerimientos del simulador de RPCG que implementará el algoritmo de simulación. Finalmente se presenta el algoritmo propuesto.

Capítulo 3

Desarrollo de un simulador de RPCG

Resumen

En este capítulo se presentan los aspectos más importantes del proceso de ingeniería de software, en particular los resultados de la etapa del diseño y planeación del simulador. La descripción incluye algunos diagramas UML: diagramas de datos y diagramas de flujos, los cuales muestran el flujo de tareas y la organización estructural del simulador.

3.1. Diseño

Durante la etapa del análisis de requerimientos se reunieron todos los requerimientos que se recogieron del cliente, éstos requerimientos fueron clasificados de acuerdo a su tipo y en base a ellos se comenzó el modelado del sistema.

Definición 3.1 (Modelo). Se define modelo como una representación abstracta de una especificación, un diseño o sistema desde un punto de vista en particular [21].

Los modelos son representaciones gráficas que describen los procesos, problemas a resolver y el sistema que tiene que ser desarrollado [19].

La etapa del diseño es el periodo donde se realizan modelos útiles que permiten visualizar desde varias perspectivas las características mas relevantes del sistema sin incluir los detalles de la implementación. Existen diferentes tipos de modelos de datos, pero el único que se menciona durante este texto es el modelado orientado a objetos, que fue el que se utilizó durante la etapa del diseño del simulador. Se eligió el modelado orientado a objetos para simplificar la transición del modelo a la implementación, con fines de poder incorporar el simulador a una herramienta para redes de Petri hecha sobre java.

Definición 3.2 (Modelado Orientado a Objetos). Es un método de implementación en la cual el principal bloque de construcción es el objeto o la clase. Los programas se organizan como cooperación de colecciones de objetos [21].

Definición 3.3 (Clase). Una clase es una descripción de un conjunto de objetos similares. Todo objeto tiene identidad(nombre), estado y comportamiento [21].

Los modelos de objetos permiten representar los datos del sistema así como su procesamiento a través de modelos de flujos de datos y modelos semánticos de datos.

Para obtener una vista completa del sistema se consideraron tres perspectivas: externa, de comportamiento y estructural. La perspectiva externa se utiliza para modelar el contexto o entorno del sistema. La perspectiva de comportamiento se utiliza para modelar el comportamiento del sistema. La perspectiva estructural permite modelar la estructura de los datos que procesa el sistema.

En este capítulo se presenta la especificación completa del simulador para redes de Petri con cronómetros globales, ilustrado gráficamente a través de un conjunto de diagramas UML que cubren las vistas estáticas y dinámicas del sistema, con el fin de presentar una visión general de su organización.

3.1.1. Arquitectura del Simulador de Análisis Temporizado

Es esencial comenzar por definir la perspectiva externa del simulador, es decir, el entorno donde se ubica el simulador, con cuales subsistemas se relaciona e intercambia información; además conviene mostrar su estructura interna, especificando los módulos tiene y cual es la función de cada uno. Se presenta a continuación un diagrama arquitectónico (Figura 3.1), donde se puede observar los bloques definidos para cubrir los requerimientos planteados inicialmente.

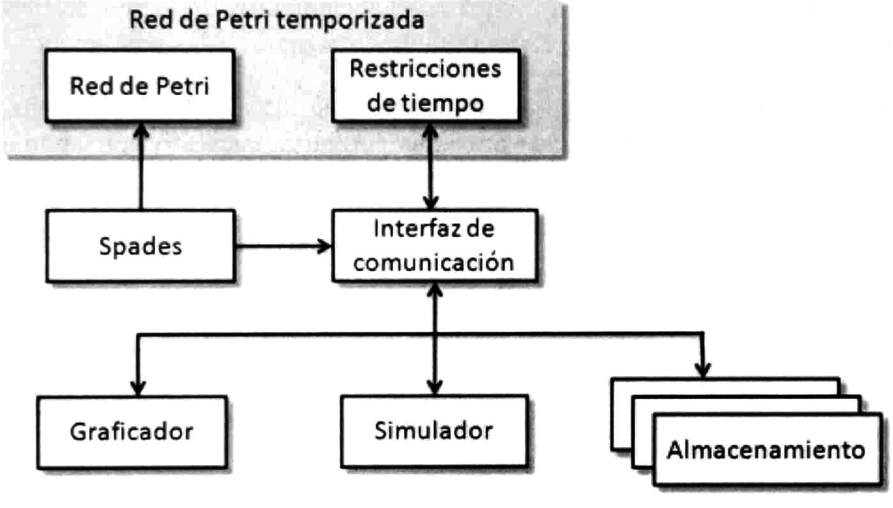


Figura 3.1: Diagrama arquitectónico

En el diagrama se puede observar una zona gris; ésta representa todos los elementos que corresponden a la red de Petri temporizada, es decir, la estructura además de las restricciones de tiempo que le son asignadas. La estructura y el marcado son obtenidos a partir de la herramienta *Spades*, mientras que las restricciones de tiempo son obtenidas a través de una interfaz que provee el simulador. Como puede observarse ésta interfaz de comunicación sirve como punto de conexión entre todos los módulos, entre ellos los tres módulos que se encargan de la funcionalidad del simulador. Esos tres módulos son: almacenamiento, simulación y graficación.

A continuación se hace una descripción breve de las funciones que le han sido asignadas a cada uno éstos módulos.

- **Módulo de Almacenamiento.** Se encarga de dos tareas fundamentales: guardar datos en archivos de texto planos y recuperar los datos almacenados en estos archivos para cargarlos de nuevo en memoria. Los datos que se pueden almacenar en archivos de texto plano son de dos tipos: elementos de temporización global, correspondientes a una red de Petri con cronómetros globales (derivadas, invariantes, inicializaciones y guardas), los resultados de la simulación, es decir, la lista de cronómetros y sus valores correspondientes durante alguna simulación,

- **Módulo de Simulación.** En éste módulo se encuentra la implementación del algoritmo de simulación propuesto en esta tesis, acompañado de una serie de procedimientos necesarios para facilitar la manipulación de todos los elementos de la red de Petri, con la finalidad de llevar a cabo la ejecución del modelo de acuerdo a las condiciones iniciales.

  Módulo de Graficación. Se encarga de graficar los datos de salida de una o varias simulaciones previas de acuerdo a las opciones configuradas, esto con la finalidad de permitir al usuario hacer comparaciones y análisis de resultados.

3.1.2. Diagramas de estructura

Los diagramas de estructura sirven para describir los elementos de una especificación que son independientes del tiempo, es decir, que la información que proporcionan es referente a los elementos y datos que componen la especificación, así como la forma en la que se relacionan entre ellos.

Como parte de los diagramas de estructura se presentan a continuación dos diagramas de clases, los cuales describen parte de la estructura de la red de Petri temporizada. Cada clase tiene asignada una lista de atributos (datos) y métodos (funciones) que incluyen niveles de acceso y que en conjunto definen las características de un conjunto de objetos. También se muestran las conexiones que existen entre clases, cada una de estas conexiones es una relación que refleja cuales clases comparten información entre si.

El diagrama de la Figura 3.2 representa la estructura de la red de Petri temporizada. Ésta red se representa con la clase *Net*, que se encarga de concentrar todos los datos de la red de Petri temporizada; a ésta red se le agregan de cero a muchos lugares y transiciones temporizadas, además de un cronómetro especial que se utiliza como de reloj de simulación. Por su parte a cada lugar se le pueden agregar: derivadas e invariantes, mientras que a una transición se le pueden agregar: guardas e inicializaciones. Como se puede observar cada una de estos elementos temporizados se compone de un cronómetro.

Por otra parte el diagrama de la Figura 3.3 representa la estructura interna del simulador. La interfaz de usuario principal se encuentra en la clase *"SimuladorFrame"*, a partir de esta interfaz se configuran los valores iniciales de la simulación, entonces se realiza la simulación

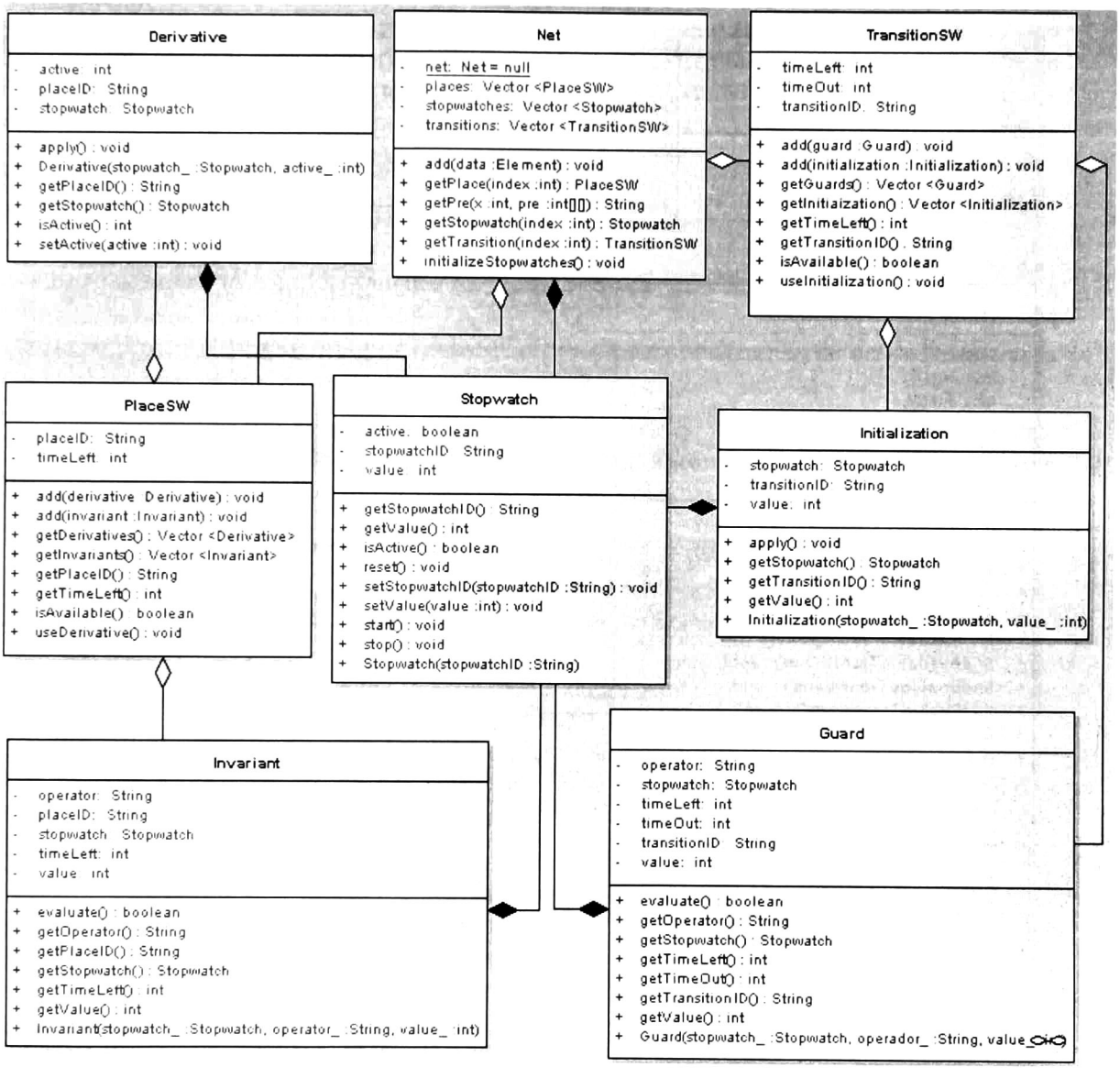


Figura 3.2: Diagrama de clases de la estructura de la red de Petri temporizada

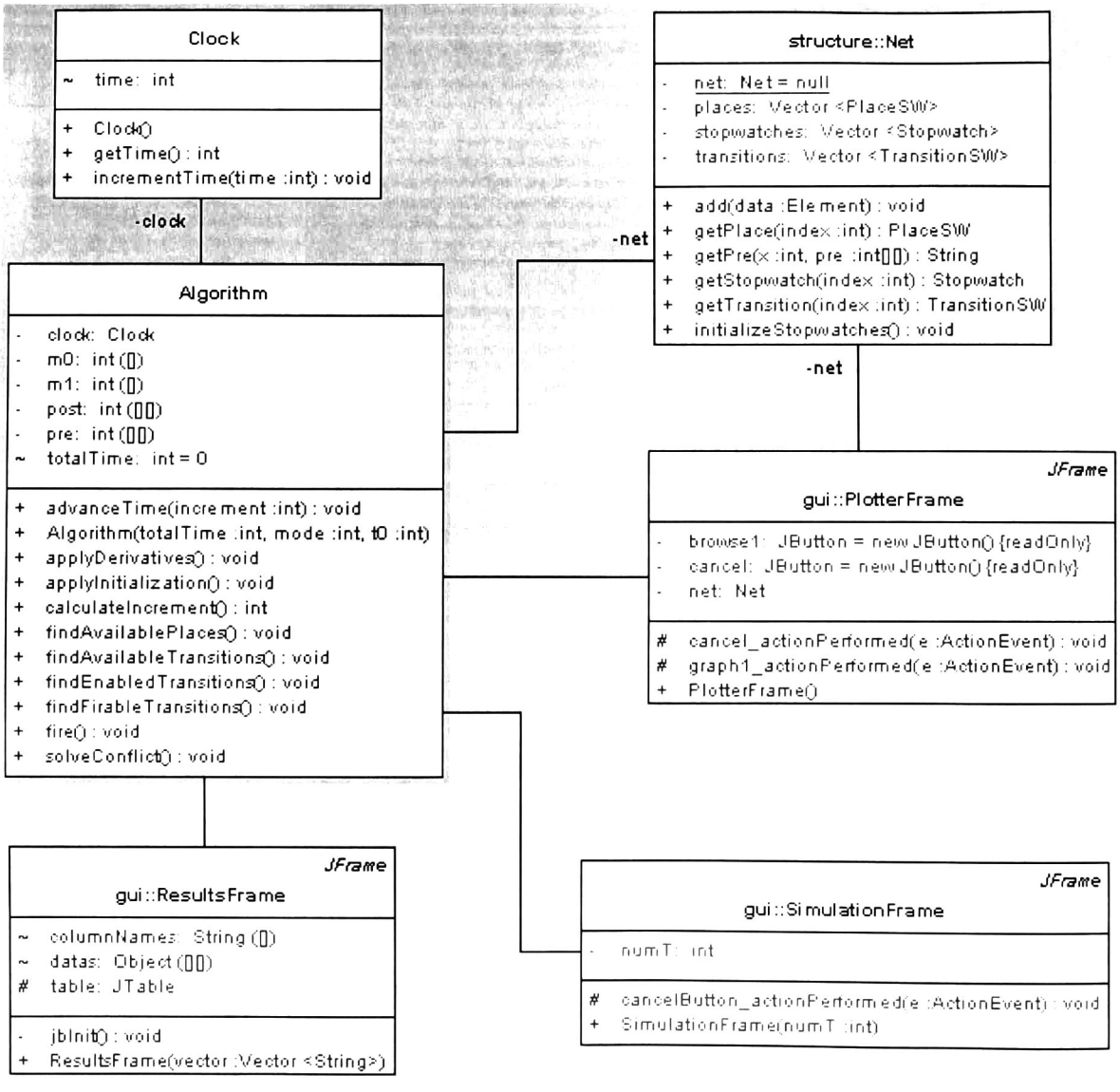


Figura 3.3: Diagrama de clases del simulador

completa, de la cual se encarga el algoritmo de simulación. Este algoritmo se relaciona directamente con el reloj de simulación y con la red de Petri (*Net*) que es donde se concentra todos los datos de la red. Finalmente, cuando se obtienen los resultados de la simulación los procesa *"ResultFrame"*. De manera independiente se encuentra la clase *"Plotter"* que retoma la información de la red además de los resultados que obtiene el algoritmo de simulación y los grafica.

3.1.3. Diagramas de comportamiento

Los diagramas de comportamiento en general sirven para ilustrar la funcionalidad que debe proveer el sistema y la interacción que pueden tener los actores con el sistema. En esta sección se presenta un diagrama de casos de uso y algunos diagramas de actividades con la finalidad de mostrarle el comportamiento del simulador.

Un diagrama de casos de uso permite ilustrar los servicios visibles externamente que proporciona el sistema en el contexto de su entorno [21]. La Figura 3.4 corresponde al diagrama de casos de uso, que describe las diferentes formas de interacción del usuario con el sistema.

Como se puede observar las funciones que puede realizar el usuario son:
a) proporcionar los datos de entrada: red, restricciones temporales y estrategia de simulación.
b) recibir los datos de salida: gráficas, lista con las actualizaciones de los cronómetros.

Las funciones que realiza de forma interna el simulador son:
a) recibir la red de Petri del usuario y procesarla.
b) realizar la simulación de acuerdo a las condiciones iniciales.
c) entregar al usuario los resultados de la simulación.

Los diagramas de actividades se utilizan para modelar una vista dinámica del sistema, que se representa como un flujo de control entre actividades, con lo que es posible mostrar las tareas que se realizan y el orden en el que son ejecutadas. El diagrama de la Figura 3.5 corresponde al diagrama de actividades del simulador, en el se muestra el flujo de actividades que se realizan para completar una simulación. Como se puede observar el primer paso consiste en tomar como datos de entrada la red de Petri marcada y después añadirle las restricciones de tiempo, para así tener completa una red de Petri temporizada; ésta será analizada por un verificador de propiedades para determinar que efectivamente se cumple con las restricciones del modelo SWPN. Hecho todo lo anterior se procede a realizar la simulación, hasta que ésta finalize de acuerdo a alguno de los criterios de terminación definidos.

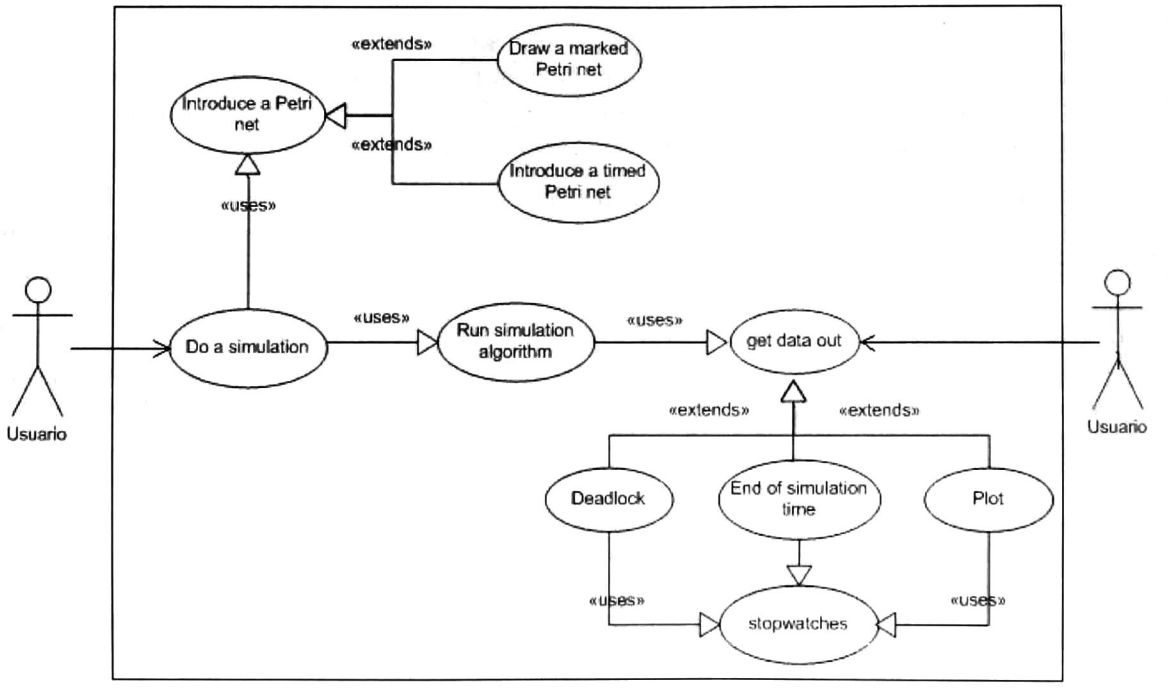


Figura 3.4: Diagrama de casos de uso

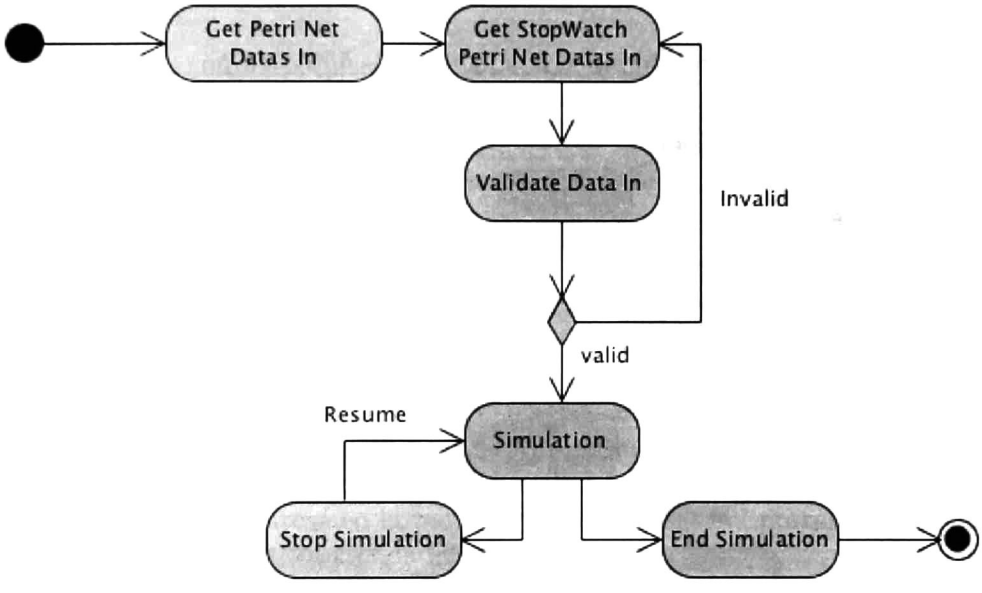


Figura 3.5: Diagrama de Actividades del simulador

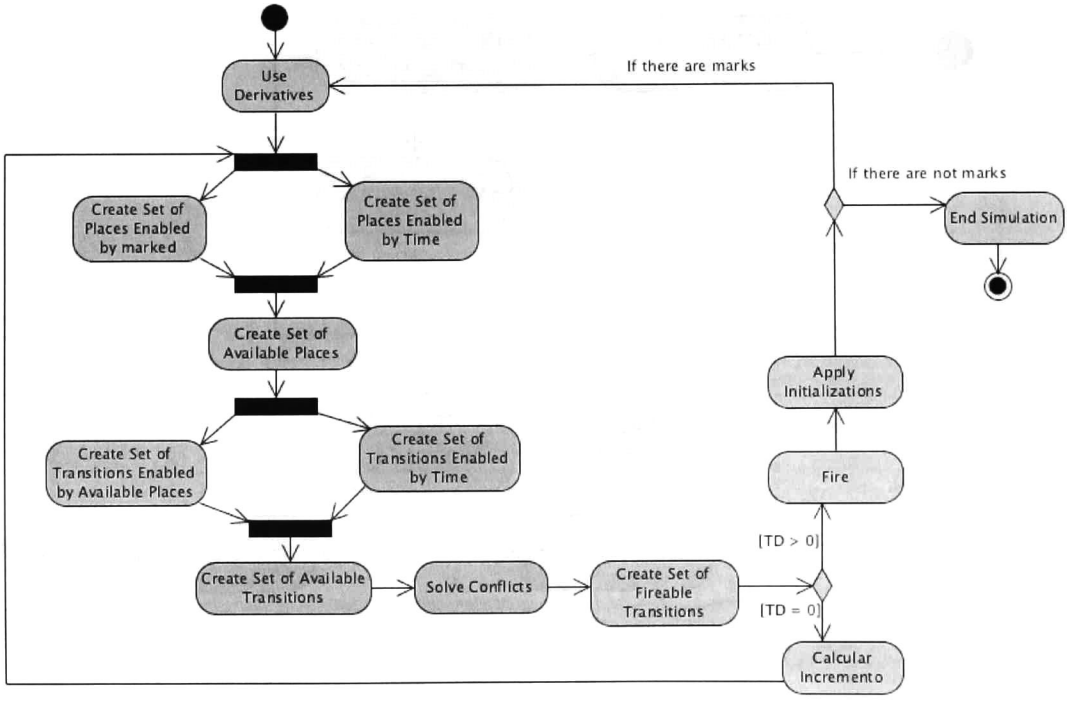


Figura 3.6: Diagrama de actividades del algoritmo de simulación

El diagrama de laFigura 3.6 muestra los pasos que realiza el algoritmo durante el proceso de simulación.

1. Inicia buscando los lugares marcados y utiliza las derivadas correspondientes.

2. Se crea el conjunto de lugares habilitados por marcado.

3. Se crea el conjunto de lugares habilitados por tiempo.

4. Se forma un nuevo conjunto que será llamado el conjunto de los lugares disponibles, que es el resultado de la intersección de los conjuntos anteriores (pasos 2 y 3).

5. Se crea el conjunto de transiciones habilitadas por lugares disponibles y además se crea el conjunto de transiciones habilitadas en tiempo y se forma un nuevo conjunto que es la intersección de estos dos, al que se le llama el conjunto de transiciones disponibles para disparo.

En seguida se resuelven los conflictos estructurales que se presenten dentro del conjunto actual y con esto se forma el conjunto de las transiciones disparables. Si se trata de un conjunto vacío entonces se busca un incremento de tiempo; si hay al menos un elemento entonces ejecuta los disparos de las transiciones y realiza las inicializaciones correspondientes. Al terminar verifica si hay marcas en la red y se vuelve a iniciar el proceso; en caso de que no haya más marcas se da por terminada la simulación.

El diagrama que aparece en la Figura 3.7 muestra una vista dinámica de uno de los procedimientos más importantes que utiliza el algoritmo de simulación, que es el procedimiento encargado de calcular el incremento necesario hasta el siguiente instante de tiempo donde suceden eventos relevantes.

Para lograrlo primero se hace una lista con todos los lugares que están marcados pero que no cumplen con sus restricciones de tiempo, y al mismo tiempo se crea otra lista con todas las transiciones habilitadas por lugares disponibles pero que aun no cumplen con sus restricciones de tiempo. Una vez que se han completado ambas listas, se calcula el tiempo que le falta a cada uno de los elementos que pertenecen a esas dos listas antes de que se cumplan sus restricciones de tiempo. Finalmente elige al que le falta el menor tiempo para cumplir con sus restricciones de tiempo. Ese lapso de tiempo es a lo que se le llama incremento, el cual representa el avance del tiempo de simulación.

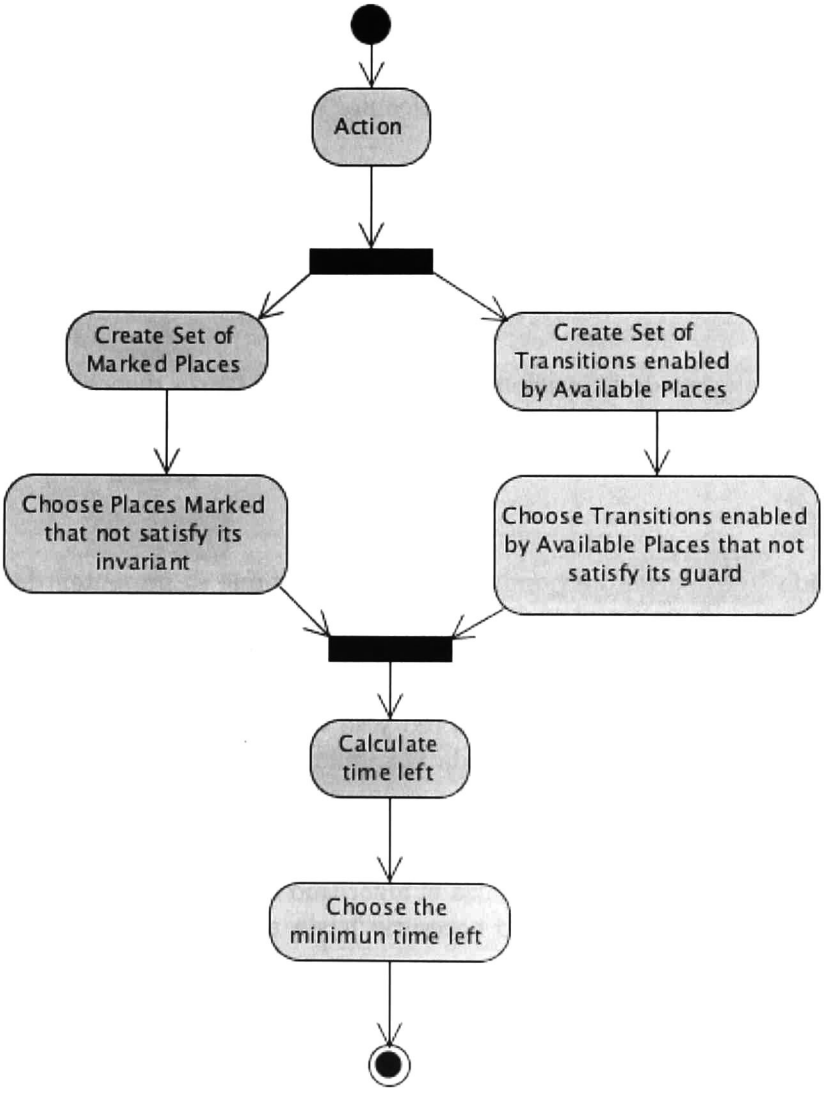


Figura 3.7: Diagrama de actividades del incremento

3.2. Implementación

Para lograr el desarrollo favorable de un proceso de software es necesario basarse en alguno de los modelos generales que definen los paradigmas del desarrollo de software; en este caso se siguió el enfoque en cascada. De manera que se estudió y trabajó por separado en cada una de las actividades que comprenden el proceso del software.

Una vez que se obtuvo un conjunto de modelos que permitían definir las características deseadas por el simulador se continuó con la parte de la implementación de código. El algoritmo propuesto fue codificado en Java con la finalidad de hacer un código portable que se pueda utilizar sobre cualquier plataforma.

Se dejo empaquetada la implementación en un módulo que se incorporó al Spades (herramienta de software para el diseño gráfico de modelos con redes de Petri).

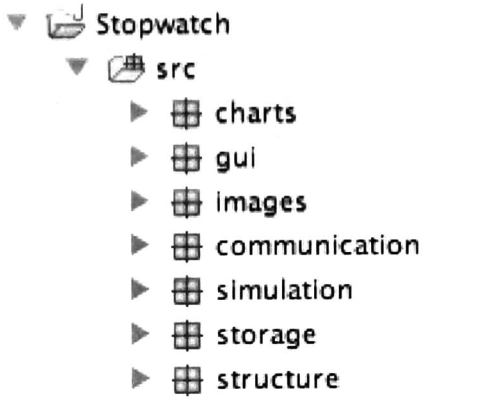


Figura 3.8: Paquetes de clases de la implementación

La implementación del software se organizó en paquetes como se muestra en la Figura 3.8. Todas las clases que pertenecen a un mismo paquete trabajan en conjunto para lograr la realización de una tarea; a continuación se describe la finalidad de cada paquete:

1. **CHARTS** (gráficas): En él se concentran todas las clases relacionadas con la creación de las gráficas.

2. **GUI**: En él se concentran todas las clases que sirven de interfaz gráfica de usuario (GUI).

3. **IMAGES**: En él se concentran todas las imágenes (PNG) que requiere el simulador.

4. **COMMUNICATION**: En él se concentran todas las clases que actúan como medio de comunicación entre el simulador y el SPADES.

5. **SIMULATION**: En él se concentran el algoritmo de simulación y todas las clases que en cooperan con él para lograr la simulación de una RPCG.

6. **STORAGE**: En él se concentran todas las clases que se encargan del manejo y tratamiento de la información de entrada y salida.

7. **STRUCTURE**: En él se concentran todas las clases que forman la estructura de la RPCG.

Módulo graficador

Es el módulo que se encarga de graficar los resultados de la simulación en una ventana independiente, éste módulo se desarrolló utilizando "**JFreeChart**", que es una librería para gráficos de Java la cual proporciona un conjunto de plantillas de gráficos, entre ellos se encuentran las gráficas de líneas, que es el tipo de gráficas que se utilizaron en la implementación. Es importante mencionar que el formato del gráfico ya está establecido, de manera que no es configurable por el usuario.
Los datos que se muestran en la gráfica son: *las series*, que corresponden al nombre de los cronómetros de la RPCG y se relacionan con una línea de color que se asigna en forma preestablecida, *el eje de las x's* corresponde al tiempo de simulación, *el eje de las y's* al valor que tienen los cronómetros y un conjunto de *líneas verticales* que corresponden al disparo de una transición.

Módulo Interfaz de comunicación

Éste módulo sirve como medio de comunicación y le permite al simulador compartir información con el usuario y/o el *Spades*. Es el vínculo que permite la comunicación entre todos los módulos del sistema y se compone de tres paquetes: *Communication, GUI e Images*. El Spades provee de una interfaz de comunicación donde están definidos los métodos que se pueden utilizar para tener acceso a: la cantidad de lugares, cantidad de transiciones, matriz de incidencia, marcado inicial, matriz pre y matriz post de la red de Petri que dibujó el usuario sobre el Spades. Las clases que se encuentran en el paquete *Communication* implementan dichos métodos para poder obtener los datos de la red de Petri.
Las interfaces de usuario que se encuentran en el paquete GUI se desarrollaron utilizando la librería **"swing"** de java, y son estas interfaces las que incorporan las imágenes que se encuentran en el paquete *Images*.

Módulo que contiene la RPCG

Es el módulo que se encarga de concentrar en el paquete *structure* toda la información sobre la red de Petri, esto incluye los datos que se obtienen del *Spades*, así como las restricciones de tiempo que le proporciona el usuario al simulador.

Módulo de almacenamiento

sirve para lecturas de entrada y escritura de salidas. Una entrada contiene la estructura de una RPCG, tras una lectura de entrada el simulador considera a la red de Petri como la más reciente. Cabe mencionar que durante una simulación siempre se utiliza la RPCG más reciente.
Una entrada se puede obtener de dos formas a través de la GUI o tras la lectura de un archivo de entrada, para éste último, en caso de que se trate de un archivo dañado o corrupto, será imposible cargar la nueva red de Petri.

Las salidas son almacenadas en archivos de texto plano, hay dos tipos de salidas: la estructura de la RPCG y los resultados de una simulación. Siempre que el usuario decida que desea almacenar esta información, este módulo es el encargado de atender esa petición.

3.2.1. Integración

Una vez finalizada la programación del código fuente, éste se exportó y se convirtió en un módulo independiente al que se le llamó simulador. Entonces comenzó el trabajo de integración con la herramienta SPADES, de manera que la misma interfaz del SPADES le permite al usuario trabajar con una RPCG, editarla, simular modelos que utilizan cronómetros globales y graficar los resultados a través de un conjunto de interfaces que le permiten al usuario hacer uso de todas las funciones del simulador.

3.3. Validación y Verificación

Para asegurar que el simulador está trabajando de acuerdo con los requerimientos solicitados se le hicieron algunos tipos de pruebas, que podemos clasificar en dos: pruebas del modelo y pruebas de integración.

Las pruebas de modelos se realizaron con una serie de ejemplos, para comprobar que los resultados obtenidos son los esperados. Se hicieron pruebas utilizando las dos estrategias de simulación, utilizando modelos no temporizados, modelos sin cotas superiores y modelos que presentan conflictos estructurales.

En el caso de las pruebas de integración se realizaron experimentos, integrando el simulador hasta lograr que trabajara de forma transparente y unificada al SPADES y se hicieron pruebas sobre las plataformas: windows y mac para asegurar la funcionalidad de la herramienta.

3.4. Conclusiones

En este capítulo se presentó el proceso de Ingeniería de software que se siguió durante el desarrollo del simulador. En el siguiente se incluyen ejemplos que muestran las características y aplicaciones del simulador, mismo que se acompaña con imágenes que muestran los tipos de pantallas que el usuario podrá utilizar.

Capítulo 4

Utilización

Resumen

En éste capítulo se presenta la herramienta de simulación para redes de Petri con cronómetros globales incluyendo imágenes que permiten ilustrar todas las características del simulador.

Después de mostrar la herramienta se presentan algunos casos de estudio con la finalidad de mostrar la funcionalidad del software, es decir, como se capturan los datos de entrada, como son procesados y como es la obtención de resultados.

Finalmente se incluyen los resultados obtenidos en varias simulaciones a través de gráficas.

4.1. Presentación de la herramienta

Para mostrar el uso de la herramienta se presenta a continuación un ejemplo. Se trata de una RPCG que tiene dos lugares y tres transiciones, en la Figura 4.1 se muestra el modelo completo que incluye las restricciones de tiempo para cada nodo de la red. Como se puede observar para los lugares se incluyen derivadas e invariantes, mientras que para las transiciones se incluyeron inicializaciones y guardas.

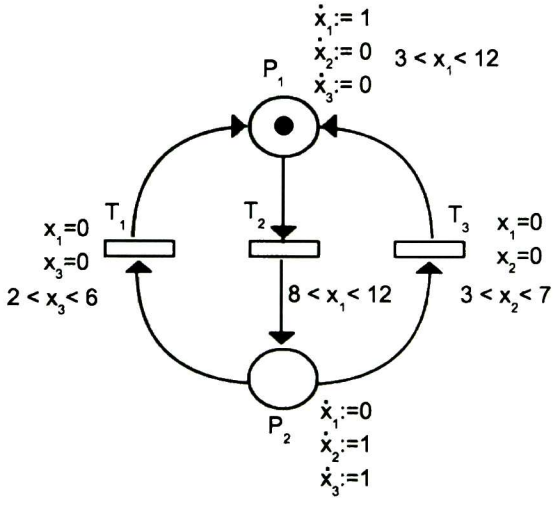


Figura 4.1: RPCG

La Figura 4.2 muestra el aspecto de la RPCG de la Figura 4.1 cuando es dibujada con la herramienta SPADES.

Hasta este momento solo se tiene una red de Petri marcada sin elementos que la hagan dependiente del tiempo; para poder agregar las restricciones temporales el simulador dispone de un conjunto de menús que facilitan las operaciones de agregar y eliminar restricciones temporales a los nodos de la red.

Lo primero que se tiene que hacer para poder dar de alta restricciones temporales sobre un modelo, es definir la cantidad de cronómetros que utiliza el modelo. En la Figura 4.3 se muestra la opción del menú que se debe elegir para agregar los cronómetros del modelo.

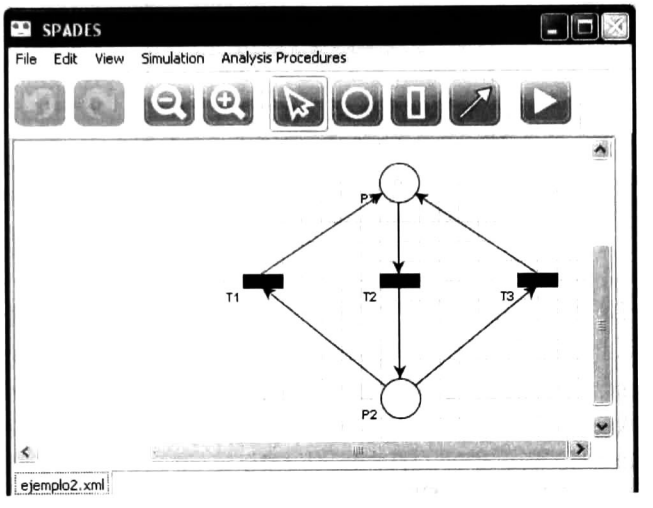


Figura 4.2: Red de Petri dibujada en el SPADES

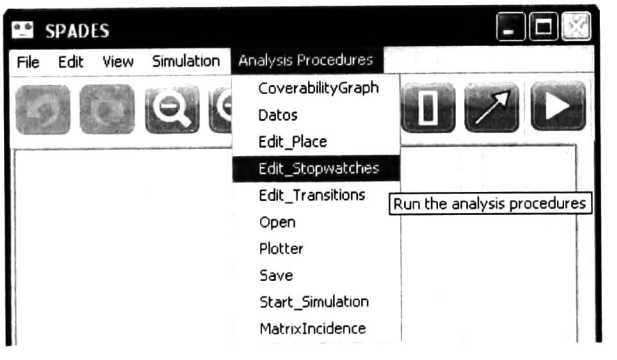


Figura 4.3: Menú de opciones

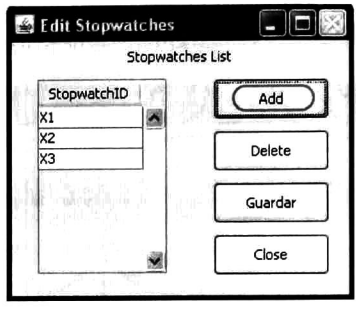


Figura 4.4: Registrar los cronómetros del modelo

La Figura 4.4 muestra la interfaz donde el usuario puede agregar o eliminar cronómetros del modelo. El botón ADD añade automáticamente un cronómetro y le asigna el identificador correspondiente. En caso de que el usuario desee retirar un cronómetro del modelo, basta con seleccionarlo y presionar el botón DELETE; entonces el cronómetro y todos los elementos relacionados con él, son eliminados del modelo. Finalmente, antes de abandonar esta sección el usuario debe decidir si guarda o descarta las modificaciones hechas sobre los cronómetros del modelo con los botones SAVE y CLOSE respectivamente.

Restricciones temporales sobre los lugares

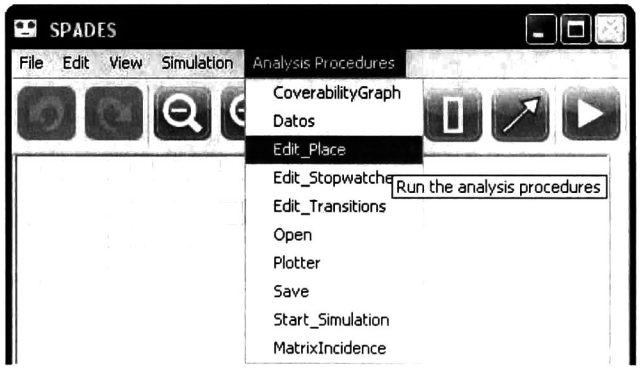


Figura 4.5: Menú de opciones

El usuario utilizará el menú que se muestra en la Figura 4.5 en cualquier momento, cada vez que necesite modificar los atributos temporales de los lugares. A los lugares se les atribuyen dos tipos de restricciones temporales: derivadas e invariantes.

La Figura 4.6 muestra como agregar una derivada.

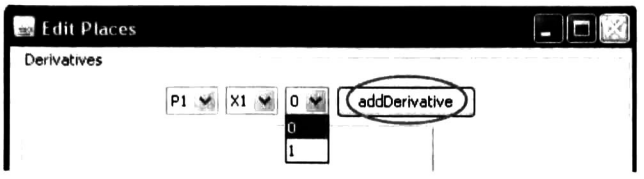


Figura 4.6: Agregar derivadas

Para facilitar este proceso el usuario cuenta con tres listas desplegables; la primera corresponde a la lista de lugares de la red, la segunda es la lista de cronómetros del modelo y el último dato representa si el cronómetro elegido está avanzando o detenido, lo cual se representa con los valores cero y uno respectivamente.
La Figura 4.7 muestra como agregar un invariante.

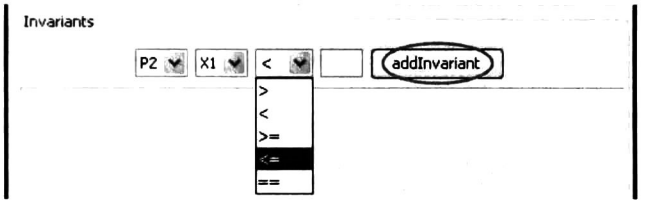


Figura 4.7: Agregar invariantes

Este menú cuenta con tres listas desplegables, la primera contiene la lista de lugares de la red, la segunda es la lista de cronómetros del modelo y la tercera los operadores permitidos para construir una expresión booleana; el último dato lo debe ingresar el usuario, con la restricción de que tiene que ser un número entero positivo.

La Figura 4.8 muestra la interfaz completa donde se pueden editar las restricciones temporales de todos los lugares del modelo. Una vez que el usuario ha seleccionado los datos de la derivada, al presionar el botón ADD-DERIVATIVE confirma que desea que ésta sea agregada a la lista de derivadas del modelo. De igual manera sucede con las invariantes, una vez que el usuario ha seleccionado todos los datos del invariante, al presionar el botón ADD-INVARIANT confirma que desea que ésta sea agregada a la lista de invariantes del modelo.

Si el usuario selecciona una derivada o una invariante de una de las listas respectivamente y presiona el botón DELETE entonces ésta será eliminada del modelo. El botón SAVE sirve para guardar los cambios hechos en el modelo y el botón CLOSE para cerrar la ventana de propiedades de los lugares.

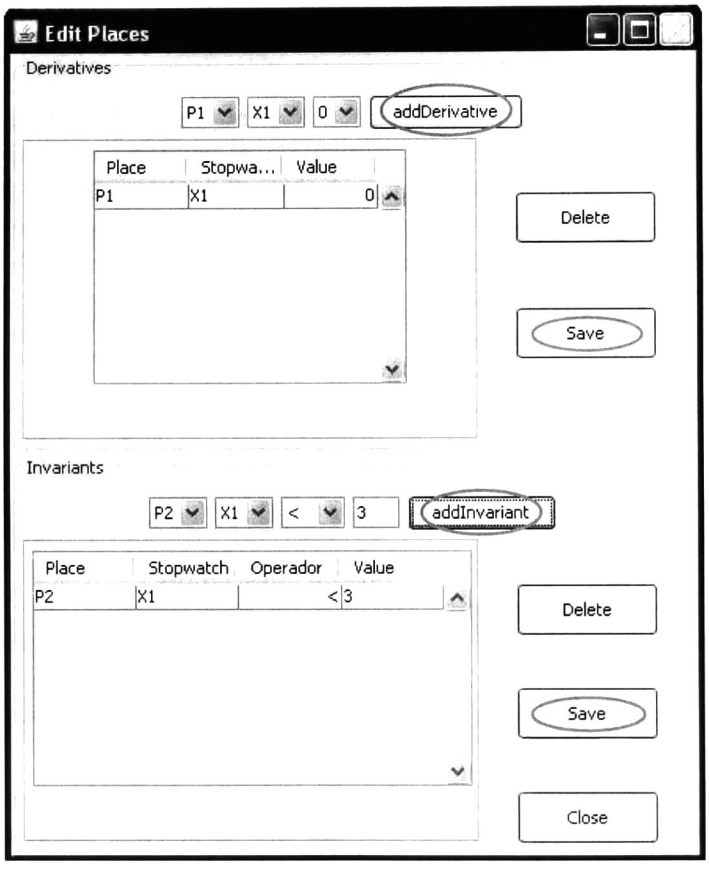


Figura 4.8: Editar lugares

Restricciones temporales sobre las transiciones

El usuario utilizará el menú que se muestra en la Figura 4.9 en cualquier momento, cuando necesite modificar los atributos temporales de las transiciones. A las transiciones se les atribuyen dos tipos de restricciones temporales: inicializaciones y guardas.

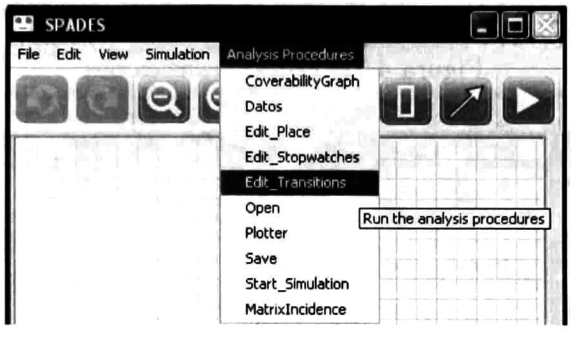


Figura 4.9: Menú de opciones

La Figura 4.10 muestra como agregar una inicialización. Para facilitar este proceso el usuario cuenta con dos listas desplegables, la primera corresponde a la lista de transiciones de la red, la segunda es la lista de cronómetros del modelo y el último espacio es para un dato numérico que el usuario debe teclear, este representa el nuevo valor con el que será inicializado el cronómetro elegido.

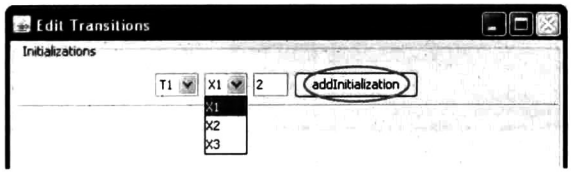


Figura 4.10: Editar transiciones

La Figura 4.11 muestra como agregar una guarda. Este menú cuenta con tres listas desplegables, la primera contiene la lista de transiciones de la red, la segunda es la lista de cronómetros del modelo y la tercera, los operadores permitidos para construir una expresión booleana; el último es un dato que debe ingresar el usuario, con la restricción de que tiene que ser un número entero positivo.

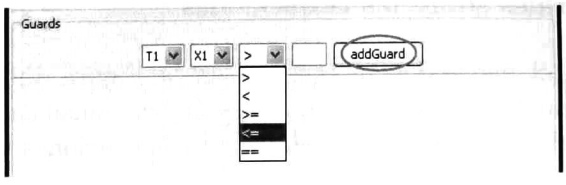


Figura 4.11: Editar transiciones

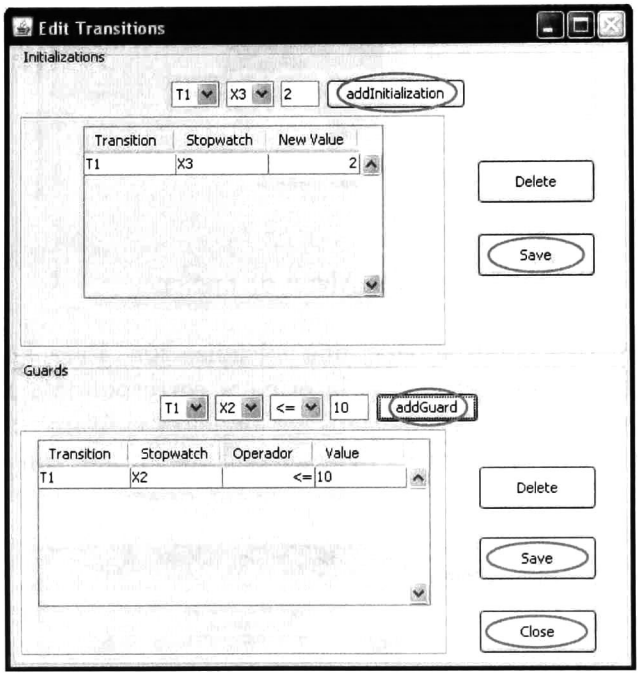


Figura 4.12: Editar transiciones

La Figura 4.12 muestra la interfaz completa donde se pueden editar las restricciones temporales de todas las transiciones del modelo. Una vez que el usuario ha seleccionado los datos de la inicialización, al presionar el botón ADD-INICIALIZATION confirma que desea que ésta sea agregada a la lista de inicializaciones del modelo. De igual manera sucede con las guardas, una vez que el usuario ha seleccionado todos los datos de la guarda, al presionar el botón ADD-GUARD confirma que desea que ésta sea agregada a la lista de guardas del modelo.

Si el usuario selecciona una inicialización o una guarda de una de las listas respectivamente y presiona el botón DELETE entonces ésta será eliminada del modelo. El botón SAVE sirve

para guardar los cambios hechos en el modelo y el botón CLOSE para cerrar la ventana de propiedades de las transiciones.

Simulación

El usuario utilizará el menú que se muestra en la Figura 4.13 en cualquier momento que desee iniciar una simulación.

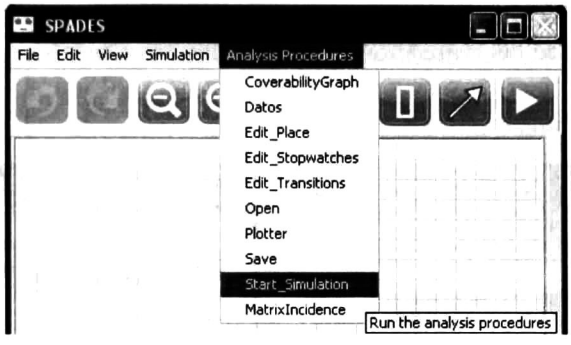


Figura 4.13: Menú de opciones

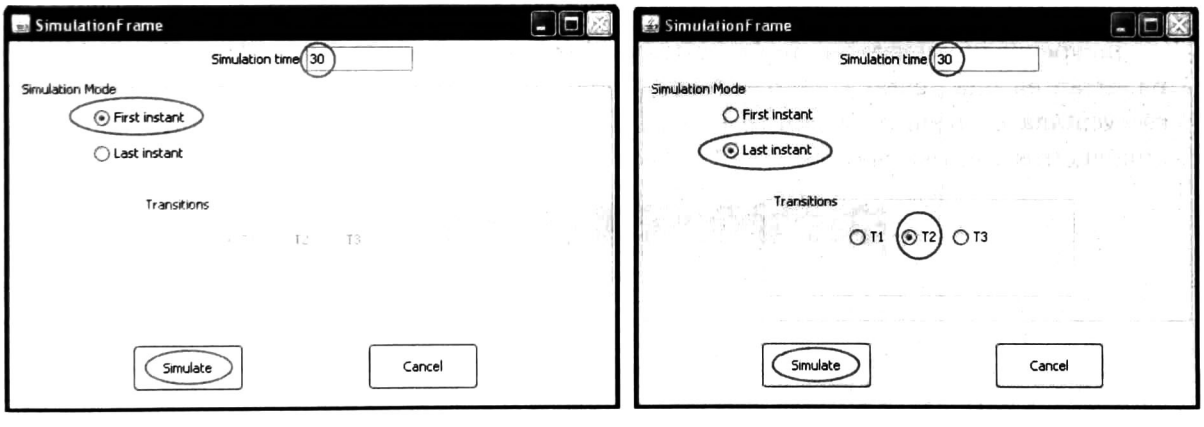


Figura 4.14: Estrategias de simulación

En la Figura 4.14 se muestra la ventana donde se configuran los parámetros de la simulación. Estos parámetros incluyen el tiempo total de simulación y la elección de la estrategia. La imagen del lado izquierdo es para una simulación utilizando la primer estrategia, mientras que en la imagen del lado derecho se presenta un ejemplo para la segunda estrategia

de simulación. Como puede observarse en la segunda imagen cuando se elige esta estrategia, inmediatamente se habilita una lista con todas las transiciones del modelo, para que el usuario seleccione una transición que será la que se disparará al último instante, mientras que el resto de las transiciones serán disparadas al primer instante de tiempo en el que estén disponibles. En este ejemplo la segunda transición será la que se dispare hasta el último instante de tiempo en el que se encuentre disponibles.

El algoritmo de simulación se ejecutará de manera ininterrumpida hasta que se cumpla el tiempo total de simulación o se bloquee la red, lo que suceda primero. En caso de que se presente un conflicto entre las transiciones disparables, será necesario que el usuario resuelva el conflicto confirmando cual transición será disparada. La Figura 4.15 muestra un ejemplo de conflicto entre trasiciones.

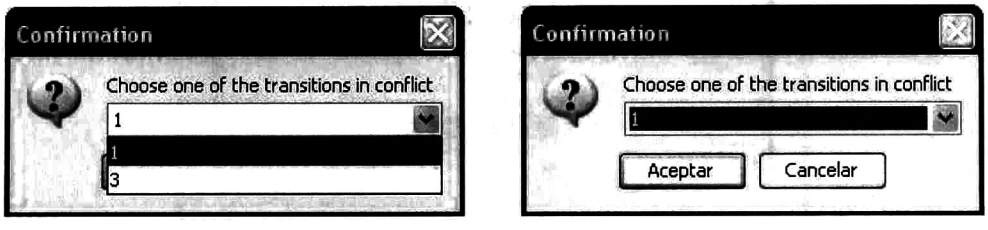
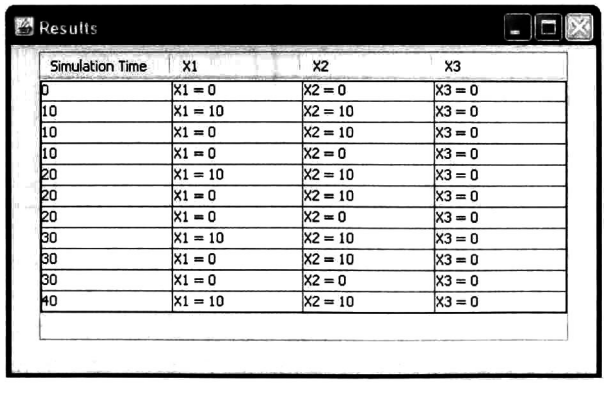


Figura 4.15: Resolver conflictos

Independientemente del tipo de estrategia que se utilice al finalizar una simulación, se muestran en una ventana los resultados obtenidos. La Figura 4.16 muestra un ejemplo de esa ventana de resultados, ahí se pueden observar todos los cambios que surgieron en los cronómetros y el tiempo total transcurrido.



Simulation Time	X1	X2	X3
0	X1 = 0	X2 = 0	X3 = 0
10	X1 = 10	X2 = 10	X3 = 0
10	X1 = 0	X2 = 10	X3 = 0
10	X1 = 0	X2 = 0	X3 = 0
20	X1 = 10	X2 = 10	X3 = 0
20	X1 = 0	X2 = 10	X3 = 0
20	X1 = 0	X2 = 0	X3 = 0
30	X1 = 10	X2 = 10	X3 = 0
30	X1 = 0	X2 = 10	X3 = 0
30	X1 = 0	X2 = 0	X3 = 0
40	X1 = 10	X2 = 10	X3 = 0

Figura 4.16: Ventana de resultados

Graficador

El usuario utilizará el menú que se muestra en la Figura 4.17 en cualquier momento, siempre que necesite representar gráficamente los resultados obtenidos en alguna simulación previa.

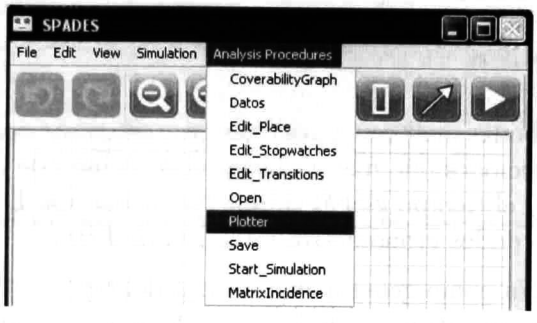


Figura 4.17: Graficar todos los cronómetros

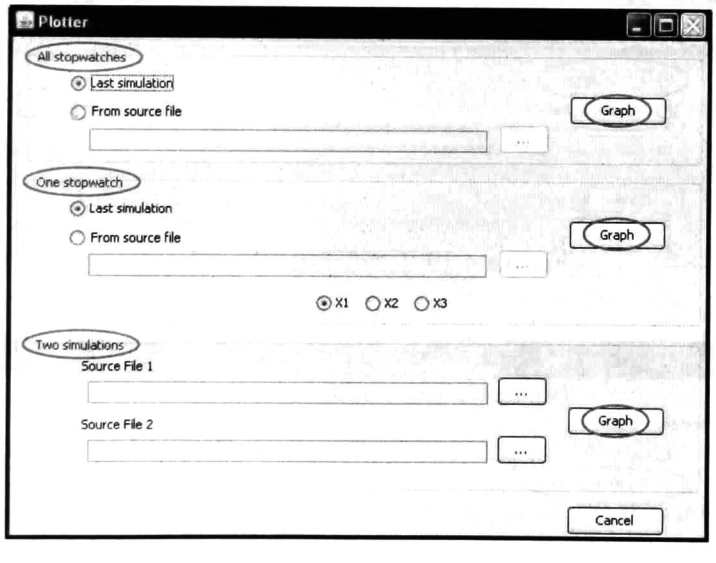


Figura 4.18: Graficador

La Figura 4.18 muestra la ventana donde se configuran los parámetros para obtener gráficas. El usuario puede obtener tres tipos de gráficas:

- **(Tipo 1)** Gráficas que incluyen todos los cronómetros del modelo y todos los cambios que sufrió cada cronómetro durante la simulación.
- **(Tipo 2)** Gráficas de un cronómetro, de manera que se pueda observar la evolución de un cronómetro en particular.
- **(Tipo 3)** Gráficas donde se combinan dos simulaciones previas, para permitirle al usuario hacer análisis comparativos.

Todas la gráficas obtenidas incluyen, para fines de análisis, un conjunto de líneas verticales que representan las transiciones que fueron disparadas. A lado de cada línea se encuentra el nombre de la transición y el tiempo global en que fue disparada. Las leyendas de las gráficas corresponden a cada uno de los cronómetros de la red de Petri.

La Figura 4.19 muestra como generar una gráfica del tipo 1 utilizando los resultados de la simulación más reciente mientras que la Figura 4.20 muestra como generar una gráfica del tipo 1 utilizando los resultados de una simulación previa cuyos resultados están almacenados en un archivo.

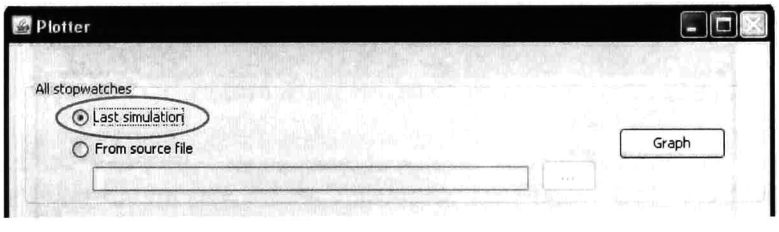


Figura 4.19: Gráfica del tipo 1 (a)

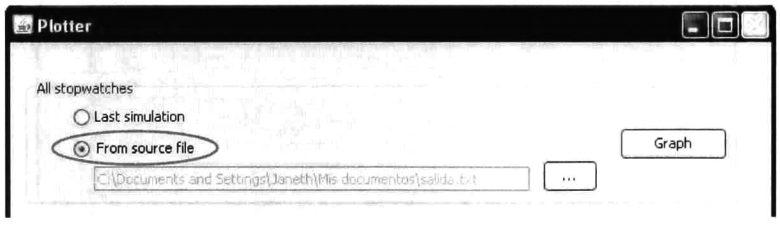


Figura 4.20: Gráfica del tipo 1 (b)

Como resultado se obtiene una gráfica donde se pueden observar todos los cronómetros del modelo, observe la Figura 4.21

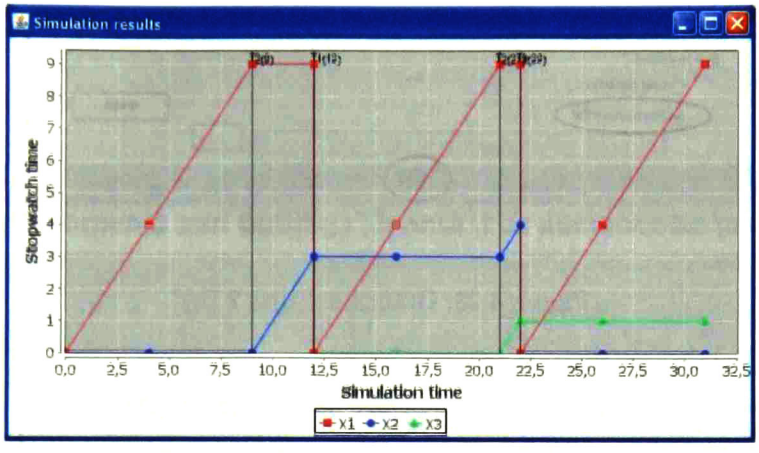


Figura 4.21: Gráfica tipo 1

La Figura 4.22 muestra como generar una gráfica del tipo 2 utilizando los resultados de la simulación más reciente, mientras que en la Figura 4.23 se muestra como generar una gráfica del tipo 2 utilizando los resultados almacenados en un archivo para una simulación previa.

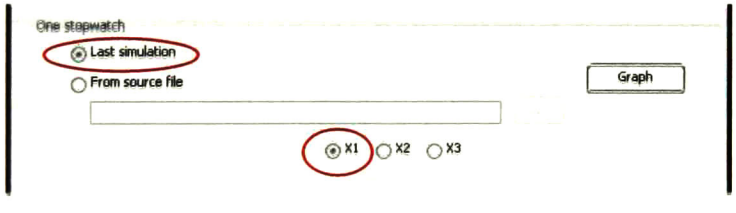


Figura 4.22: Gráfica del tipo 2 (a)

Como resultado se obtiene una gráfica donde se pueden observar todos los cronómetros del modelo, observe la Figura 4.24

La Figura 4.25 muestra como generar una gráfica del tipo 3 utilizando los resultados de dos simulaciones previas cuyos resultados fueron almacenados en archivos. Como resultado se obtiene una gráfica como la que se muestra en la Figura 4.26

Figura 4.23: Gráfica del tipo 2 (b)

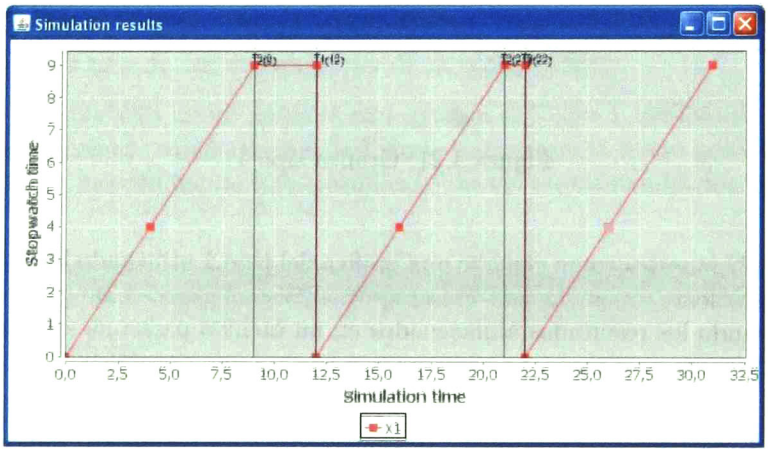


Figura 4.24: Gráfica tipo 2

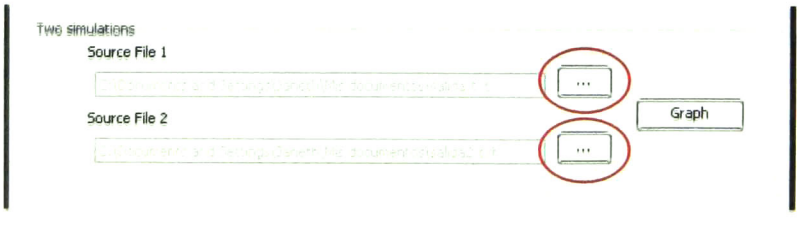


Figura 4.25: Gráfica del tipo 3

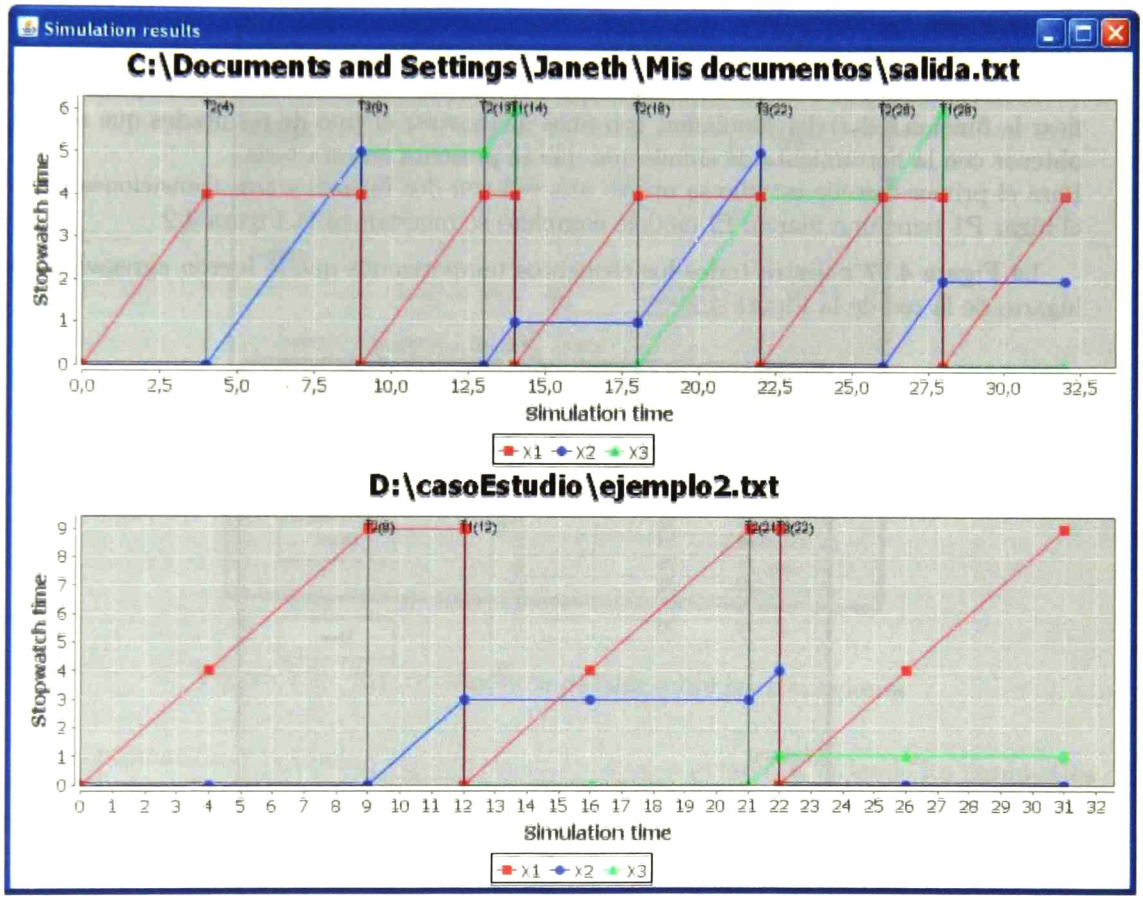


Figura 4.26: Gráfica tipo 1

4.2. Casos de Estudio

4.2.1. Caso de Estudio 1

Descripción

A continuación se muestra el modelo que se utilizará como caso de estudio para ejemplificar la funcionalidad del simulador, con fines de mostrar el tipo de resultados que se puede obtener con la herramienta de simulación que se presenta en esta tesis.
Para el primer caso de estudio se utilizo una red con dos lugares y tres transiciones. Donde el lugar P1 tiene una marca. El modelo completo se muestra en la Figura 4.2

La Figura 4.27 muestra todos los elementos temporizados que le fueron agregados a los lugares de la red de la Figura 4.2

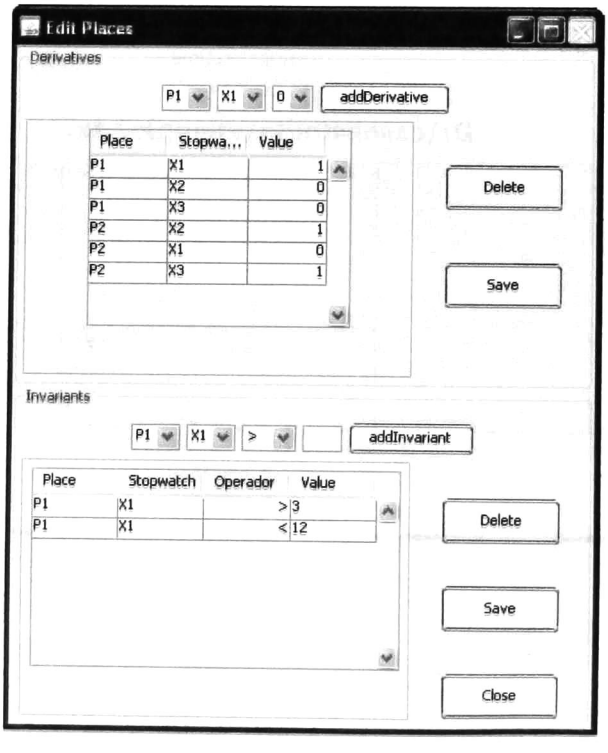


Figura 4.27: Restricciones temporales sobre los lugares

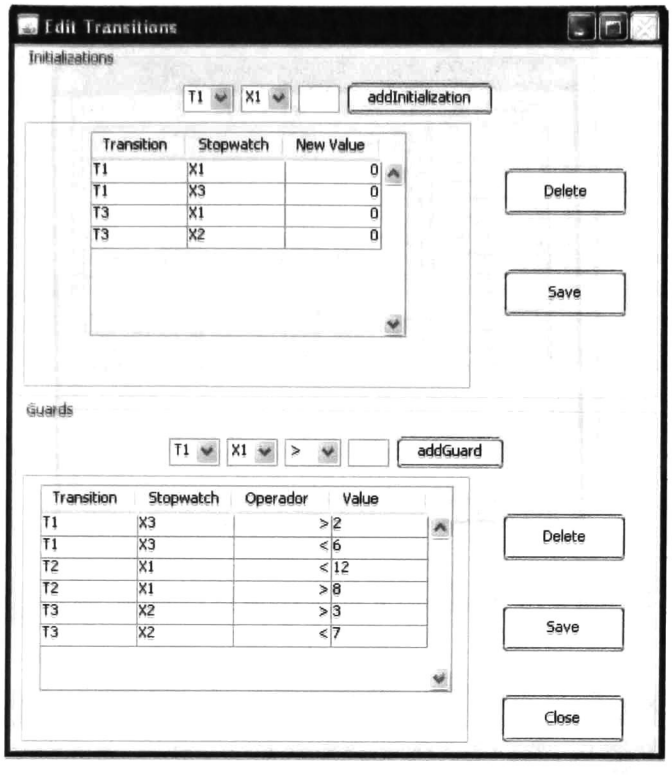


Figura 4.28: Restricciones temporales sobre las transiciones

A continuación se presentan algunos de los resultados obtenidos variando los parámetros de simulación. También se presentan algunas de las gráficas obtenidas a partir de dichos resultados.

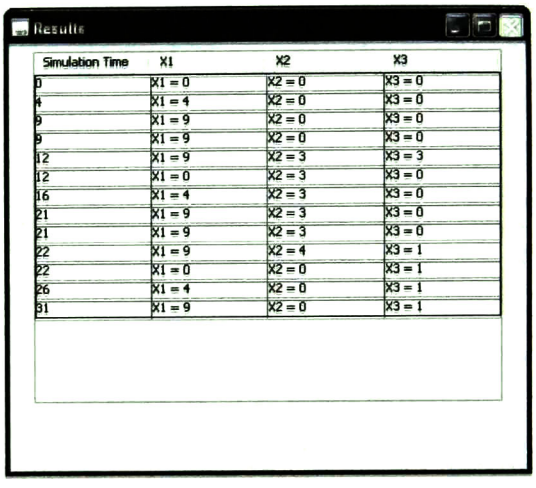


Figura 4.29: Simulación utilizando la primera estrategia

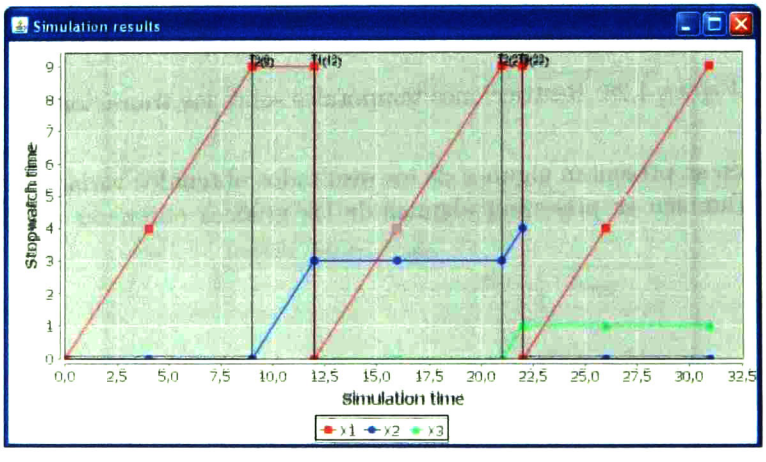


Figura 4.30: Gráfica tipo 1 de los resultados de la Figura 4.29

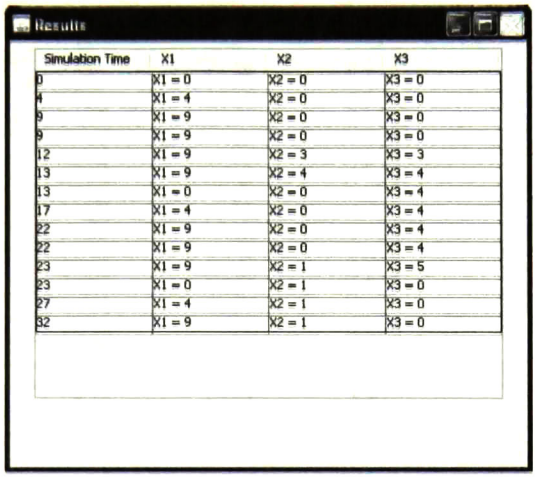


Figura 4.31: Simulación utilizando la segunda estrategia basada en T1

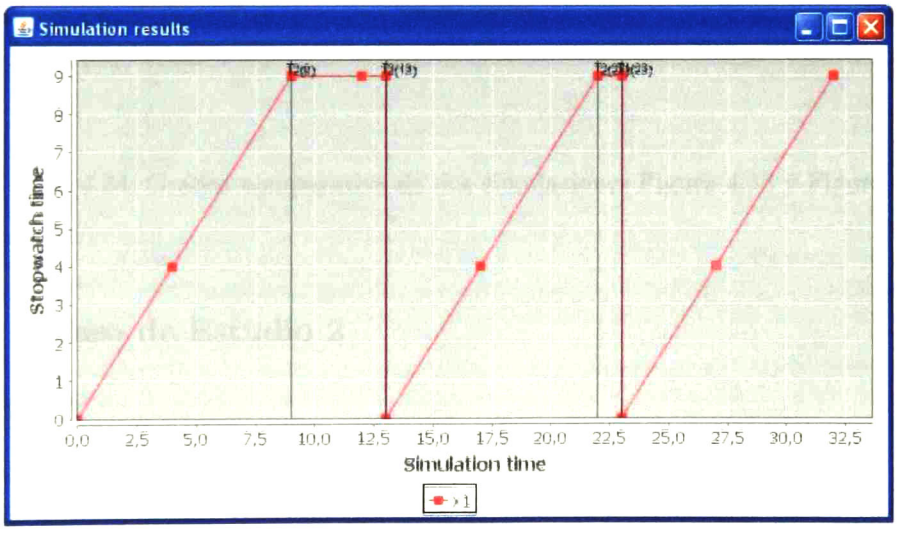


Figura 4.32: Gráfica tipo 2 de los resultados de la Figura 4.31

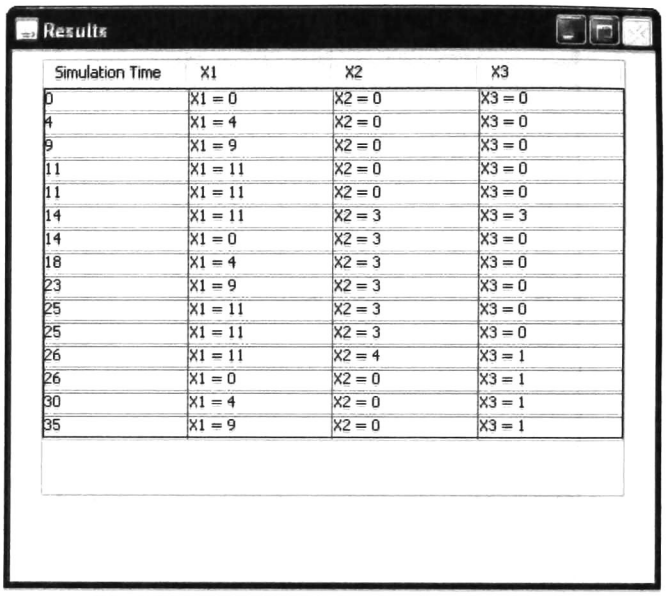


Figura 4.33: Simulación utilizando la segunda estrategia basada en T2

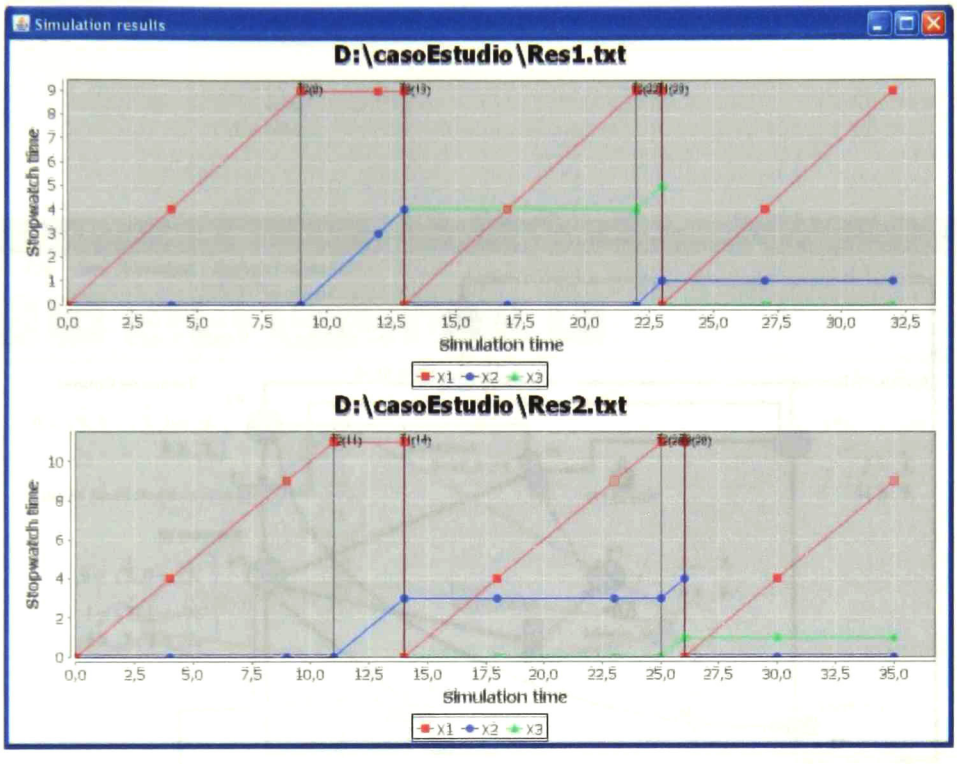


Figura 4.34: Gráfica comparativa de dos simulaciones Figura 4.33 y Figura 4.33

4.2.2. Caso de Estudio 2

Descripción

Este caso de estudio corresponde a la especificación de un protocolo de control de agentes móviles modelado con una RPCG, propuesto por M. Flores en [10].
La Figura 4.35 muestra especificación del protocolo utilizando los siguientes parámetros: $ttl = 4$, $tmax = 15$, $te = 3$, $te = 2$, $tmg1 = 1$, $tmg2 = 4$. La Figura 4.36 muestra la RPCG de la Figura 4.35 al dibujarla con el SPADES.

A continuación se muestran los resultados que se obtuvieron durante la simulación.

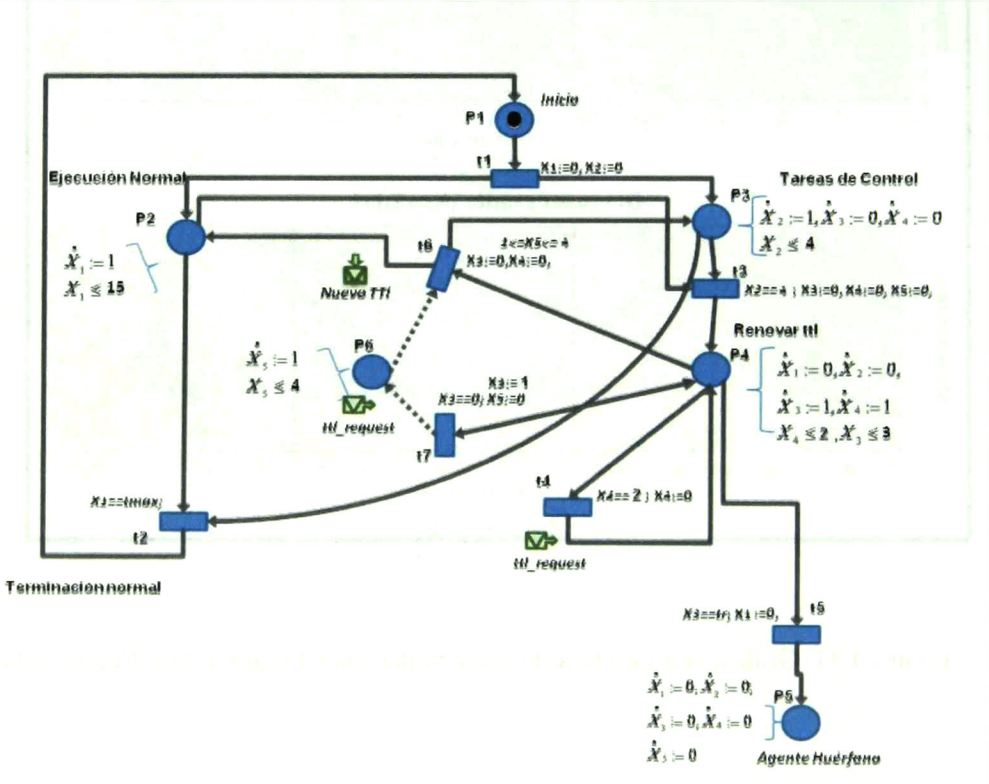


Figura 4.35: Protocolo de control de agentes móviles

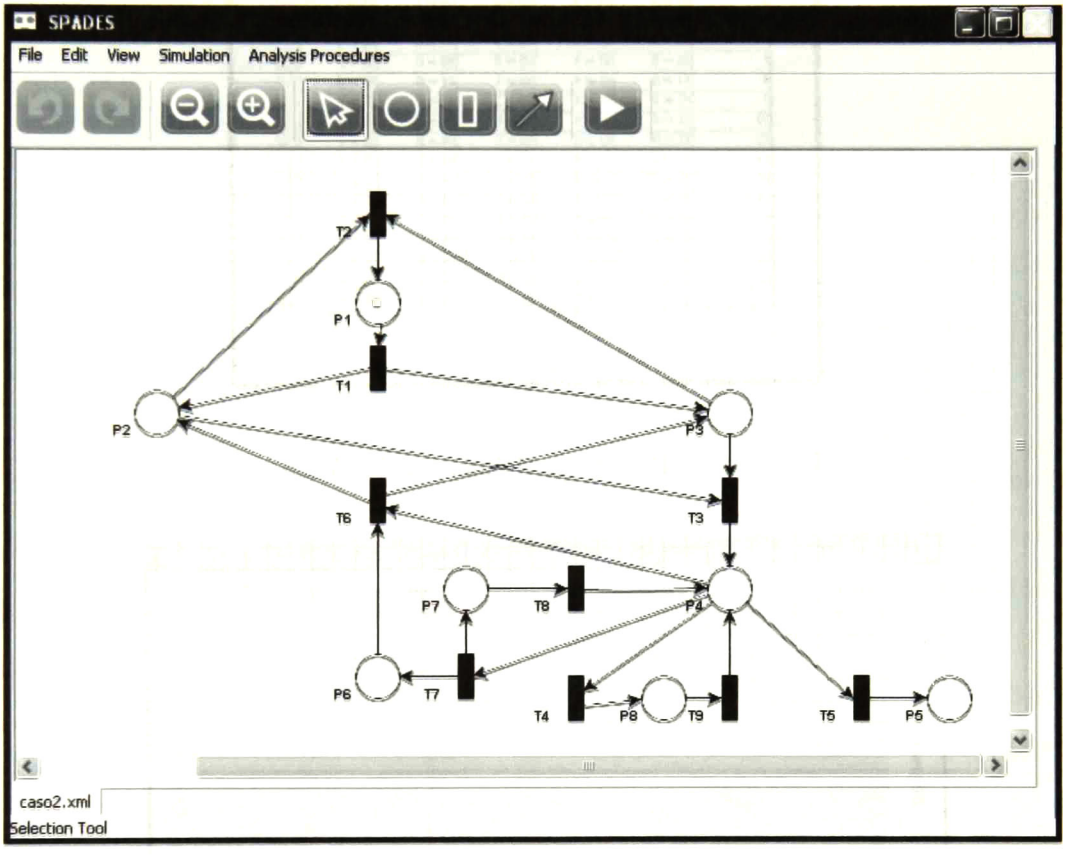
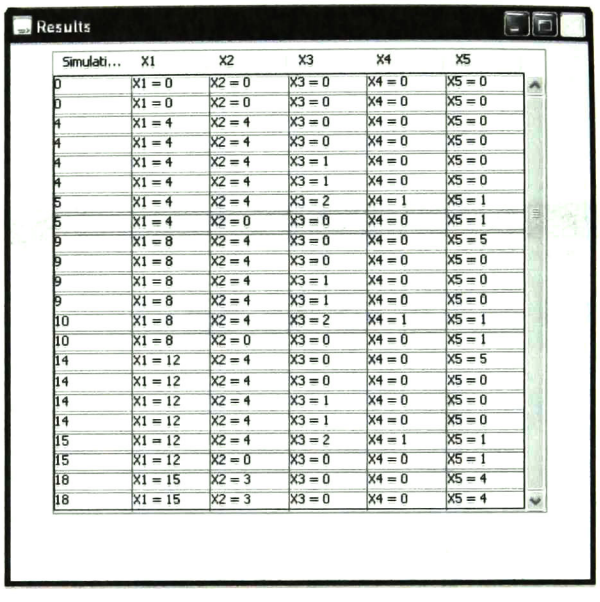


Figura 4.36: Protocolo de control de agentes móviles en el SPADES



Results

Simulati...	X1	X2	X3	X4	X5
0	X1 = 0	X2 = 0	X3 = 0	X4 = 0	X5 = 0
0	X1 = 0	X2 = 0	X3 = 0	X4 = 0	X5 = 0
4	X1 = 4	X2 = 4	X3 = 0	X4 = 0	X5 = 0
4	X1 = 4	X2 = 4	X3 = 0	X4 = 0	X5 = 0
4	X1 = 4	X2 = 4	X3 = 1	X4 = 0	X5 = 0
4	X1 = 4	X2 = 4	X3 = 1	X4 = 0	X5 = 0
5	X1 = 4	X2 = 4	X3 = 2	X4 = 1	X5 = 1
5	X1 = 4	X2 = 0	X3 = 0	X4 = 0	X5 = 1
9	X1 = 8	X2 = 4	X3 = 0	X4 = 0	X5 = 5
9	X1 = 8	X2 = 4	X3 = 0	X4 = 0	X5 = 0
9	X1 = 8	X2 = 4	X3 = 1	X4 = 0	X5 = 0
9	X1 = 8	X2 = 4	X3 = 1	X4 = 0	X5 = 0
10	X1 = 8	X2 = 4	X3 = 2	X4 = 1	X5 = 1
10	X1 = 8	X2 = 0	X3 = 0	X4 = 0	X5 = 1
14	X1 = 12	X2 = 4	X3 = 0	X4 = 0	X5 = 5
14	X1 = 12	X2 = 4	X3 = 0	X4 = 0	X5 = 0
14	X1 = 12	X2 = 4	X3 = 1	X4 = 0	X5 = 0
14	X1 = 12	X2 = 4	X3 = 1	X4 = 0	X5 = 0
15	X1 = 12	X2 = 4	X3 = 2	X4 = 1	X5 = 1
15	X1 = 12	X2 = 0	X3 = 0	X4 = 0	X5 = 1
18	X1 = 15	X2 = 3	X3 = 0	X4 = 0	X5 = 4
18	X1 = 15	X2 = 3	X3 = 0	X4 = 0	X5 = 4

Figura 4.37: Resultados de una simulación tipo 1

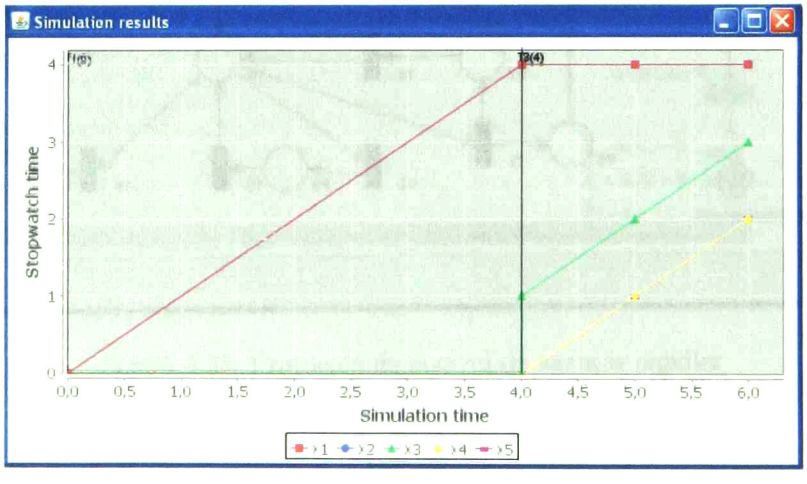


Figura 4.38: Gráfica de los resultados de la simulación anterior

4.3. Conclusiones

En este capítulo se presentan las características y funciones que provee la herramienta de simulación obtenida. Se presentan además algunos casos de estudio utilizados para comprobar su correcto funcionamiento.

Conclusiones

En esta tesis se abordó el problema de simulación de sistemas de eventos discretos con tareas interrumpibles, modelados mediante redes de Petri con cronómetros globales (RPCG).

La principal aportación de este trabajo es un algoritmo de simulación que permite encontrar una secuencia de disparos de transiciones para una RPCG de forma máxima paralela. Además, se desarrolló un módulo de software basado en algoritmo, el cual ha sido incorporado a la herramienta Spades.

El producto final es una herramienta que permite al usuario dibujar y modificar un modelo en RPCG, agregar restricciones temporales y almacenar el modelo editado. Los resultados de la simulación pueden ser visualizados a través de gráficas que muestran la evolución de los cronómetros en el tiempo de simulación y la evolución de un cronómetro respecto a otro. Los casos de estudio que se presentan forman parte del conjunto de pruebas realizadas con el simulador para comprobar su funcionalidad.

La herramienta constituye un auxiliar eficaz en la evaluación de un diseño, permitiendo ajustar iterativamente un modelo tanto en la estructura como en las restricciones temporales. La posibilidad de combinar evolución de los cronómetros de una simulación, proporciona elementos valiosos en el análisis del sistema modelado con RPCG.

El uso frecuente de esta versión beta del software desarrollado conducirá posiblemente a sugerir mejoras la definición del formalismo, el algoritmo de simulación y el simulador mismo. Por ejemplo:

- Las RPCG pueden extenderse permitiendo una estructura de RP generalizadas acotadas.
- El algoritmo puede incluir el manejo de eventos asíncronos, los cuales se calendarizan en escenarios. También se puede analizar la conveniencia de proponer otras estrategias de transiciones disponibles, adicionalmente a las dos definidas en algoritmo actual.

  El simulador puede brindar ayuda al usuario, revisando algunas propiedades cualitativas del modelo, o bien verificando algunas inconsistencias en la utilización de cronómetros.

- Otra ayuda posible es la facilidad de introducir al simulador políticas de solución de conflictos en el disparo de transiciones.
- Las interfaces con el usuario tanto en la introducción de parámetros de RPCG, como las gráficas puede ajustarse y extenderse.

Referencias

[1] M. Silva. *Las redes de Petri en la automática y la informática.* Ac, Madrid, 1985.

[2] E. López. Analysis of discrete event systems by simulation of timed petri net models. *Mathematics and Computers in Simulation*, pages 53–59, 2002.

[3] O.H. Roux y A. Déplanche. A t-time Petri net extension for real time-task scheduling modeling. *European Journal of Automation (JESA)*, pages 973 – 987, 2002.

[4] O.H. Roux D. Lime. Expressiveness and analysis of schedulling extended time Petri nets. In *5th IFAC International Conference on Fieldbus Systems and their Applications, (FET03)*, pages 193–202. Elsevier, July 2003.

[5] E. Vicario G. Bucci, A. Fedeli. Timed state space analysis of real-time preemptive systems. In *IEEE Trans. On Software Engineering*, February 2004.

[6] O.H. Roux M. Magnin, P. Molinaro. Decidability, expressivity and state-space computation of stopwatch Petri nets with discrete-time semantics. In *8th International Workshop on Discrete Event Systems*, pages 33–38, July 2006.

[7] H.Alla A. Allahham. Post and pre-inicialized stopwatch Petri nets: formal semantics and state space computation. *Nonlinear Analysis: Hybrid System*, pages 307–326, September 2008.

[8] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *HSCC*, volume 2289, pages 258–273. Springer, 2005.

[9] E. López M. Flores, A. Padilla. A population control protocol for mobile agent based workflow automation. In *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, pages 4059–4064, Texas, USA, October 2009.

[10] M. Flores. *Protocolos de tolerancia a fallas en sistemas de agentes móviles para la Metodologías TCAD para diseñar diodos gestión de flujo de trabajo epitaxiales de recuperación rápida de silicio interorganizacional.* PhD thesis, Centro de Investigación y Estudios Avanzados del IPN, Guadalajara, México, 2010.

[11] R. E. Shannon. *Simulación de los Sistemas.* Editorial Trillas, 1988.

[12] M. Magnin y O.H. Roux G. Gardey, D. Lime. Romeo: A tool for analyzing time Petri nets. In *17th International Conference on Computer Aided Verification (CAV)*, pages 418–423. Springer, July 2005.

[13] F. Vernadat B. Berthomieu, P.-O. Ribet. The tool tina - construction of abstract state spaces for petri nets and time petri nets. *International Journal of Production Research, Vol. 42, No 14*, pages 2741 – 2756, July 2004.

[14] O.H. Roux M. Magnin, D. Lime. Improved algorithm for computing exact state space of Petri nets with stopwatches. In *Technical report, Institut de Recherche en Communication et Cybernétique de Nantes (IRCCyN)*, 2005.

[15] D. Lime y O.H. Roux M. Magnin. An efficient method for computing exact state space of Petri nets with stopwatches. In *3th International Workshop on Software Model-Checking (SoftMC05)*, pages 33–38. Elsevier, July 2005.

[16] O.H.Roux M. Magnin, D. Lime. Symbolic state space of stopwatch Petri nets with discrete-time semantics. In *29th International Conference on Application and Theory of Petri Nets and other models of concurrency ICATPN*, pages 307–326. Springer, June 2008.

[17] O.H. Roux y F. Vernadat B. Berthomieu, D. Lime. Reachability problems and abstract state spaces for time Petri nets with stopwatches. *Journal of Discrete Event Dynamic Systems Theory and Applications (DEDS)*, pages 133–158, 2007.

[18] D. Lime O.H. Roux. Time Petri nets with inhibitor hyperarcs. formal semantics and state space computation. In *25th International Conference on Application and Theory of Petri Nets ICATPN*, pages 371–390, June 2004.

[19] I. Sommerville. *Ingeniería de software.* Addison wesley, 2005.

[20] A. K. Ríos. *Una Herramienta para el Análisis de Modelos en redes de Petri.* Tesis de Maestría, Centro de Investigación y Estudios Avanzados del IPN, Guadalajara, México, 2010.

[21] I. Jacobson G. Booch, J. Rumbaugh. *El lenguaje unificado de modelado.* Addison wesley, 2002.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

"2010, Año de la Patria, Bicentenario del Inicio de la Independencia
y Centenario del Inicio de la Revolución"

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Simulación de sistemas de eventos discretos temporizados
modelados mediante redes de Petri con cronómetros

del (la) C.

Janeth Gabriela RIVERA AGUILAR

el día 16 de Febrero de 2011.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Antonio Ramírez Treviño
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara