

**UNIVERSIDAD NACIONAL DE
CÓRDOBA**

**FACULTAD DE CIENCIAS EXACTAS
FÍSICAS Y NATURALES**



**Trabajo Práctico Final –
Programación Concurrente**

Profesores: Dr. Ing. Orlando Micolini

Ing. Luis Orlando Ventre

Alumnos: Danilo José Colazo

Esteban Andrés Pérez

ÍNDICE

Introducción	3
Desarrollo	4
Invariantes de plazas	7
Invariantes de transiciones	10
Codificación	11
Test	16
Test Invariantes de plazas	16
Test Invariantes de transiciones	19
Conclusión	22

INTRODUCCIÓN

En el presente trabajo práctico se resuelve el problema de control de un circuito ferroviario. Como dato se propone la red de Petri que modela una planta con 4 estaciones, un vagón, sin barreras. La red se modifica con el fin de modelar la planta requerida y evitar interbloqueos.

La red a modelar estará formada por 4 estaciones (Estación A, Estación B, Estación C y Estación D), una máquina y un vagón. En cada estación los pasajeros podrán subir o bajar al tren o al vagón, no pudiendo descender, en cada estación, los pasajeros que han ascendido en esa.

Los tramos de unión entre las estaciones A y B y las estaciones C y D tienen un paso a nivel. En este paso a nivel controla una barrera para el paso de los vehículos y el tren.

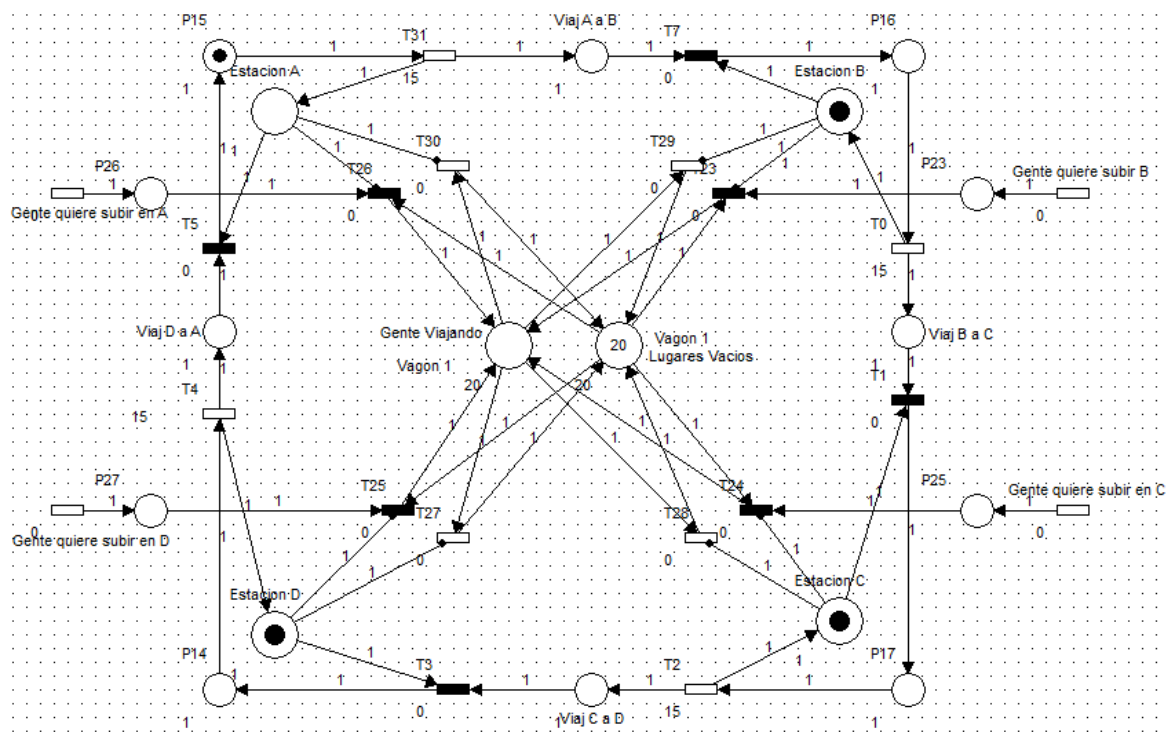


Figura 1: Red de Petri modelo.

Antes de empezar a describir el modelo realizado presentaremos algunas definiciones.

La Red De Petri es una representación matemática o gráfica de un sistema a eventos discretos en el cual se puede describir la topología de un sistema distribuido, paralelo o concurrente.

Una Red De Petri está formada por plazas, transiciones, arcos y tokens que ocupan posiciones dentro de las plazas.

Los arcos conectan una plaza con una transición y viceversa. No puede haber arcos entre plazas ni entre transiciones.

Las plazas contienen un número finito de tokens, también llamados marcas. Estos últimos representa el valor específico de la condición o estado, generalmente se interpretan como la presencia de recursos de un cierto tipo.

Las transiciones representan el conjunto de sucesos cuyo disparo (activación) provoca la modificación de los estados del sistema.

DESAROLLO

Para realizar la red de Petri que represente la planta utilizamos la herramienta PIPE. Esta posee herramientas muy útiles a la hora de analizar una red, buscar sus invariantes, calcular su matriz de incidencia y sobre todo probar que no contiene interbloqueos.

Analizado la red propuesta encontramos que hay una simetría con respecto a las estaciones. Es por esta razón que, si probamos que la red de una sola estación no tiene interbloqueo entonces esta condición se cumple para la toda la red.

Siguiendo este análisis procedemos a realizar la red de una sola estación. Esta se presenta a continuación.

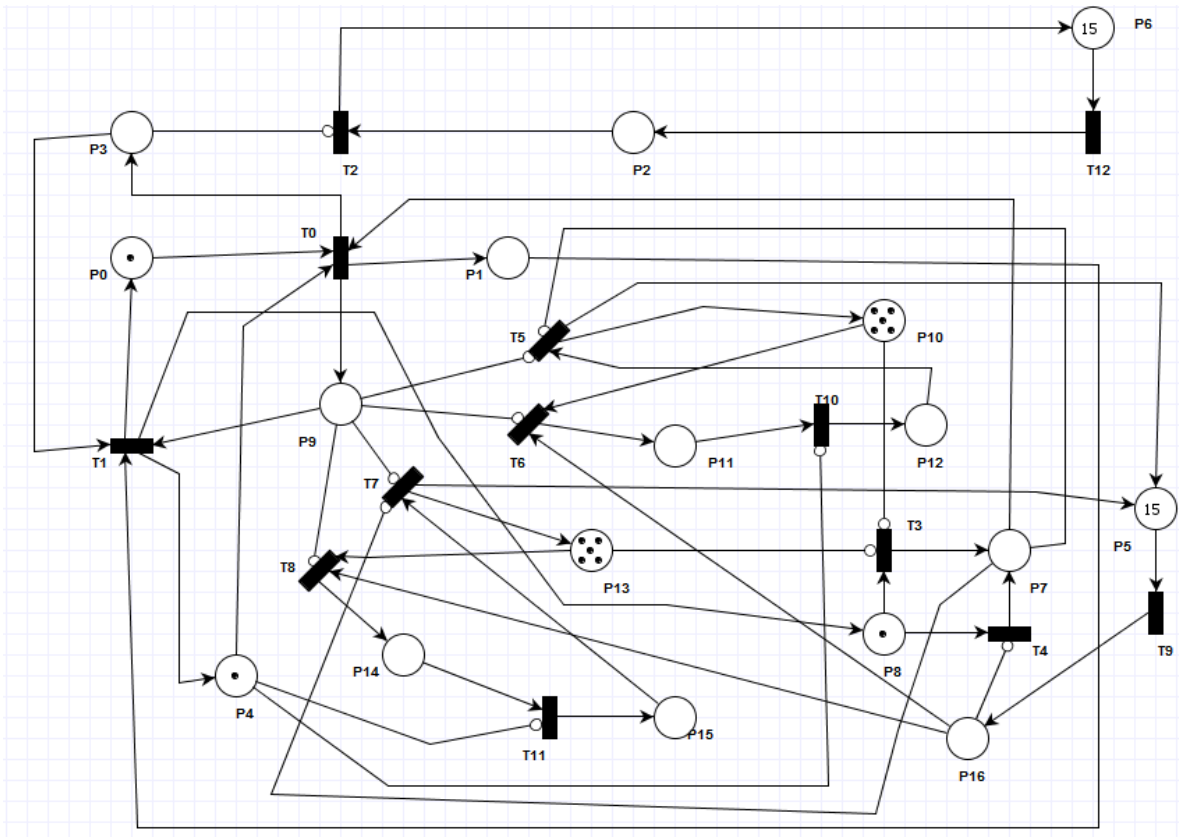


Figura 2: Red de Petri de una sola estación.

Al hacer uso de la herramienta, del software PIPE, llamada “*state spaces analysis*” obtuvimos el siguiente resultado.

Petri net state space analysis results

Bounded	true
Safe	false
Deadlock	false

Figura 3: Resultado mostrado por la herramienta de PIPE *state space analysis*

Como se puede observar de la figura 3 la red no presenta interbloqueos y está limitada. Por lo explicado anteriormente podemos decir que la red completa no tendrá interbloqueos.

A continuación se presentan las tablas de estados y las tablas de transiciones para tener una mejor comprensión de la red.

PLAZAS	DESCRIPCIÓN
P0	Representa al tren junto con el vagón en la estación. Esperando pasajeros.
P1	Representa el viaje del tren.
P9	Controla que el tren y el vagón estén en la estación para permitir el abordaje de pasajeros.
P5	Personas esperando llegar a la estación.
P16	Personas esperando subir al tren o al vagón.
P4	Controla el abordaje de los pasajeros en la estación. Estos no pueden bajar apenas ingresan. Necesitan realizar por lo menos un viaje.
P8	Controla que solamente haya un motivo por el cual el tren deba partir. O no hay gente para subir o no hay lugar.
P7	Contiene el motivo por el cual el tren parte.
P2	Autos esperando en la cola para cruzar el paso nivel.
P3	Barrera del paso nivel. Controla que los autos solamente crucen si el tren no está viajando.
P6	Autos esperando llegar al paso nivel.
P10	Representa los lugares disponibles del vagón.
P13	Representa los lugares disponibles del tren.
P11	Lleva la cuenta de las personas que acaban de abordar el tren.
P14	Lleva la cuenta de las personas que acaban de subir al vagón.
P12	Lleva la cuenta de las personas habilitadas a bajarse del tren.
P15	Lleva la cuenta de las personas habilitadas a bajarse del vagón.

Figura 4: Tabla de estados. Red de una sola estación.

TRANSICIONES	DESCRIPCIÓN
T0	Salida del tren de la estación.
T1	Arribo del tren a la estación.
T2	Autos que cruzan el paso nivel.
T12	Llegada de autos al paso nivel.

T9	Llegada de personas a la estación.
T3	Disparo que se produce por el motivo de no haber lugar disponible.
T4	Disparo producido debido a que no hay pasajeros para subir.
T5	Descenso de pasajeros desde el vagón.
T6	Subida de pasajeros al vagón.
T7	Descenso de pasajeros desde el tren.
T8	Subida de pasajeros al tren.
T10	Pasa al pasajero subido al vagón a la cola para bajar.
T11	Pasa al pasajero subido al tren a la cola para bajar.

Figura 5: Tabla de eventos. Red de una sola estación.

Invariantes de plazas

Para verificar que la red de Petri diseñada cumple con los requerimientos del problema se utilizó la herramienta pipe. Con ella se generaron las ecuaciones de los P-invariantes. Estas nos muestran las restricciones que tiene la red. A continuación procedemos al análisis de dichas ecuaciones.

Ecuación 1

$$M(P0) + M(P1) = 1$$

La ecuación muestra que cuando un tren está en la estación, la gente puede subir a bordo. Caso contrario tiene que esperar en la cola de espera.

Ecuación 2

$$M(P2) + M(P6) = 15$$

Esta ecuación representa al cruce, del paso nivel, de los autos. La cantidad de autos esperando para llegar a al paso nivel debe ser igual a la cantidad de autos esperando cruzar el pazo nivel.

Ecuación 3

$$M(P0) + M(P3) = 1$$

Ecuación que representa la barrera del paso nivel. Cuando el tren está en la estación la barrera no se encuentra baja. Cuando el tren parte para la siguiente estación la barrera se baja.

Ecuación 4

$$M(P5) + M(P11) + M(P12) + M(P14) + M(P15) + M(P16) = 15$$

Ecuación que representa el recorrido del pasajero por la estación. Establece que la cantidad de pasajeros esperando llegar a la estación, más las personas esperando subir, más las personas subidas al tren y al vagón y más las personas pronto a bajar del tren y del vagón debe ser igual a quince.

Ecuación 5

$$M(P10) + M(P11) + M(P12) = 5$$

Ecuación que representa los lugares del vagón. Los lugares vacíos, más los lugares ocupados, más los lugares por liberar debe sumar cinco.

Ecuación 6

$$M(P13) + M(P14) + M(P15) = 5$$

Ecuación que representa los lugares del tren. Los lugares vacíos, más los lugares ocupados, más los lugares por liberar debe sumar cinco.

Ecuación 7

$$M(P1) + M(P4) = 1$$

Ecuación que representa la partida del tren con el permiso para que el pasajero pueda pasar a la cola de listos para bajar. Cuando el tren parte el permiso se activa. Cuando el tren llega a la estación el permiso se desactiva.

Ecuación 8

$$M(P3) + M(P4) = 1$$

Ecuación que relaciona la barrera con el permiso para pasar a la cola de listos para bajar. Cuando la barrera está en el alto, el permiso está activo y viceversa.

Ecuación 9

$$M(P4) + M(P9) = 1$$

Ecuación que relaciona el permiso para pasar a la cola de listos para bajar con el permiso para subir (tren en la estación). Cuando la estación está ocupada el permiso está desactivado y viceversa.

Ecuación 10, 11 y 12

$$M(P7) + M(P8) + M(P9) = 1$$

$$M(P1) + M(P7) + M(P8) = 1$$

$$M(P3) + M(P7) + M(P8) = 1$$

Estas ecuaciones relacionan el motivo por el cual el tren comienza su viaje (P7, P8) con: la condición que requiere que el tren esté en la estación (P9), el viaje del tren (P1) y la barrera (P3).

En la primera se puede deducir que cuando el tren no está en la estación es porque o no había pasajeros o no había lugares disponibles.

En la segunda, si el tren está en viaje es porque subieron todos los pasajeros o se agotaron los lugares vacíos.

En la tercera, si la barrera está en alto es porque el tren partió debido a que no hay más pasajeros a subir o no hay más lugares libres.

Invariantes de transiciones

Es necesario, para corroborar el buen funcionamiento de la red, ver si se cumplen los invariantes de transiciones. Estas son un conjunto de transiciones que ejecutadas en secuencia vuelven a sus estado inicial.

A continuación se muestran y se explican los invariantes de transiciones calculados por la herramienta PIPE.

Invariante 1

$$\{T12, T2\}$$

Este invariante representa la llegada de los autos al paso nivel y el cruce realizado por estos.

Invariante 2

$$\{T3, T0, T1\}$$

Este invariante representa el recorrido del tren debido a que se acabaron los lugares disponibles para pasajeros.

Invariante 3

$$\{T4, T0, T1\}$$

Este invariante representa el recorrido del tren debido a que se subieron todas las personas que se encontraban en la estación.

Invariante 4

$$\{T9, T6, T10, T5\}$$

Esta ecuación representa el viaje de un pasajero en el vagón. Desde su llegada a la estación (T9), hasta que se baja (T5).

Invariante 5

{T9, T8, T11, T7}

Esta ecuación representa el viaje de un pasajero en el vagón. Desde su llegada a la estación (T9), hasta que se baja (T5).

Una vez verificado que la red de una sola estación cumpliera con todos los requerimientos se procede a realizar la red completa que se muestra a continuación.

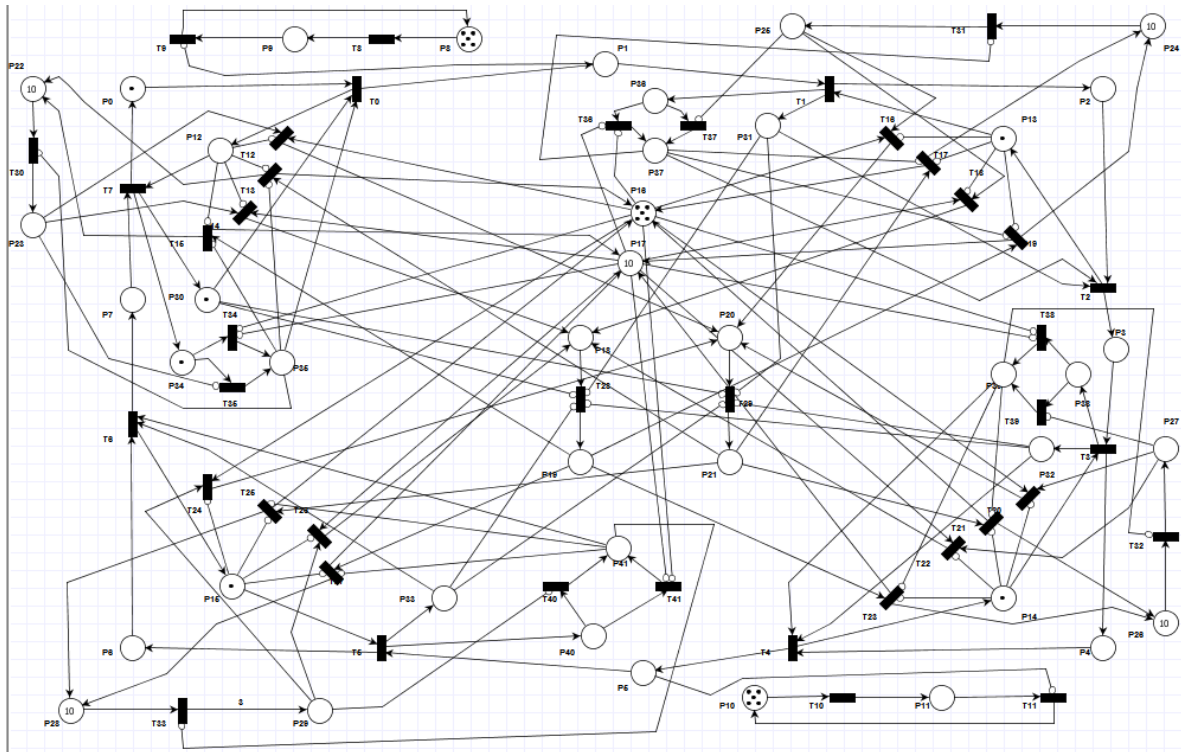


Figura 6: Red de Petri modelo completo con las cuatro estaciones

CODIFICACIÓN

Terminado el modelado de la estación mediante la red de Petri se procede a realizar la codificación del programa mediante el lenguaje de programación Java. Se utiliza para llevar a cabo esta tarea el IDE Eclipse.

Antes de comenzar se presentan el diagrama de clases y de secuencias del programa.

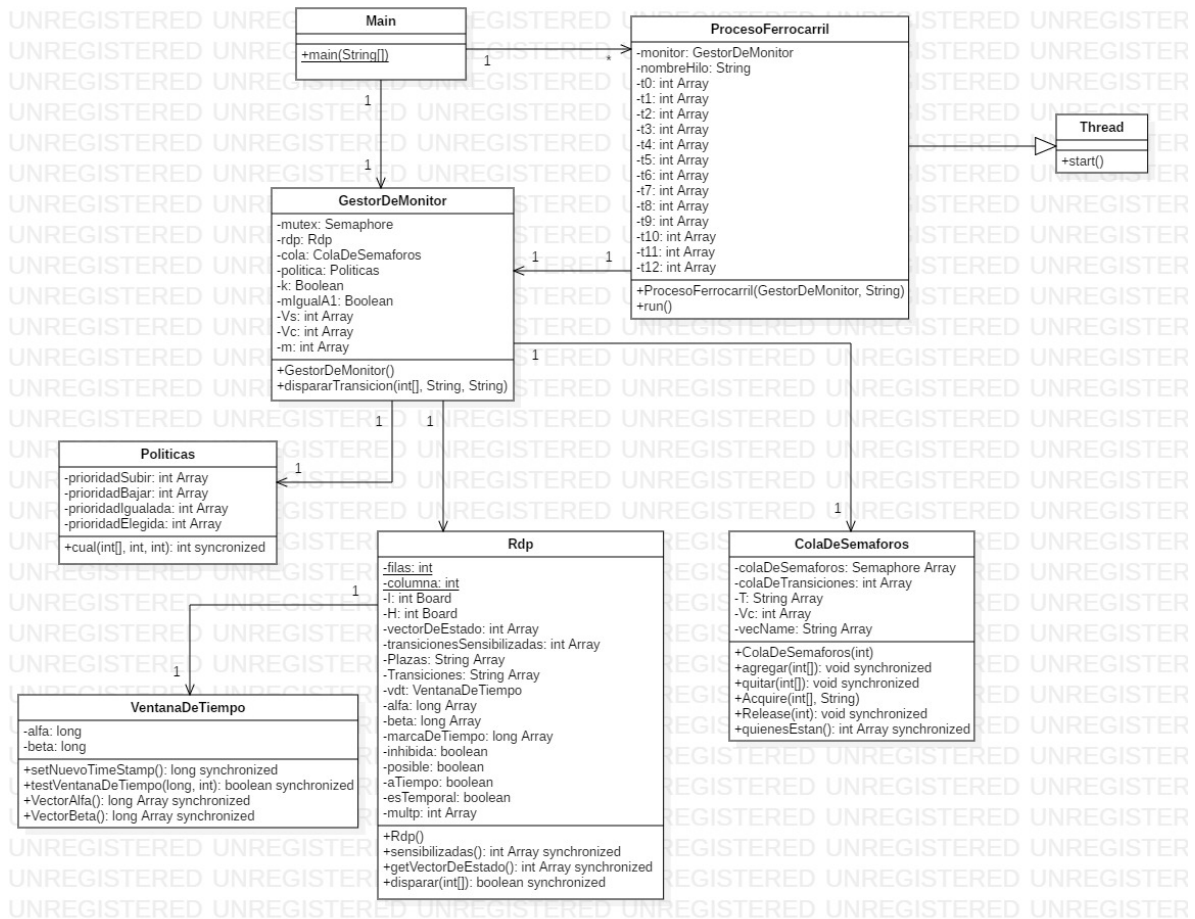


Figura 7: Diagrama de clases



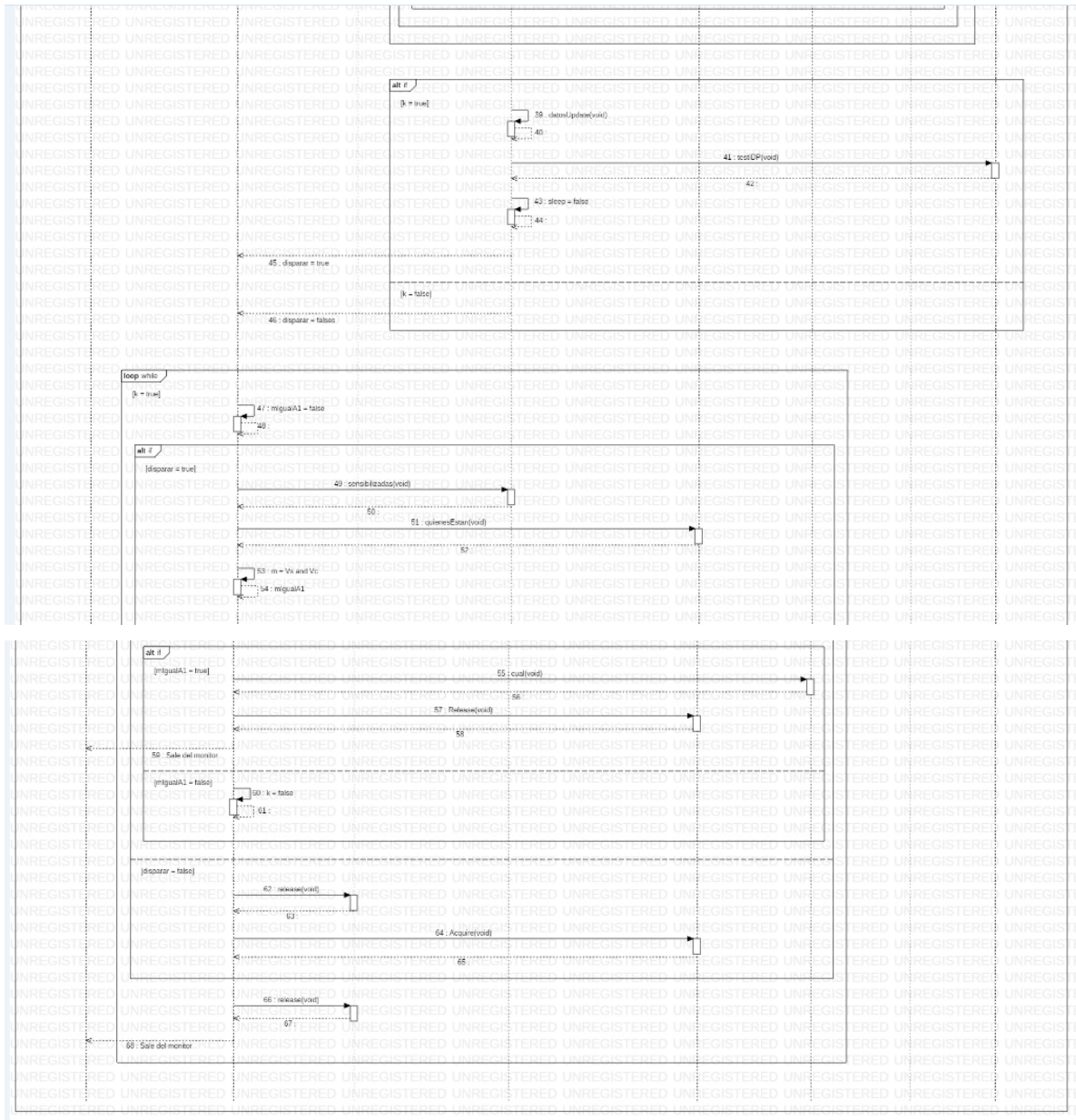


Figura 8: Diagrama de secuencias

Clase Main

Clase encargada de generar de crear un monitor que será compartido por todos los hilos. También se crearán y lanzarán nueve hilos de trabajo.

Clase ProcesoFerrocarril

Clase que hereda de Thread la función run(). Cada hilo tiene una secuencia de disparo que antes de su ejecución debe tomar el monitor.

Clase GestorDeMonitor

Clase que garantiza la ejecución de un único hilo. Esto es así ya que utiliza un objeto semáforo de la clase Semaphore de java. Dicho hilo lleva una determinada transición que para dispararla deberá acceder al monitor.

Si el hilo logra acceder al monitor será el único hilo que se ejecute y los demás hilos solo esperaran a que se libere el monitor.

El hilo tendrá asociada una transición que si está sensibilizada y llega dentro de la ventana de tiempo podrá disparar y acto siguiente liberar el objeto semáforo. Si la transición asociada al hilo esta desensibilizada no se disparará y se encolara en una cola de transiciones no disparadas. Puede pasar que la transición asociada al hilo está sensibilizada y llegue antes de la ventana de tiempo en este caso espera un cierto tiempo y se duerme hasta que le toque su turno.

Clase RdP

Clase que verifica que la transición esté sensibilizada, luego verifica si la transición llega antes o dentro de la ventana de tiempo.

En el caso de que llegue antes de la ventana de tiempo libera el objeto semáforo, espera un tiempo para estar dentro de la ventana de tiempo y se va a dormir.

En el caso de que llegue dentro de la ventana de tiempo dispara y realiza actualizado de vector de estado, vector de sensibilizadas y vector de timeStamp.

Clase ColaDeSemáforos

Esta clase tiene como función alojar a aquellas transiciones que no pudieron dispararse para que luego mediante una determinada política se pueda elegir que transición desencolar primero.

Clase Políticas

Esta clase tiene como función priorizar transiciones. Básicamente la función es seleccionar una transición basándose en la política que se esté utilizando, es decir, que las transiciones tienen asociadas una prioridad y en este caso la transición que tenga mayor prioridad será la elegida.

Clase VentanaDeTiempo

Esta clase tiene la función de indicar si la transición llegó antes o dentro de las variables α y β . Hay que mencionar que el α y el β se los consideraron valores en unidades de tiempo más precisamente en milisegundos.

Test

Test invariantes de plazas

Para verificar los invariante de plazas se muestra a continuación una imagen del vector de estado luego de haber ejecutado un disparo. Luego se comprobará el invariante de plaza de acuerdo a las ecuaciones descriptas anteriormente.

Invariante de plaza 1

$$M(P0) + M(P1) = 1$$

07:26:42:683 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	0	0	0	1	14	15	0	1	0	5	0	0	5	0	0	1

Invariante de plaza 2

$$M(P2) + M(P6) = 15$$

07:26:42:793 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	0	1	0	1	14	14	0	1	0	4	1	0	5	0	0	0

Invariante de plaza 3

$$M(P0) + M(P3) = 1$$

07:26:42:683 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	0	0	0	1	14	15	0	1	0	5	0	0	5	0	0	1

Invariante de plaza 4

$$M(P5) + M(P11) + M(P12) + M(P14) + M(P15) + M(P16) = 15$$

07:26:42:918 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
0	1	1	1	0	13	14	0	0	1	4	0	1	5	0	0	1

Invariante de plaza 5

$$M(P10) + M(P11) + M(P12) = 5$$

07:26:43:020 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	0	1	0	1	13	14	0	1	0	3	1	1	5	0	0	0

Invariante de plaza 6

$$M(P13) + M(P14) + M(P15) = 5$$

07:26:43:020 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	0	1	0	1	13	14	0	1	0	3	1	1	5	0	0	0

Invariante de plaza 7

$$M(P1) + M(P4) = 1$$

07:26:43:020 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	0	1	0	1	13	14	0	1	0	3	1	1	5	0	0	0

Invariante de plaza 8

$$M(P3) + M(P4) = 1$$

07:26:43:020 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	0	1	0	1	13	14	0	1	0	3	1	1	5	0	0	0

Invariante de plaza 9

$$M(P4) + M(P9) = 1$$

07:26:43:020 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	0	1	0	1	13	14	0	1	0	3	1	1	5	0	0	0

Invariante de plaza 10, 11 y 12

$$M(P7) + M(P8) + M(P9) = 1$$

07:26:43:020 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	0	1	0	1	13	14	0	1	0	3	1	1	5	0	0	0

$$M(P1) + M(P7) + M(P8) = 1$$

07:26:43:020 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	0	1	0	1	13	14	0	1	0	3	1	1	5	0	0	0

$$M(P3) + M(P7) + M(P8) = 1$$

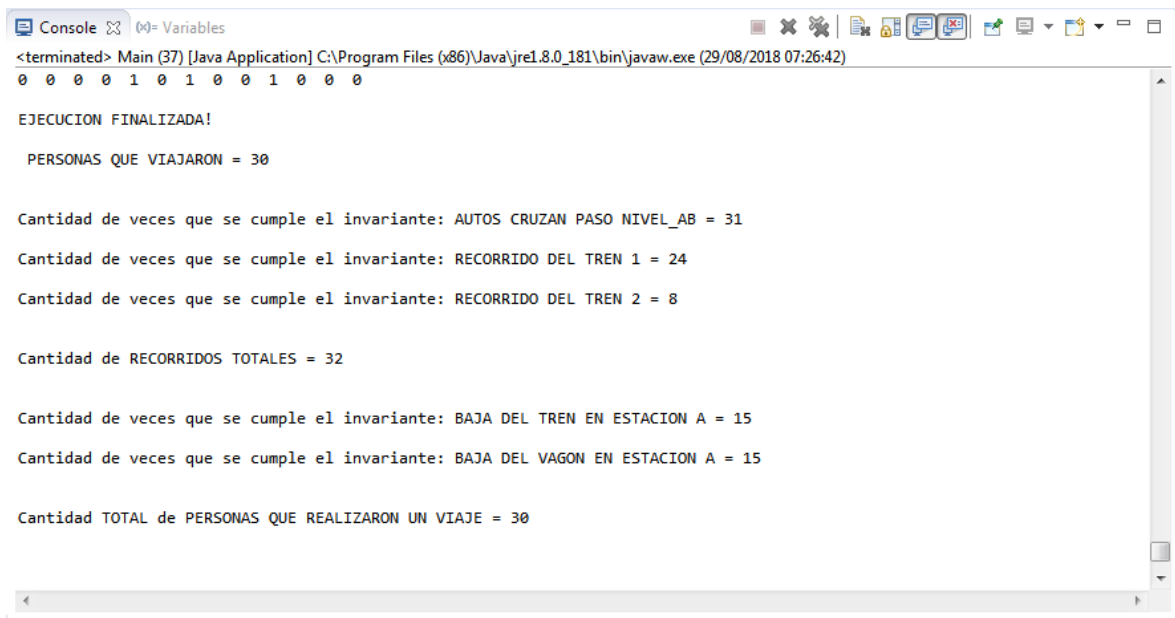
07:26:43:020 - TRANSICION DISPARADA

VECTOR DE ESTADO:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	0	1	0	1	13	14	0	1	0	3	1	1	5	0	0	0

Test Invariantes de transiciones

Para verificar el invariantes de transición se realizó un test en java. A continuación se muestran los resultados de la ejecución del programa con una sola estación y con cuatro estaciones.



```
<terminated> Main (37) [Java Application] C:\Program Files (x86)\Java\jre1.8.0_181\bin\javaw.exe (29/08/2018 07:26:42)
0 0 0 0 1 0 1 0 0 1 0 0 0
EJECUCION FINALIZADA!
PERSONAS QUE VIAJARON = 30
Cantidad de veces que se cumple el invariante: AUTOS CRUZAN PASO NIVEL_AB = 31
Cantidad de veces que se cumple el invariante: RECORRIDO DEL TREN 1 = 24
Cantidad de veces que se cumple el invariante: RECORRIDO DEL TREN 2 = 8
Cantidad de RECORRIDOS TOTALES = 32
Cantidad de veces que se cumple el invariante: BAJA DEL TREN EN ESTACION A = 15
Cantidad de veces que se cumple el invariante: BAJA DEL VAGON EN ESTACION A = 15
Cantidad TOTAL de PERSONAS QUE REALIZARON UN VIAJE = 30
```

Figura 9: Ejecución del programa con una sola estación.

Como puede apreciarse en la figura 9, los invariantes transición se cumplen y la información que se puede sacar es que: en 32 recorridos, del tren, viajaron 30 pasajeros, 15 en el vagón y 15 en el tren. Además hubo 31 cruces de autos mientras el tren estaba en la estación.

```
Console  Variables
<terminated> Main (38) [Java Application] C:\Program Files (x86)\Java\jre1.8.0_181\bin\javaw.exe (29/08/2018 07:30:26)
EJECUCION FINALIZADA!

PERSONAS QUE VIAJARON = 121

Cantidad de veces que se cumple el invariante: AUTOS CRUZAN PASO NIVEL_AB = 64
Cantidad de veces que se cumple el invariante: AUTOS CRUZAN PASO NIVEL_CD = 64
Cantidad de veces que se cumple el invariante 1: RECORRIDO DEL TREN = 1
Cantidad de veces que se cumple el invariante 2: RECORRIDO DEL TREN = 0
Cantidad de veces que se cumple el invariante 3: RECORRIDO DEL TREN = 1
Cantidad de veces que se cumple el invariante 4: RECORRIDO DEL TREN = 0
Cantidad de veces que se cumple el invariante 5: RECORRIDO DEL TREN = 1
Cantidad de veces que se cumple el invariante 6: RECORRIDO DEL TREN = 0
Cantidad de veces que se cumple el invariante 7: RECORRIDO DEL TREN = 1
Cantidad de veces que se cumple el invariante 8: RECORRIDO DEL TREN = 0
Cantidad de veces que se cumple el invariante 9: RECORRIDO DEL TREN = 14
Cantidad de veces que se cumple el invariante 10: RECORRIDO DEL TREN = 0
Cantidad de veces que se cumple el invariante 11: RECORRIDO DEL TREN = 1
Cantidad de veces que se cumple el invariante 12: RECORRIDO DEL TREN = 0
Cantidad de veces que se cumple el invariante 13: RECORRIDO DEL TREN = 1
Cantidad de veces que se cumple el invariante 14: RECORRIDO DEL TREN = 0
Cantidad de veces que se cumple el invariante 15: RECORRIDO DEL TREN = 1
```

```
Cantidad de veces que se cumple el invariante 16: RECORRIDO DEL TREN = 0

Cantidad de RECORRIDOS TOTALES = 19

Cantidad de veces que se cumple el invariante: BAJA DEL TREN EN ESTACION A = 15
Cantidad de veces que se cumple el invariante: BAJA DEL VAGON EN ESTACION A = 15
Cantidad de veces que se cumple el invariante: BAJA DEL TREN EN ESTACION B = 15
Cantidad de veces que se cumple el invariante: BAJA DEL VAGON EN ESTACION B = 15
Cantidad de veces que se cumple el invariante: BAJA DEL TREN EN ESTACION C = 16
Cantidad de veces que se cumple el invariante: BAJA DEL VAGON EN ESTACION C = 15
Cantidad de veces que se cumple el invariante: BAJA DEL TREN EN ESTACION D = 15
Cantidad de veces que se cumple el invariante: BAJA DEL VAGON EN ESTACION D = 15

Cantidad TOTAL de PERSONAS QUE REALIZARON UN VIAJE = 121
```

Figura 10: Ejecución del programa con cuatro estaciones.

Como se aprecia en la Figura 10, se cumplen los invariantes de transición y la información que se puede sacar es que en 19 recorridos viajaron 121 pasajeros, 30 se bajaron del tren y del vagón en la estación “A”, 30 se bajaron del tren y del vagón en la estación “B”, 31 se bajaron del tren y del vagón en la estación “C” y 30 se bajaron del tren y del vagón en la estación “D”. Además se produjo el cruce de 64 autos tanto en el paso nivel “A_B” como en el paso nivel “C_D”.

Es importante aclarar que los invariantes del recorrido del tren son 16 debido a que a medida que llega a una estación hay dos posibles caminos. Esto es así debido a que en cada estación el tren únicamente partirá de viaje cuando no haya gente para subir o no haya lugar disponible.

Conclusión

En el transcurso de la resolución del problema comprendimos la utilidad de las redes de Petri para modelar diversos sistemas como fue el caso del circuito ferroviario.

Una vez realizada la Red de Petri simplificada, analizamos que eventos se producen y que estados o actividades están presentes en el sistema. A partir de esto se realizó unas tablas indicativas con esta información.

Un problema que tuvimos fue la asignación de hilos ya que en algunos casos las relaciones entre ciertas transiciones hacían imposible el compartir un hilo por ello es que en la red simplificada quedaron 9 hilos y en la red completa 31 hilos.

Una vez finalizado todo el armado de la Red de Petri se realizó el código en JAVA ECLIPSE, en donde la actividad principal fue la de armar un Monitor y lograr su correcto funcionamiento.

Finalizado la codificación se realizaron los test de invariantes de plazas y transiciones logrando resultados satisfactorios.

Como conclusión final podemos decir que se han adquiridos los conocimientos básicos de las redes de Petri y se ganó experiencia en cuanto al planteo y resolución de problemas difíciles.