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

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.

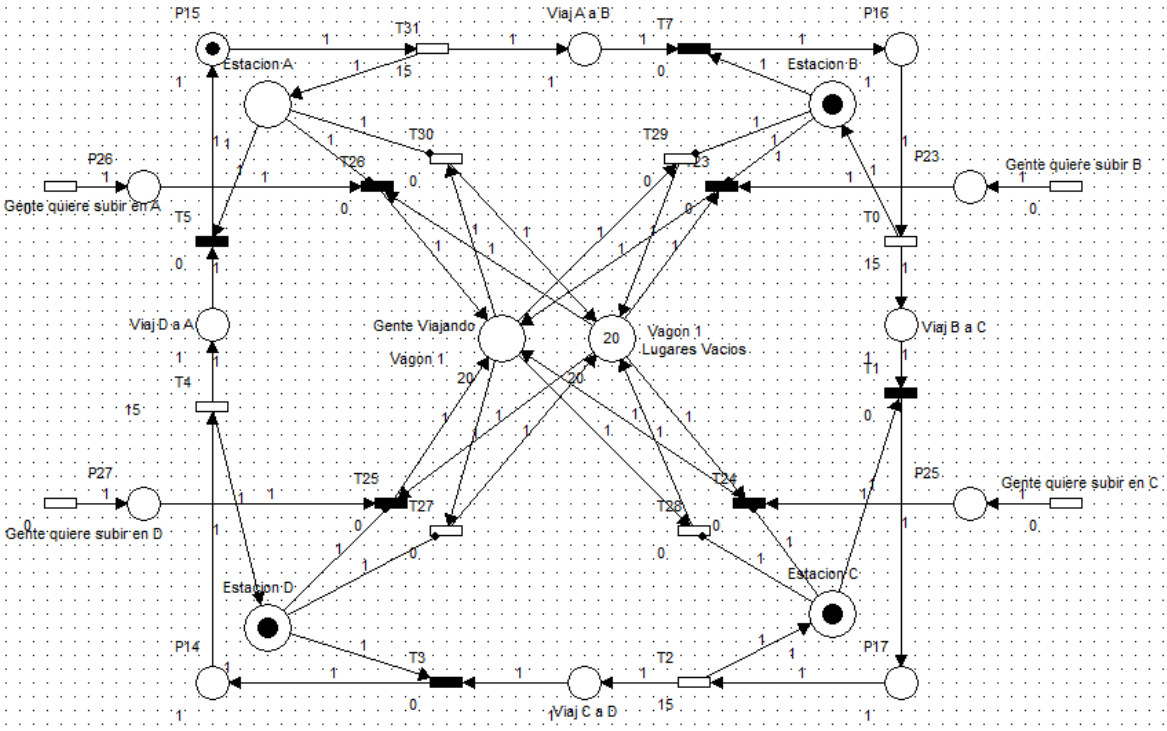


Figura 1: Red de Petri modelo.

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

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. Estos últimos 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 se utiliza el software PIPE. Este posee herramientas muy útiles a la hora de analizar una red, buscar invariantes, calcular la matriz de incidencia y sobre todo probar que está libre de interbloqueos.

Analizando la red propuesta encontramos que hay una simetría con respecto a las estaciones. Es por esta razón que, si probamos simular la red con una sola estación y encontramos que no tiene interbloqueo entonces esta condición se cumple para la red completa (con cuatro estaciones).

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

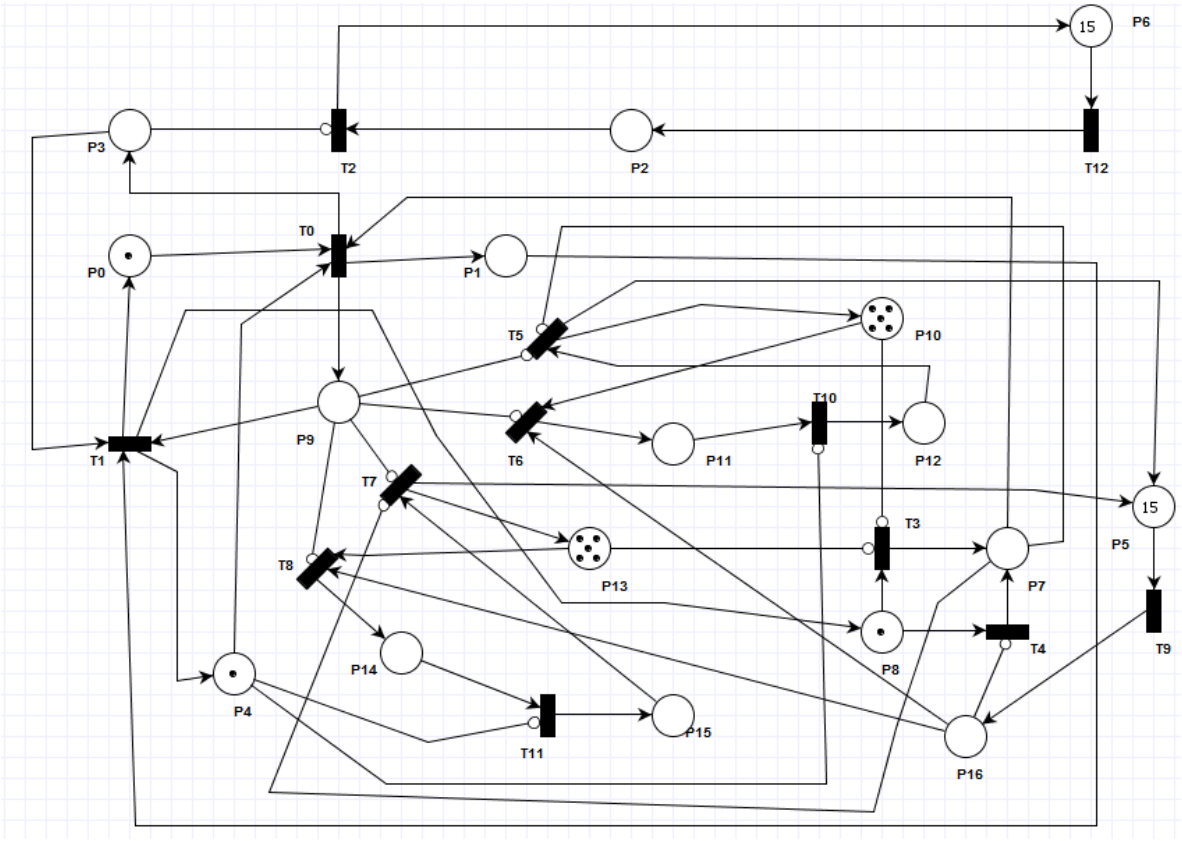


Figura 2: Red de Petri de una sola estación.

Al hacer uso de la herramienta *"state spaces analysis"*, del software PIPE, se obtuvo el siguiente resultado.

Petri net state space analysis results

Bounded	true
Safe	false
Deadlock	false

Figura 3: Resultado mostrado por la herramienta *state space analysys*

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 eventos 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 abordo de pasajeros.
P5	Personas esperando llegar a la estación.
P16	Personas esperando subir al tren o al vagón.
P4	Controla el abordo 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	Motivo por el cual el tren parte de la estación (no hay gente o no hay lugar).
P2	Autos esperando en la cola para cruzar el paso nivel.
P3	Barrera del paso nivel. Controla que los autos solamente crucen cuando el tren esté en la estación.
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.

Tabla 1: Tabla de estados. Red de una sola estación.

TRANSICIONES	DECRIPCIÓN
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 que viajaron en el vagón.
T6	Subida de pasajeros al vagón.
T7	Descenso de pasajeros que viajaron en 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.

Tabla 2: Tabla de eventos. Red de una sola estación.

Invariantes de plazas

Para que la red de Petri diseñada cumpla con los requerimientos del problema se deben cumplir los invariantes de plazas. Estas nos muestran las restricciones que tiene la red.

Para empezar este análisis se generan las ecuaciones de P-invariantes mediante el software PIPE. A continuación se procede al análisis de dichas ecuaciones.

Ecuación 1

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

La ecuación relaciona al conjunto de tren y vagón, ubicados en la estación, y el viaje de estos hacia otra estación. Esta muestra que el tren y vagón pueden estar en la estación o en viaje. La suma de estas variables debe ser igual a uno para que se cumpla el invariante.

Ecuación 2

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

Esta ecuación representa al cruce, del paso nivel, que realizan los autos. Estos pueden estar esperando para llegar al paso nivel o pueden estar esperando para cruzar el paso nivel. La suma de estas variables debe ser igual a quince para que se cumpla el invariante.

Ecuación 3

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

Ecuación que representa la barrera del paso nivel relacionado con el conjunto de tren y vagón ubicados en la estación. Cuando el tren, junto con el vagón, se encuentra en la estación la barrera está levantada. Cuando parten hacia otra estación la barrera se baja. La suma de estas variables debe ser igual a uno para que se cumpla el invariante.

Ecuación 4

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

Ecuación que representa la ubicación de las personas. Establece que estas pueden estar esperando llegar a la estación, pueden haber abordado el tren o el vagón, pueden haber pasado a la cola de listos para bajar del tren o del vagón, o pueden estar esperando subir (al tren o al vagón). La suma de estas variables debe ser igual a quince para que se cumpla el invariante.

Ecuación 5

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

Ecuación que representa los lugares del vagón. Estos son: lugares vacíos, lugares ocupados y lugares por liberar. La suma de estas tres variables debe ser cinco para que el invariante se cumpla.

Ecuación 6

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

Ecuación que representa los lugares del tren. Estos son: lugares vacíos, lugares ocupados y lugares por liberar. La suma de estas tres variables debe ser cinco para que el invariante se cumpla.

Ecuación 7

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

Ecuación que representa la relación entre el viaje del tren-vagón y el permiso del pasajero para que puedan pasar a la cola de listos para bajar.

Cuando el tren-vagón parte de la estación las personas tienen permitido pasar a la cola de listos para bajar. Caso contrario el permiso es negado. La suma de estas variables debe ser igual a uno para que se cumpla el invariante.

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. En caso contrario el permiso es negado. La suma de estas variables debe ser igual a uno para que se cumpla el invariante.

Ecuación 9

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

Ecuación que relaciona al permiso para pasar a la cola de listos para bajar con el conjunto de tren y vagón, ubicados en la estación.

Cuando la estación está ocupada (por el tren y el vagón) el permiso está activo. En caso contrario el permiso es negado. La suma de estas variables debe ser igual a uno para que se cumpla el invariante.

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: el tren y vagón 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.

La suma de estas variables debe ser igual a uno para que se cumpla el invariante.

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 permiten volver al 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).

Para determinar la cantidad de hilos que se asignarán a las distintas transiciones se realiza lo siguiente. Se buscan aquellas transiciones que formen una secuencia. Hay que tener en cuenta que cada hilo tendrá su secuencia de transiciones y que otro hilo no puede tener involucrada alguna transición de la anterior, pero si puede un hilo tener la misma secuencia de transición que otro hilo. De esta forma se llegó a la conclusión de que el programa contará con nueve hilos de trabajo y se presentan a continuación.

HILOS	DESCRIPCIÓN
Personas	Maneja la transición encargada de hacer llegar gente a la estación.
SUBEN_TREN	Maneja la transición encargada de subir personas al tren.
SUBEN_VAGÓN	Maneja la transición encargada de subir personas al vagón.
BAJAN_TREN	Maneja la transición encargada de pasar las personas a la cola listo para bajar, y la transición que realiza el descenso de la persona del

	tren.
BAJAN_VAGÓN	Maneja la transición encargada de pasar las personas a la cola listo para bajar, y la transición que realiza el descenso de la persona del vagón.
NoGente	Maneja la transición encargada de activar el permiso de arranque cuando no haya gente para subir.
Recorrido	Maneja la transición encargada del recorrido del tren-vagón
NoLugar	Maneja la transición encargada de activar el permiso de arranque cuando no haya lugar disponible.
Autos	Maneja la transición encargada del cruce de los autos por el paso nivel.

Tabla 3: Hilos de trabajo del programa de una estación.

Finalizada la asignación de hilos se procede a realizar el cálculo de la matriz de incidencia, matriz inhibidora, el vector de marcado inicial y el de transiciones sensibilizadas inicialmente.

Combined incidence matrix *I*

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
P0	-1	1	0	0	0	0	0	0	0	0	0	0	0
P1	1	-1	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	-1	0	0	0	0	0	0	0	0	0	1
P3	1	-1	0	0	0	0	0	0	0	0	0	0	0
P4	-1	1	0	0	0	0	0	0	0	0	0	0	0
P5	0	0	0	0	0	1	0	1	0	-1	0	0	0
P6	0	0	1	0	0	0	0	0	0	0	0	0	-1
P7	-1	0	0	1	1	0	0	0	0	0	0	0	0
P8	0	1	0	-1	-1	0	0	0	0	0	0	0	0
P9	1	-1	0	0	0	0	0	0	0	0	0	0	0
P10	0	0	0	0	0	1	-1	0	0	0	0	0	0
P11	0	0	0	0	0	0	1	0	0	0	-1	0	0
P12	0	0	0	0	0	-1	0	0	0	0	1	0	0
P13	0	0	0	0	0	0	0	1	-1	0	0	0	0
P14	0	0	0	0	0	0	0	0	1	0	0	-1	0
P15	0	0	0	0	0	0	0	-1	0	0	0	1	0
P16	0	0	0	0	0	0	-1	0	-1	1	0	0	0

Figura 4: Matriz de incidencia. Red de una estación

Inhibition matrix *H*

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
P0	0	0	0	0	0	0	0	0	0	0	0	0	0
P1	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	0	0	0	0	0	0	0	0	0	0
P3	0	0	1	0	0	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0	1	1	0
P5	0	0	0	0	0	0	0	0	0	0	0	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0
P7	0	0	0	0	0	1	0	1	0	0	0	0	0
P8	0	0	0	0	0	0	0	0	0	0	0	0	0
P9	0	0	0	0	0	1	1	1	1	0	0	0	0
P10	0	0	0	1	0	0	0	0	0	0	0	0	0
P11	0	0	0	0	0	0	0	0	0	0	0	0	0
P12	0	0	0	0	0	0	0	0	0	0	0	0	0
P13	0	0	0	1	0	0	0	0	0	0	0	0	0
P14	0	0	0	0	0	0	0	0	0	0	0	0	0
P15	0	0	0	0	0	0	0	0	0	0	0	0	0
P16	0	0	0	0	1	0	0	0	0	0	0	0	0

Figura 5: Matriz inhibidora. Red de una estación

Marking

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

Enabled transitions

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
no	no	no	no	yes	no	no	no	no	yes	no	no	yes

Figura 6: vector de marcado inicial y de transiciones sensibilizadas inicialmente. Red de una estación

El comportamiento dinámico de la Red De Petri está determinado por la habilitación o sensibilizado y disparo de sus transiciones. Las reglas de ejecución de la red es: la red se ejecuta por los disparos de sus transiciones. Para que una transición pueda ser disparada debe estar habilitada. Una transición está habilitada si cada una de las plazas de entrada a la transición tiene al menos la cantidad de tokens indicados en el peso, que está en el arco que une la plaza con la transición. Por lo tanto la red progresa siguiendo esta expresión:

$$M_{K+1} = M_K + Ix\,\mathrm{t}$$

Figura 6: Ecuación de estado.

La expresión de la Figura 6 describe el comportamiento dinámico de la red.

En nuestro caso hubo que agregar la matriz inhibidora lo que cambia un poco la ecuación de estado. Sin embargo por simplicidad se decidió utilizar la expresión de la Figura 6. Para ello se utilizó la matriz inhibidora para determinar si la transición a disparar estaba inhibida de acuerdo al marcado actual.

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.

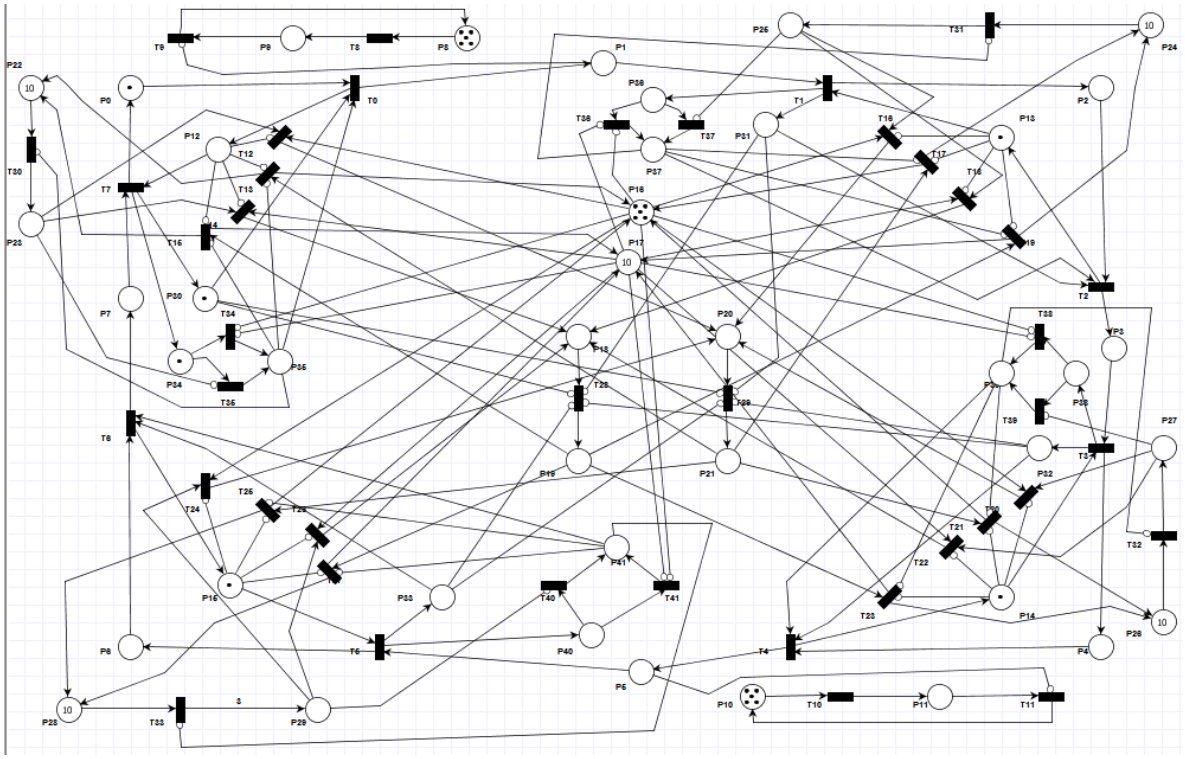


Figura 7: Red de Petri modelo completo con las cuatro estaciones

Para su correcto funcionamiento se necesitaron asignar treinta y un hilos de trabajo. Estos son los mismos mencionados en “Tabla 3” anterior solo que repartido para 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 Eclipce.

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

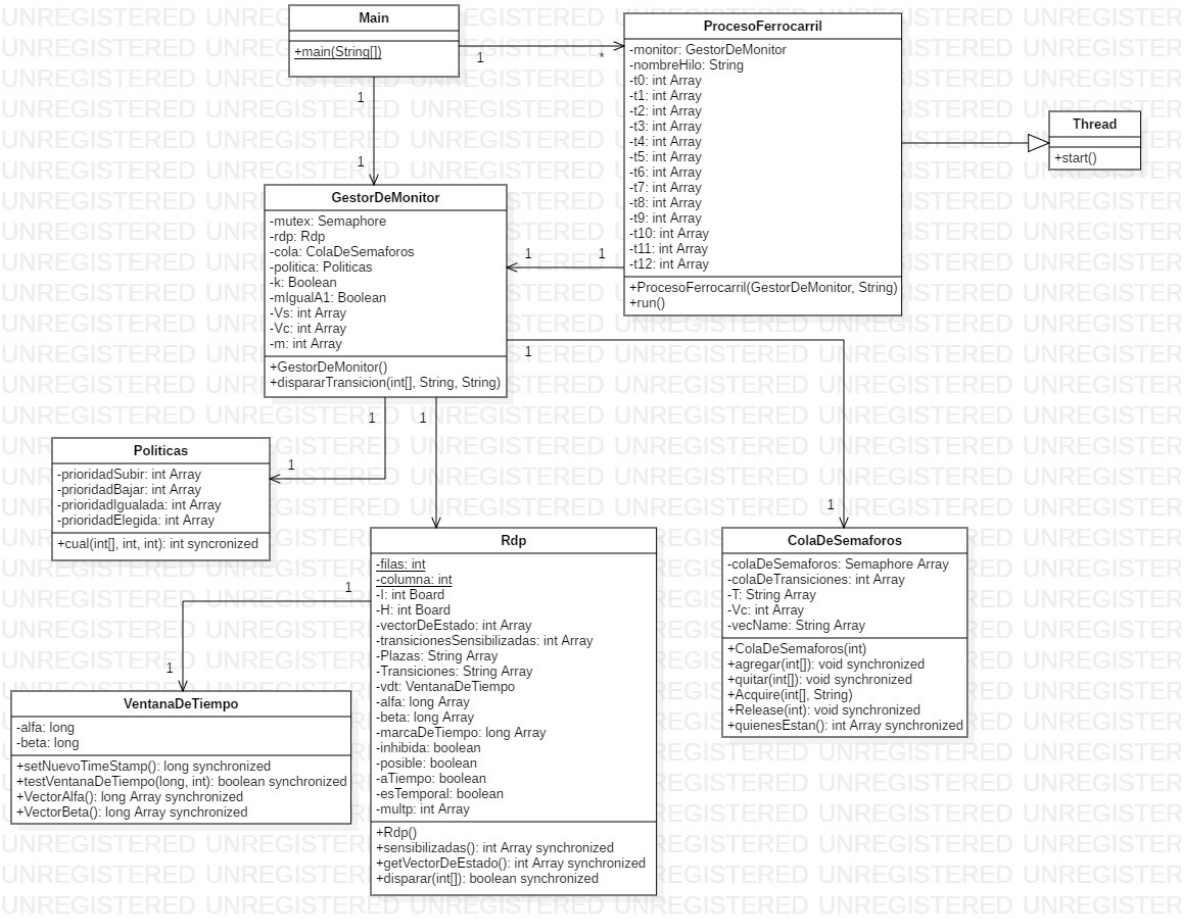
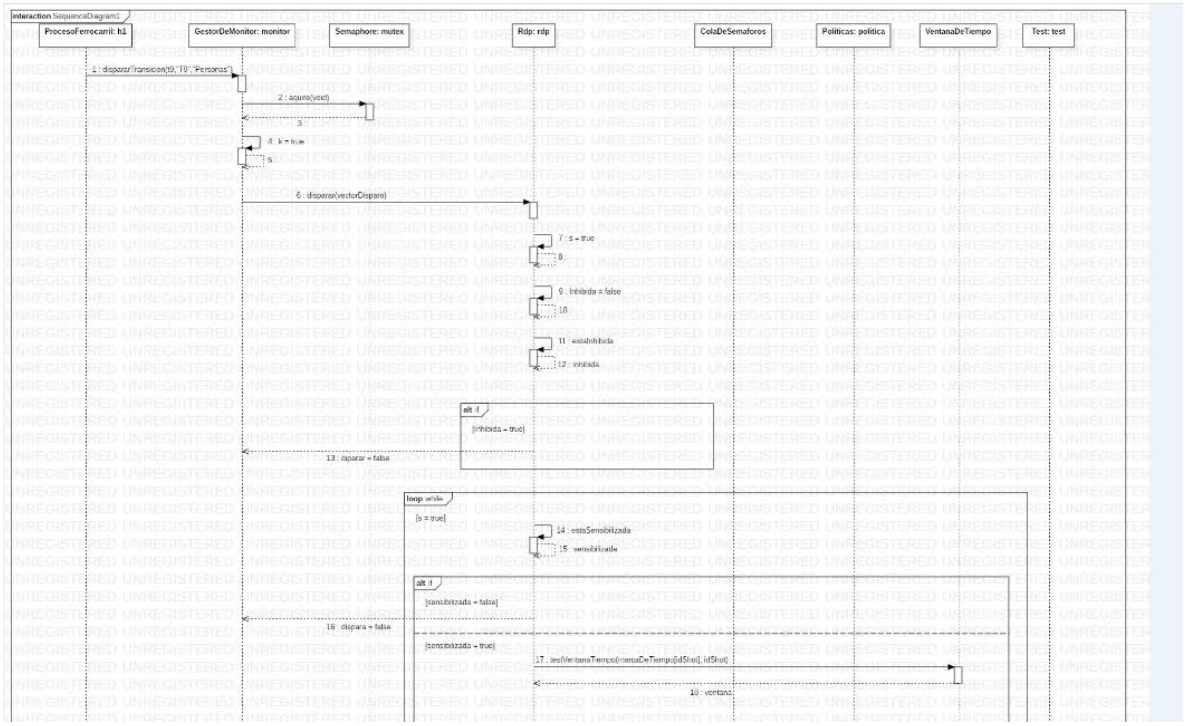
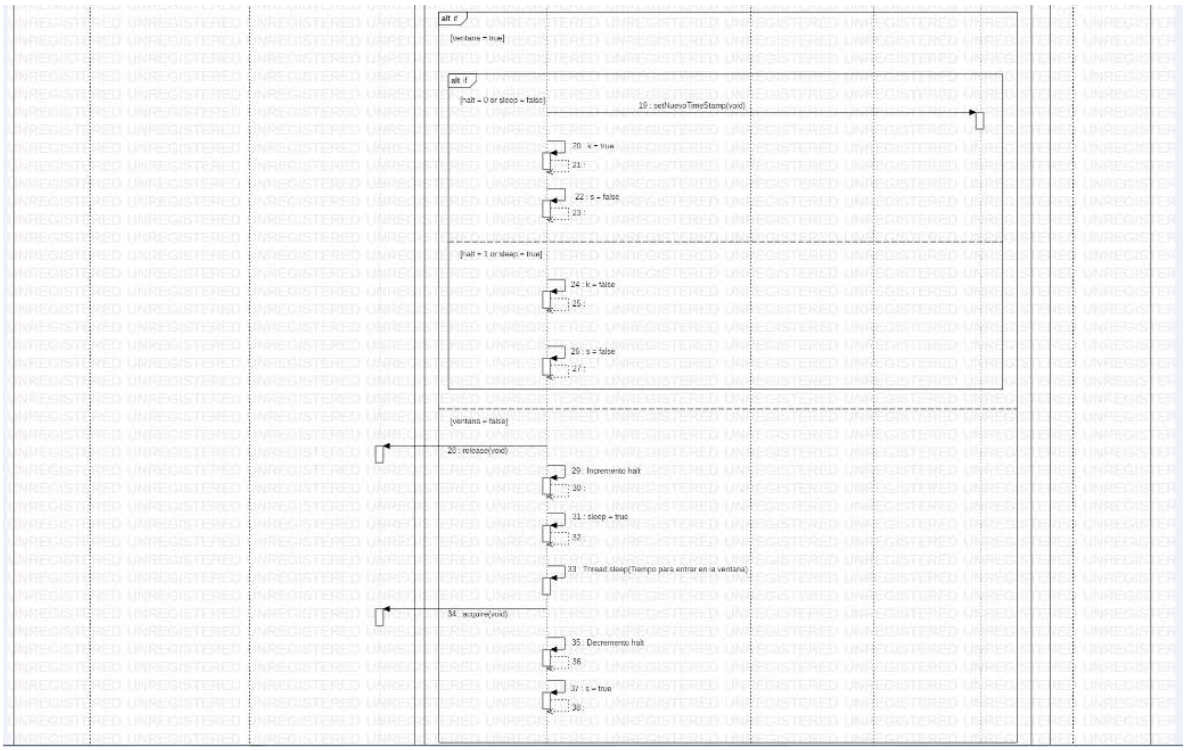


Figura 8: Diagrama de clases


interaction SecuenciaDiagram1
ProcesoFerrocarril: h1
GestorDeMonitor: monitor
Semaphore: mutex
Rdp: rdp
ColaDeSemaforos
Politicas: politica
VentanaDeTiempo
Test: test
1 : disparar(Transicion(t9, T9), "Personas")
2 : acquire(void)
3
4 : k = true
5
6 : disparar(vectorDisparo)
7 : s = true
8
9 : inhibida = false
10
11 : estaInhibida
12 : inhibida
alt if
[inhibida = true]
13 : disparar = false
loop while
[s = true]
14 : estaSensibilizada
15 : sensibilizada
alt if
[sensibilizada = false]
16 : dispara = false
[sensibilizada = true]
17 : sesVentanaTiempo(marcaDeTiempo[idShot], idShot)
18 : ventana


alt if
[ventana = true]
alt if
[halt = 0 or sleep = false]
19 : setNuevoTimeStamp(void)
20 : k = true
21
22 : s = false
23
[halt = 1 or sleep = true]
24 : k = false
25
26 : s = false
27
[ventana = false]
28 : release(void)
29 : Incremento halt
30
31 : sleep = true
32
33 : Thread.sleep(Tiempo para entrar en la ventana)
34 : acquire(void)
35 : Decremento halt
36
37 : s = true
38

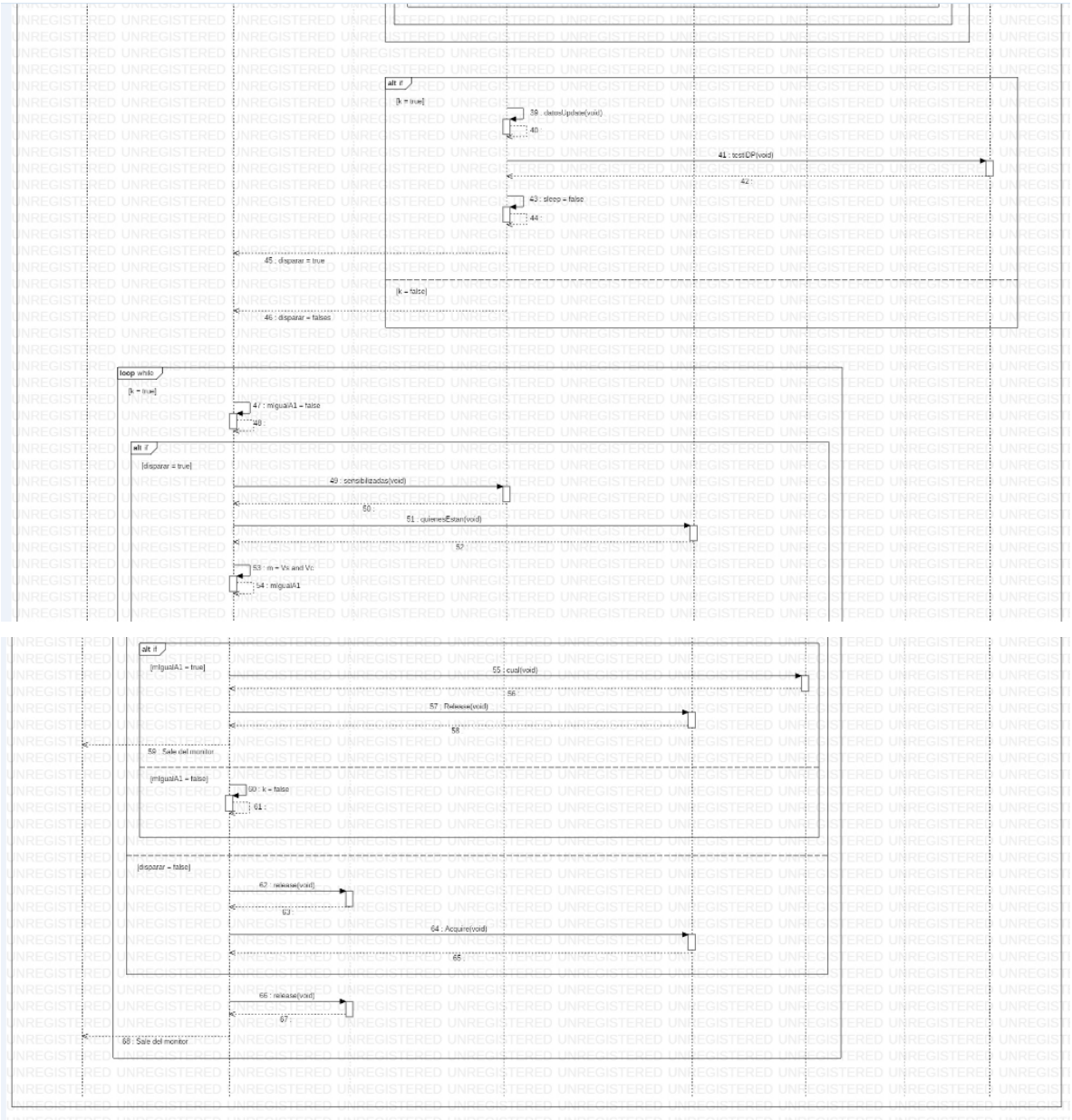


Figura 9: Diagrama de secuencias

Clase Main

Clase encargada 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 tomar el monitor. Si lo logra será el único hilo que se ejecute y los demás hilos solo esperaran a que se libere el monitor.

Si la transición está sensibilizada y llega dentro de la ventana de tiempo podrá disparar y acto siguiente liberar el monitor. Por el contrario, 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 esté sensibilizada pero llegue antes de la ventana de tiempo. En este caso espera el tiempo que le falta para entrar en la ventana 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 duerme.

En el caso de que llegue dentro de la ventana de tiempo, la transición se dispara y se realiza una actualización del 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 que tiene dicha clase es seleccionar una transición basándose en la política que se esté utilizando. Esto quiere decir que las transiciones tienen asociadas una prioridad y en este caso la transición que tenga mayor prioridad será la elegida.

En nuestro caso se implementaron tres políticas: prioridad para subir, prioridad para bajar y prioridad igualada. La lógica es la siguiente, cuando la cantidad de lugares disponibles supere el 60 % del total de lugares entonces se asignará la política prioridad para subir. Si la cantidad de lugares disponibles es menor al 40 % del total de lugares entonces se asignará la política prioridad para bajar. Por último, cuando la cantidad de lugares disponibles esté entre el 40 % y el 60 % del total de lugares entonces se asignará la política prioridad igualada.

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

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

Transiciones implicadas T0, T1.

T0: Dispara.

T1: No dispara.

```
07:26:42:793 - MUTEX .acquire-> Hilo: Arranque - Tr: T0_____________________________

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
0   1   1   1   0   14  14  0   0   1   4    1    0    5    0    0    0
------------------------
```

Invariante de plaza 2

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

Transiciones implicadas T12, T2.

T12: Dispara.

T2: No dispara.

```
07:26:42:684 - MUTEX .acquire-> Hilo: Autos - Tr: T12_____________________________

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

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

Transiciones implicadas T0, T1.

T0: Dispara.

T1: No dispara.

```
07:26:42:793 - MUTEX .acquire-> Hilo: Arranque - Tr: T0_____________________________

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
0   1   1   1   0   14  14  0   0   1   4    1    0    5    0    0    0
-------------------------
```

Invariante de plaza 4

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

Transiciones implicadas T9, T6, T10, T5, T8, T11, T7.

T9: Dispara.

T6: No dispara.

T10: No dispara.

T5: No dispara.

T8: No dispara.

T11: No dispara.

T7: No dispara.

```
07:26:42:644 - MUTEX .acquire-> Hilo: Personas - Tr: T9_____________________________

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 5

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

Transiciones implicadas T6, T10, T5.

T6: Dispara.

T10: No dispara.

T5: No dispara.

```
07:26:42:684 - MUTEX .acquire-> Hilo: SUBEN - Tr: T6_____________________________

07:26:42:684 - 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   4    1    0    5    0    0    0
-------------------------
```

Invariante de plaza 6

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

Transiciones implicadas T8, T11, T7.

T8: Dispara.

T11: Dispara.

T7: No dispara.

```
07:26:45:309 - MUTEX .acquire-> Hilo: BAJAN - Tr: T11_____________________________

07:26:45:309 - TRANSICION DISPARADA
VECTOR DE ESTADO:
P0  P1  P2  P3  P4  P5  P6  P7  P8  P9  P10  P11  P12  P13  P14  P15  P16
0   1   0   1   0   6   15  0   0   1   0    4    1    1    3    1    0
-------------------------
```

Invariante de plaza 7

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

Transiciones implicadas T0, T1.

T0: Dispara.

T1: No dispara.

```
07:26:42:793 - MUTEX .acquire-> Hilo: Arranque - Tr: T0_____________________________

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
0   1   1   1   0   14  14  0   0   1   4    1    0    5    0    0    0
-------------------------
```

Invariante de plaza 8

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

Transiciones implicadas T0, T1.

T0: Dispara.

T1: No dispara.

```
07:26:42:793 - MUTEX .acquire-> Hilo: Arranque - Tr: T0_____________________________

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
0   1   1   1   0   14  14  0   0   1   4    1    0    5    0    0    0
-------------------------
```

Invariante de plaza 9

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

Transiciones implicadas T0, T1.

T0: Dispara.

T1: No dispara.

```
07:26:42:793 - MUTEX .acquire-> Hilo: Arranque - Tr: T0_____________________________

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
0   1   1   1   0   14  14  0   0   1   4    1    0    5    0    0    0
-------------------------
```

Invariante de plaza 10, 11 y 12

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

Transiciones implicadas T0, T1.

T0: Dispara.

T1: No dispara.

T3: No dispara.

T4: No dispara.

```
07:26:42:793 - MUTEX .acquire-> Hilo: Arranque - Tr: T0_____________________________

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
0   1   1   1   0   14  14  0   0   1   4    1    0    5    0    0    0
------------------------
```

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

Transiciones implicadas T0, T1.

T0: Dispara.

T1: No dispara.

T3: No dispara.

T4: No dispara.

```
07:26:42:793 - MUTEX .acquire-> Hilo: Arranque - Tr: T0_____________________________

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
0   1   1   1   0   14  14  0   0   1   4    1    0    5    0    0    0
------------------------
```

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

Transiciones implicadas T0, T1.

T0: Dispara.

T1: No dispara.

T3: No dispara.

T4: No dispara.

```
07:26:42:793 - MUTEX .acquire-> Hilo: Arranque - Tr: T0_____________________________

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
0   1   1   1   0   14  14  0   0   1   4    1    0    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.

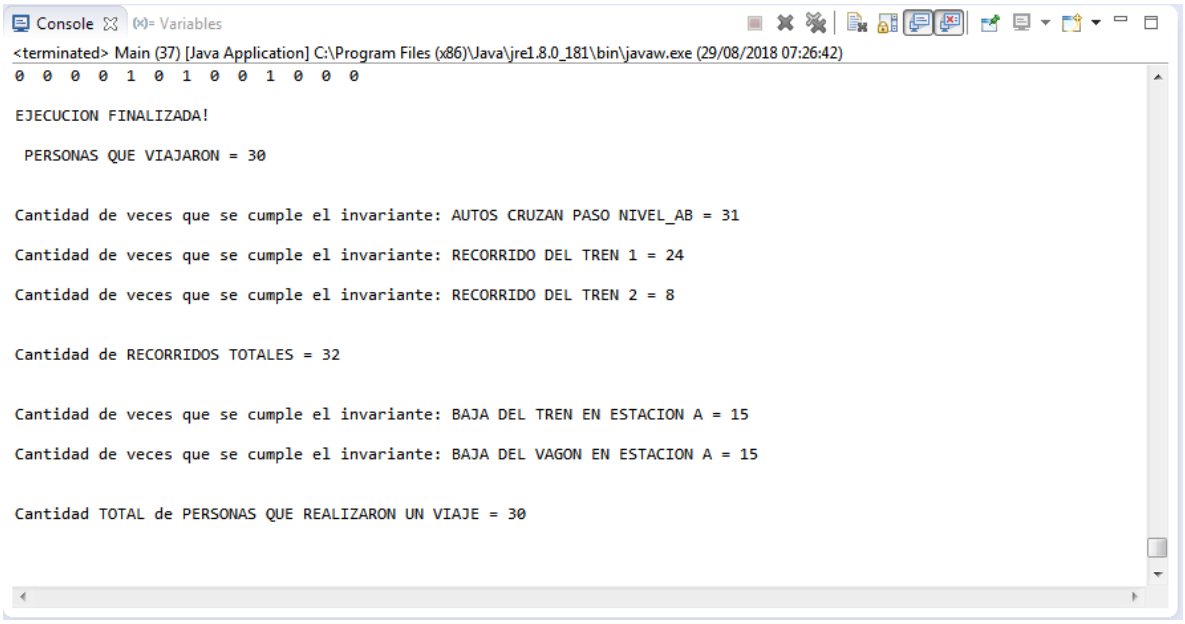


Figura 10: Ejecución del programa con una sola estación.

Como puede apreciarse en la Figura 10, 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.

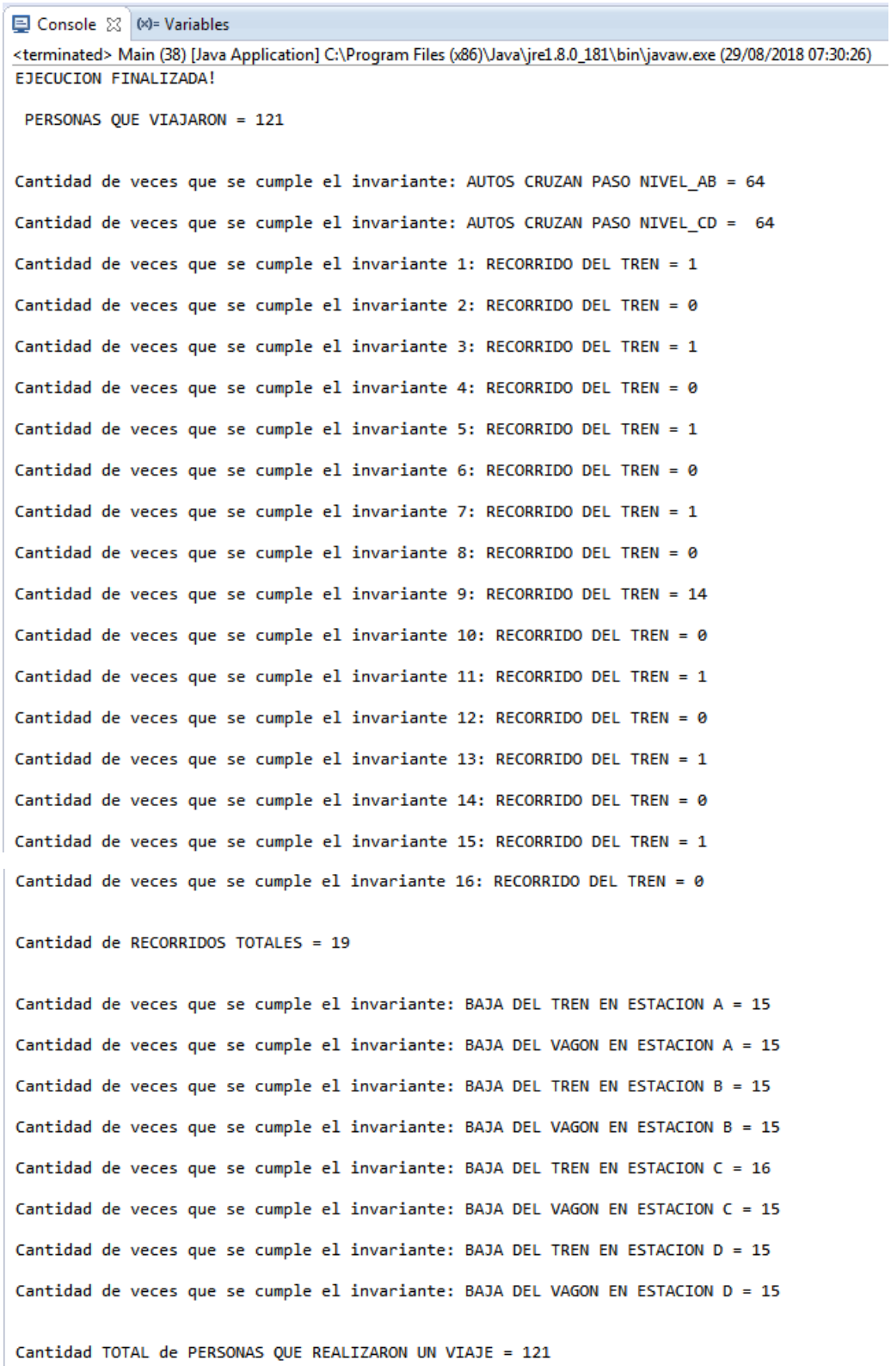


Figura 11: Ejecución del programa con cuatro estaciones.

Como se aprecia en la Figura 11, 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.

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 de una dificultad elevada.