UNIVERSIDAD NACIONAL DE CÓRDOBA
Facultad de Ciencias Exactas, Físicas y Naturales


# TRABAJO PRÁCTICO INTEGRADOR DE PROGRAMACIÓN CONCURRENTE

## Tren con 4 estaciones

**Integrantes:**

| | |
|---|---|
| **SORIANO, JUAN.** | **37.753.154** |
| **VIGNOLLES, IVAN.** | **36.407.987** |
| **D'ANDREA, F. DAVID.** | **33.178.206** |

**DOCENTES: Micolini, Orlando. Ventre, Luis.**

**AÑO: 2017**

# Introducción y objetivos

**Enunciado**

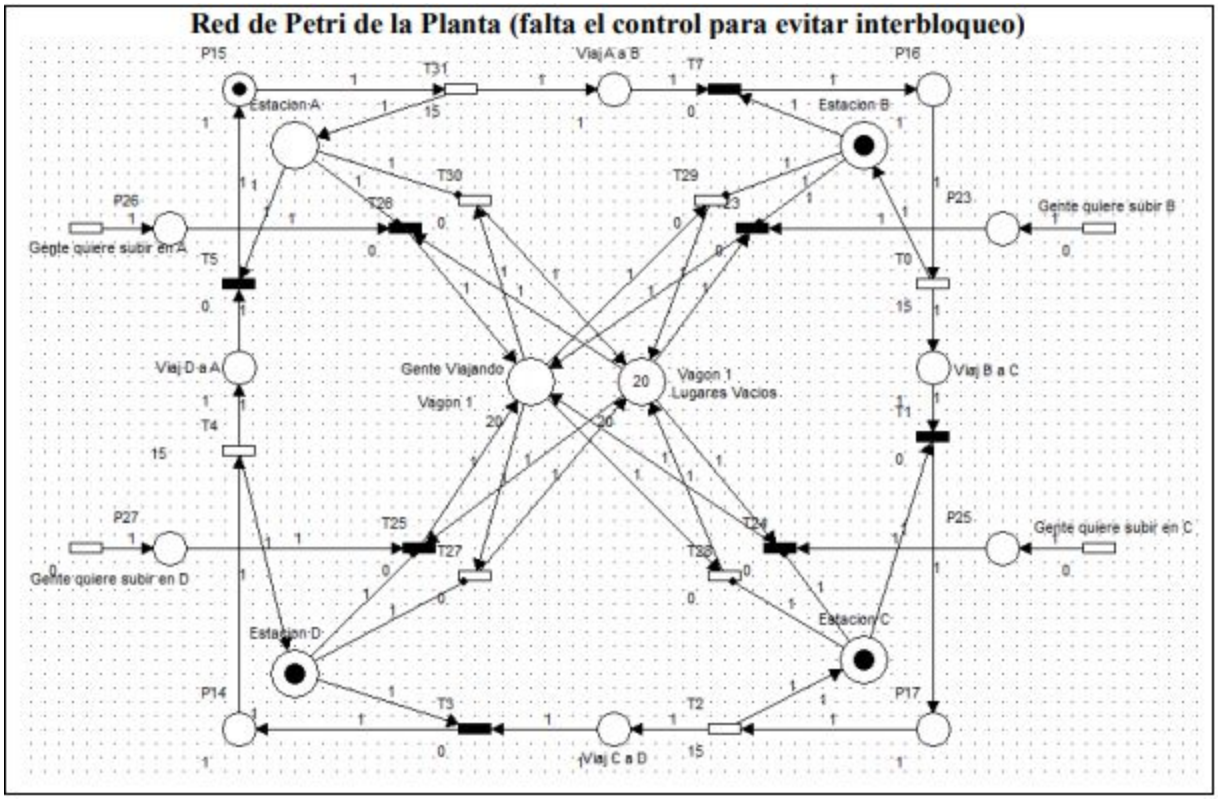
En este práctico se debe resolver el problema de control de un circuito ferroviario. Como dato se propone la red de Petri que modela una planta con 4 estaciones, un vagón, sin barreras. La red debe ser modificada con el fin de modelar la planta requerida y evitar interbloqueos. Luego simular la solución en un proyecto desarrollado con la herramienta adecuada (explique porque eligió la herramienta usada).

La planta requerida está formada por 4 estaciones (Estación A, Estación B, Estación C y Estación D), una máquina y un vagón. La capacidad de la maquina es de 30 pasajeros, mientras que la capacidad del vagón es de 20 pasajeros. En cada estación los pasajeros pueden subir o bajar; no pudiendo descender en cada estación los pasajeros que han ascendido en esa (no es necesario identificar los pasajeros, solo número).

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 se debe controlar la barrera para el paso de los vehículos y el tren. La barrera debe bajar 30 metros antes que llegue el tren a paso nivel y subir después de 20 metros que el tren a atravesado el paso a nivel.

El tren debe detenerse en cada estación no menos de 10 segundos y debe arrancar una vez que hayan subido todos los pasajeros o no haya lugar en maquina ni vagón.

El sistema controlador debe estar conformado por distintos hilos, los cuales deben ser asignados a cada conjunto de responsabilidades afines en particular. Por ej. Manejar el tren, manejar las barreras, etc.

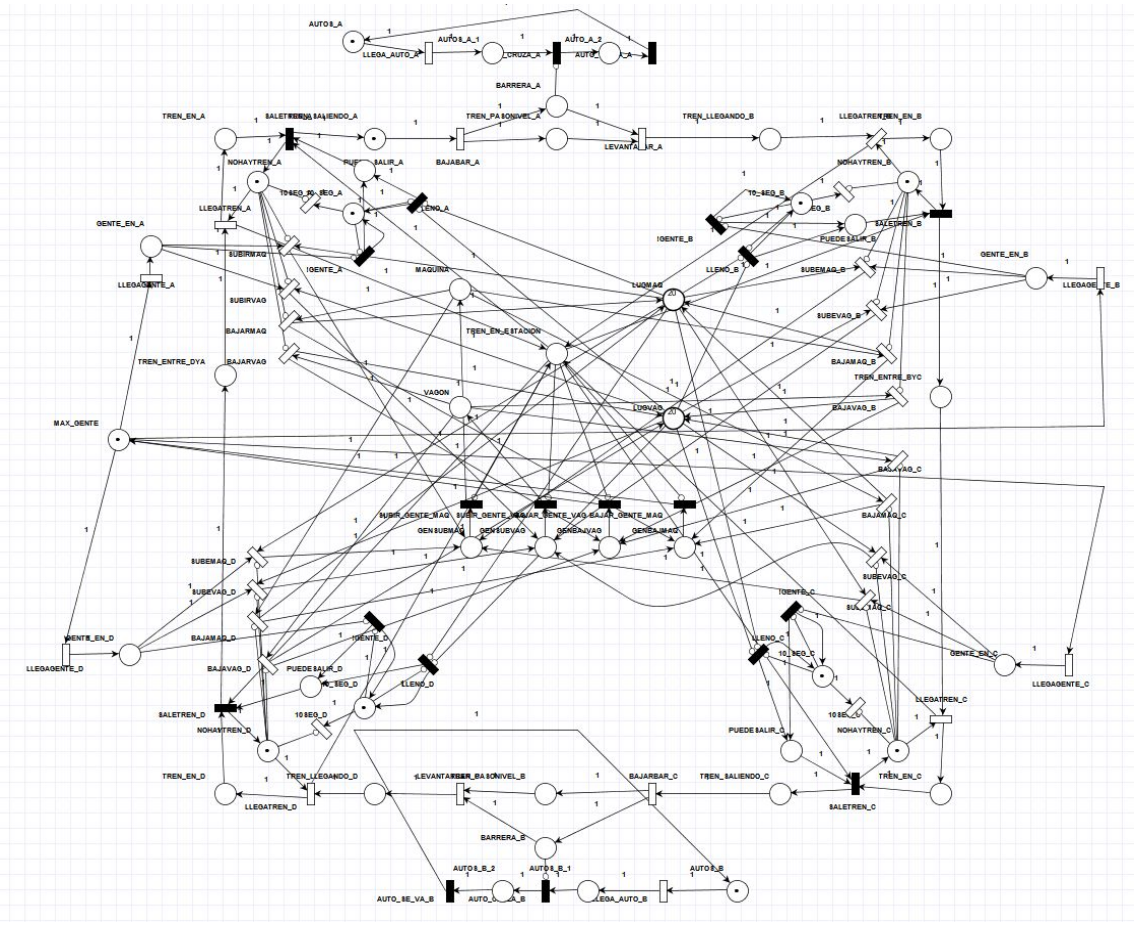


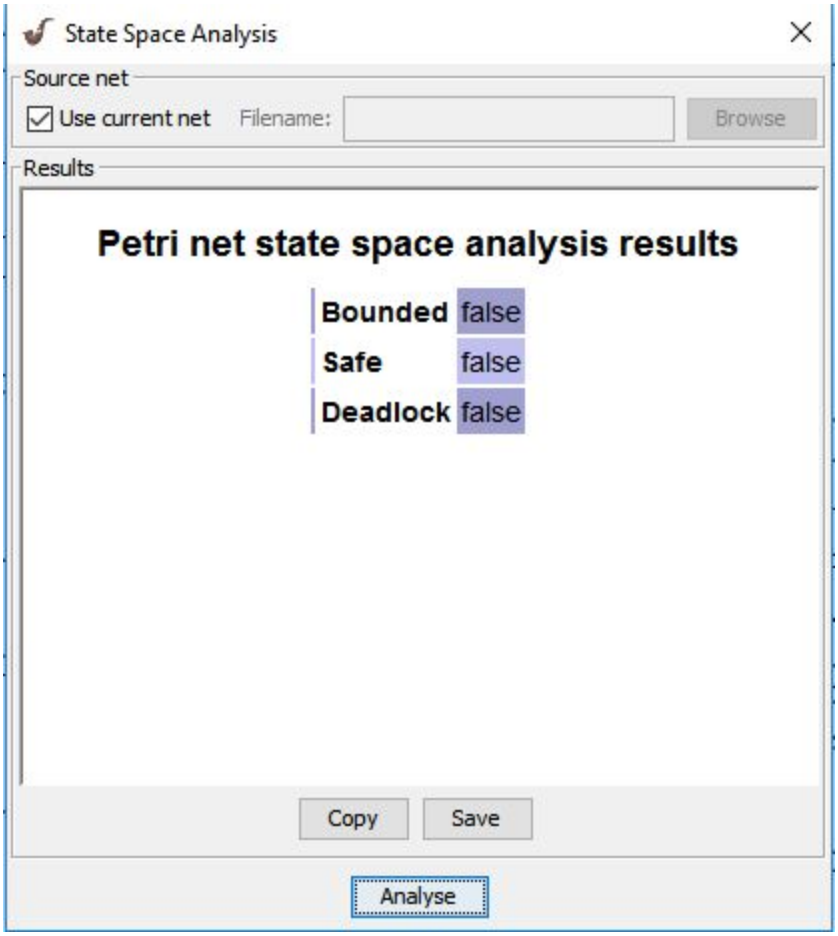
**Realizar**

- Colocar las restricciones a la RdP para evitar el interbloqueo, mostrarlo con la herramienta elegida y justificarlo.
- Colocar los tiempos en las estaciones (en las transiciones correspondientes).
- Hacer la tabla de eventos.
- Hacer la tabla de estados o actividades.
- Determinar la cantidad de hilos necesarios (justificarlo)
- Implementar dos caso de Políticas para producir:
    - a. Prioridad a los pasajeros que bajan
    - b. Prioridad a los pasajeros que suben
- Hacer el diagrama de clases.
- Hacer los diagramas de secuencias.
- Hacer el código.
- Hacer el testing.

# Desarrollo

Utilizando el pipe 4 y 5, se dibujó y se analizó la red. A continuación se muestra la red implementada:






El análisis fue hecho sobre una red con 2 estaciones en lugar de 4 debido a las limitaciones que presenta el pipe a la hora de analizar redes grandes. Como se puede ver, no presenta deadlocks. Por otra parte, la red es limitada y se puede verlo fácilmente. Sin embargo, el pipe arroja que la red no es “Bounded”, lo cual nos llamó la atención y se lo atribuimos a una mala implementación del análisis o un bug de la red al dibujarla.

## Tabla de eventos

| Descripción del evento | Transición |
|---|---|
| No hay más gente en A | !GENTE_A |
| No hay más gente en B | !GENTE_B |
| No hay más gente en C | !GENTE_C |
| No hay más gente en D | !GENTE_D |
| Tren espera 10 segundos para salir de A | 10SEG_A |
| Tren espera 10 segundos para salir de B | 10SEG_B |
| Tren espera 10 segundos para salir de C | 10SEG_C |
| Tren espera 10 segundos para salir de D | 10SEG_D |

| | |
|---|---|
| Auto cruza el pasonivel A-B | AUTO_CRUZA_A |
| Auto cruza el pasonivel C-D | AUTO_CRUZA_B |
| Auto se va del sistema en el pasonivel A-B | AUTO_SE_VA_A |
| Auto se va del sistema en el pasonivel C-D | AUTO_SE_VA_B |
| Se baja la barrera en el pasonivel A-B | BAJABAR_A |
| Gente baja de la máquina en la estación B | BAJAMAQ_B |
| Gente baja de la máquina en la estación C | BAJAMAQ_C |
| Gente baja de la máquina en la estación D | BAJAMAQ_D |
| La gente se baja de la máquina | BAJAR_GENTE_MAQ |
| La gente se baja del vagón | BAJAR_GENTE_VAG |
| Se baja la barrera en el pasonivel C-D | BAJARBAR_C |
| Gente baja de la máquina en la estación A | BAJARMAQ |
| Gente baja del vagón en la estación A | BAJARVAG |
| Gente baja del vagón en la estación B | BAJAVAG_B |
| Gente baja del vagón en la estación C | BAJAVAG_C |
| Gente baja del vagón en la estación D | BAJAVAG_D |
| Se levanta la barrera en el pasonivel A-B | LEVANTABAR_A |
| Se levanta la barrera en el pasonivel C-D | LEVANTARBAR_C |
| Llega un auto a esperar el pasonivel A-B | LLEGA_AUTO_A |
| Llega un auto a esperar el pasonivel C-D | LLEGA_AUTO_B |
| Llega gente a la estación A | LLEGAGENTE_A |
| Llega gente a la estación B | LLEGAGENTE_B |
| Llega gente a la estación C | LLEGAGENTE_C |
| Llega gente a la estación D | LLEGAGENTE_D |
| Llega el tren a la estación A | LLEGATREN_A |
| Llega el tren a la estación B | LLEGATREN_B |
| Llega el tren a la estación C | LLEGATREN_C |
| Llega el tren a la estación D | LLEGATREN_D |
| El tren se llenó en la estación A | LLENO_A |
| El tren se llenó en la estación B | LLENO_B |
| El tren se llenó en la estación C | LLENO_C |
| El tren se llenó en la estación D | LLENO_D |
| El tren sale de A | SALETREN_A |
| El tren sale de B | SALETREN_B |

| | |
|---|---|
| El tren sale de C | SALETREN_C |
| El tren sale de D | SALETREN_D |
| Sube gente a la máquina en la estación B | SUBEMAQ_B |
| Sube gente a la máquina en la estación C | SUBEMAQ_C |
| Sube gente a la máquina en la estación D | SUBEMAQ_D |
| Sube gente al vagón la estación B | SUBEVAG_B |
| Sube gente al vagón la estación C | SUBEVAG_C |
| Sube gente al vagón la estación D | SUBEVAG_D |
| La gente sube a la máquina | SUBIR_GENTE_MAQ |
| La gente sube al vagón | SUBIR_GENTE_VAG |
| Gente sube a la máquina en la estación A | SUBIRMAQ |
| Gente sube al vagón en la estación A | SUBIRVAG |

## Tabla de estados o actividades

| Descripción del estado | Plaza |
|---|---|
| Tren esperando 10 segundos para salir de A | 10_SEG_A |
| Tren esperando 10 segundos para salir de B | 10_SEG_B |
| Tren esperando 10 segundos para salir de C | 10_SEG_C |
| Tren esperando 10 segundos para salir de D | 10_SEG_D |
| Auto ha pasado pasonivel entre A y B | AUTO_A_2 |
| Auto a la espera de llegar al pasonivel A-B | AUTOS_A |
| Auto previo al pasonivel entre A y B | AUTOS_A_1 |
| Auto a la espera de llegar al pasonivel C-D | AUTOS_B |
| Auto previo al pasonivel entre C y D | AUTOS_B_1 |
| Auto ha pasado pasonivel entre C y D | AUTOS_B_2 |
| Barrera baja en A-B | BARRERA_A |
| Barrera baja en C-D | BARRERA_B |
| Gente que se baja de la máquina en la estación | GENBAJMAQ |
| Gente que se baja del vagón en la estación | GENBAJVAG |
| Gente que se sube a la máquina en la estación | GENSUBMAQ |
| Gente que se sube al vagón en la estación | GENSUBVAG |
| Gente esperando al tren en la estación A | GENTE_EN_A |
| Gente esperando al tren en la estación B | GENTE_EN_B |
| Gente esperando al tren en la estación C | GENTE_EN_C |
| Gente esperando al tren en la estación D | GENTE_EN_D |
| Lugares disponibles en la máquina | LUGMAQ |

| | |
|---|---|
| Lugares disponibles en el vagón | LUGVAG |
| Lugares ocupados en la máquina | MAQUINA |
| Máximo de gente en el "universo" | MAX_GENTE |
| El tren no está en la estación A | NOHAYTREN_A |
| El tren no está en la estación B | NOHAYTREN_B |
| El tren no está en la estación C | NOHAYTREN_C |
| El tren no está en la estación D | NOHAYTREN_D |
| El tren puede salir de la estación A | PUEDE_SALIR_A |
| El tren puede salir de la estación B | PUEDESALIR_B |
| El tren puede salir de la estación C | PUEDESALIR_C |
| El tren puede salir de la estación D | PUEDESALIR_D |
| El tren está en la estación A | TREN_EN_A |
| El tren está en la estación B | TREN_EN_B |
| El tren está en la estación C | TREN_EN_C |
| El tren está en la estación D | TREN_EN_D |
| El tren está en alguna estación | TREN_EN_ESTACION |
| El tren está entre la estación B y la C | TREN_ENTRE_BYC |
| El tren está entre la estación D y la A | TREN_ENTRE_DYA |
| El tren está llegando a la estación B | TREN_LLEGANDO_B |
| El tren está llegando a la estación D | TREN_LLEGANDO_D |
| El tren está cruzando por el pasonivel A-B | TREN_PASONIVEL_A |
| El tren está cruzando por el pasonivel C-D | TREN_PASONIVEL_B |
| El tren está saliendo de la estación A | TREN_SALIENDO_A |
| El tren está saliendo de la estación C | TREN_SALIENDO_C |
| Lugares ocupados en el vagón | VAGON |

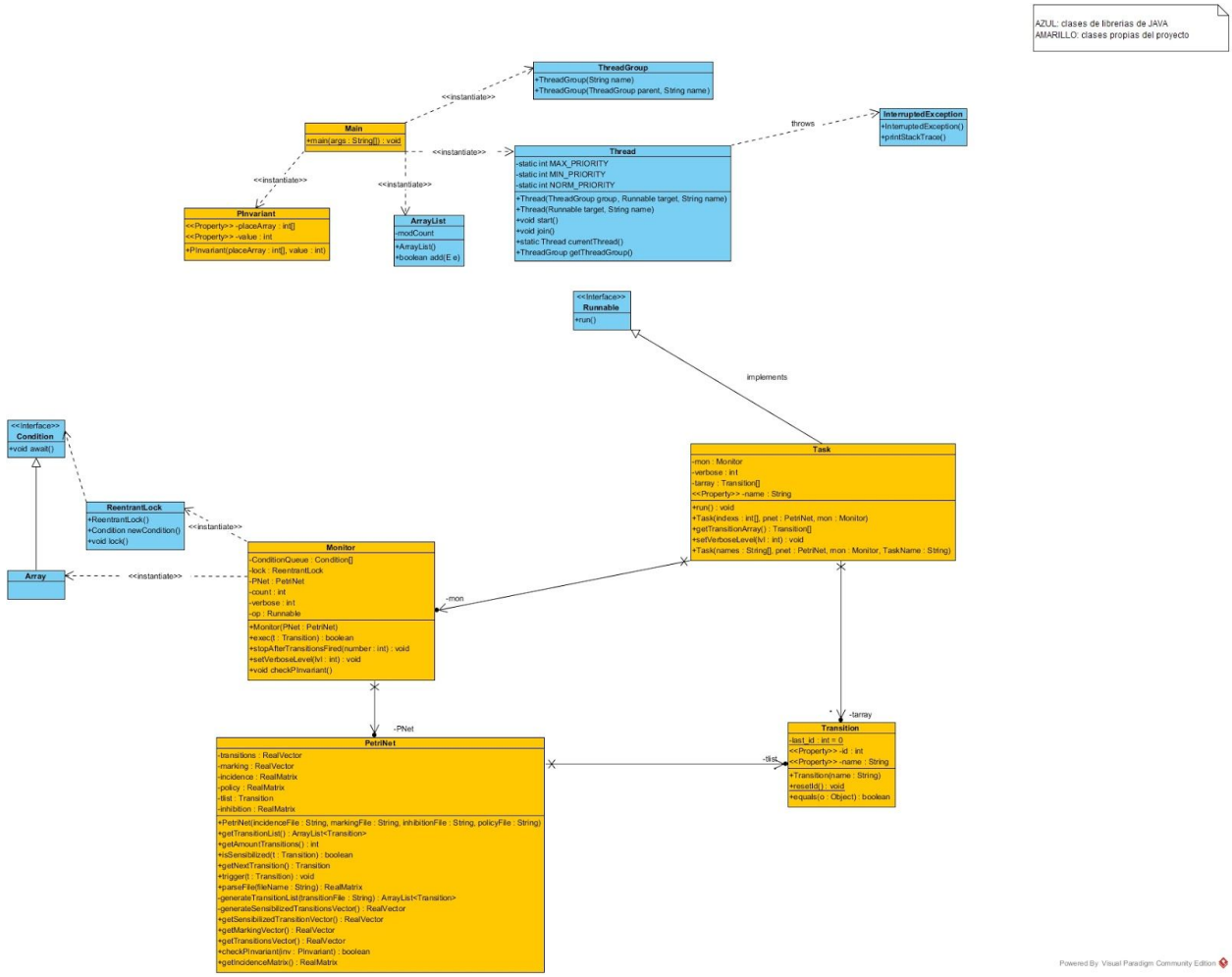
## Cantidad de hilos necesarios

Analizando el problema se llega a la conclusión de que, como fue dibujada la red, el tren se ejecuta con un thread debido a que es un conjunto de transiciones secuenciales. De la misma forma, los autos en los pasonivel tienen el mismo efecto, siempre y cuando se limite la cantidad máxima a 1. No fue posible agrupar muchas transiciones bajo la ejecución de un mismo hilo debido a la naturaleza del problema.

## Política implementada

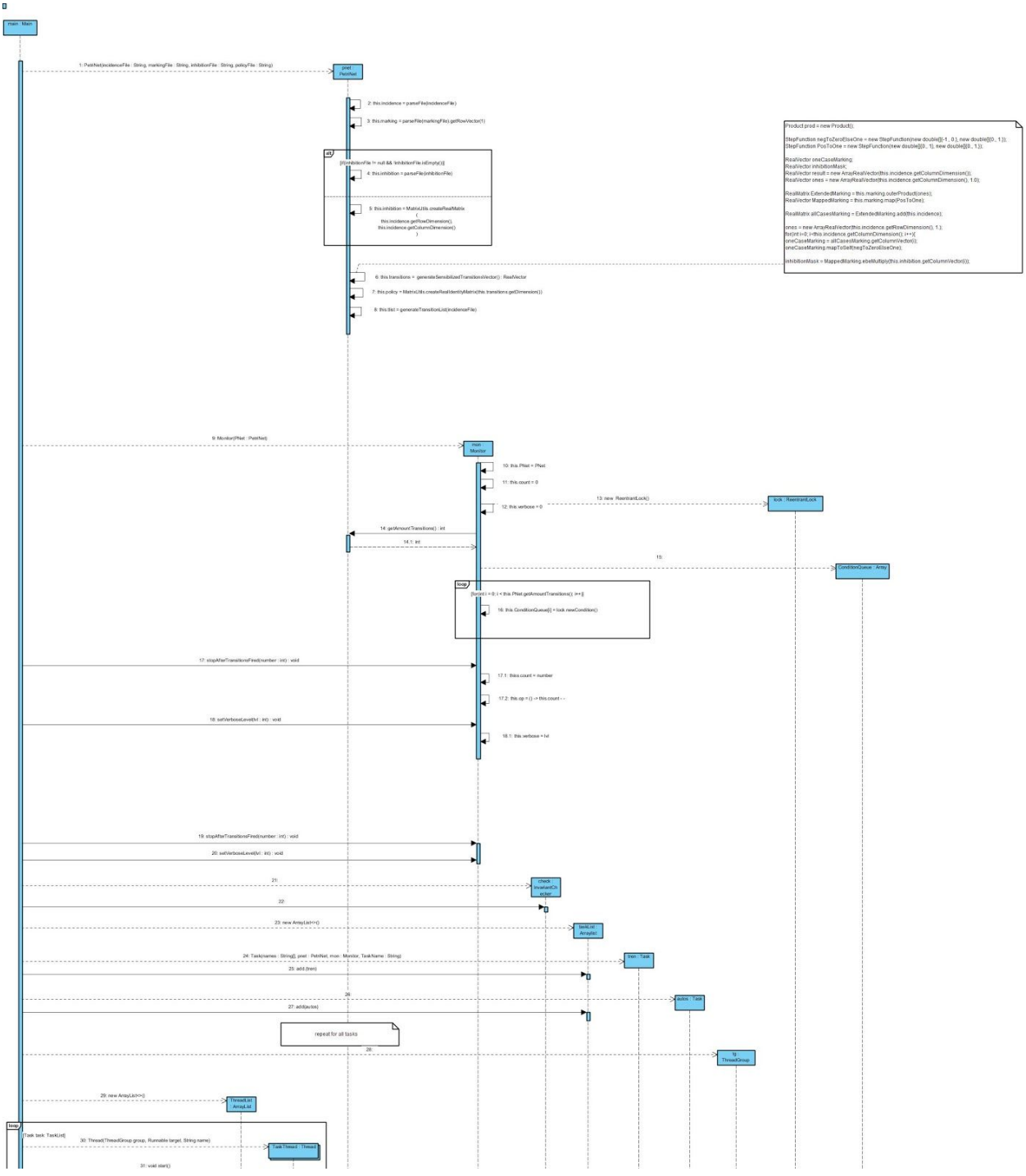
Se implementaron dos políticas, una para dar prioridad a los pasajeros que suben y otra a los que bajan, esto se ve reflejado en la matriz de política que se utiliza en el proyecto.

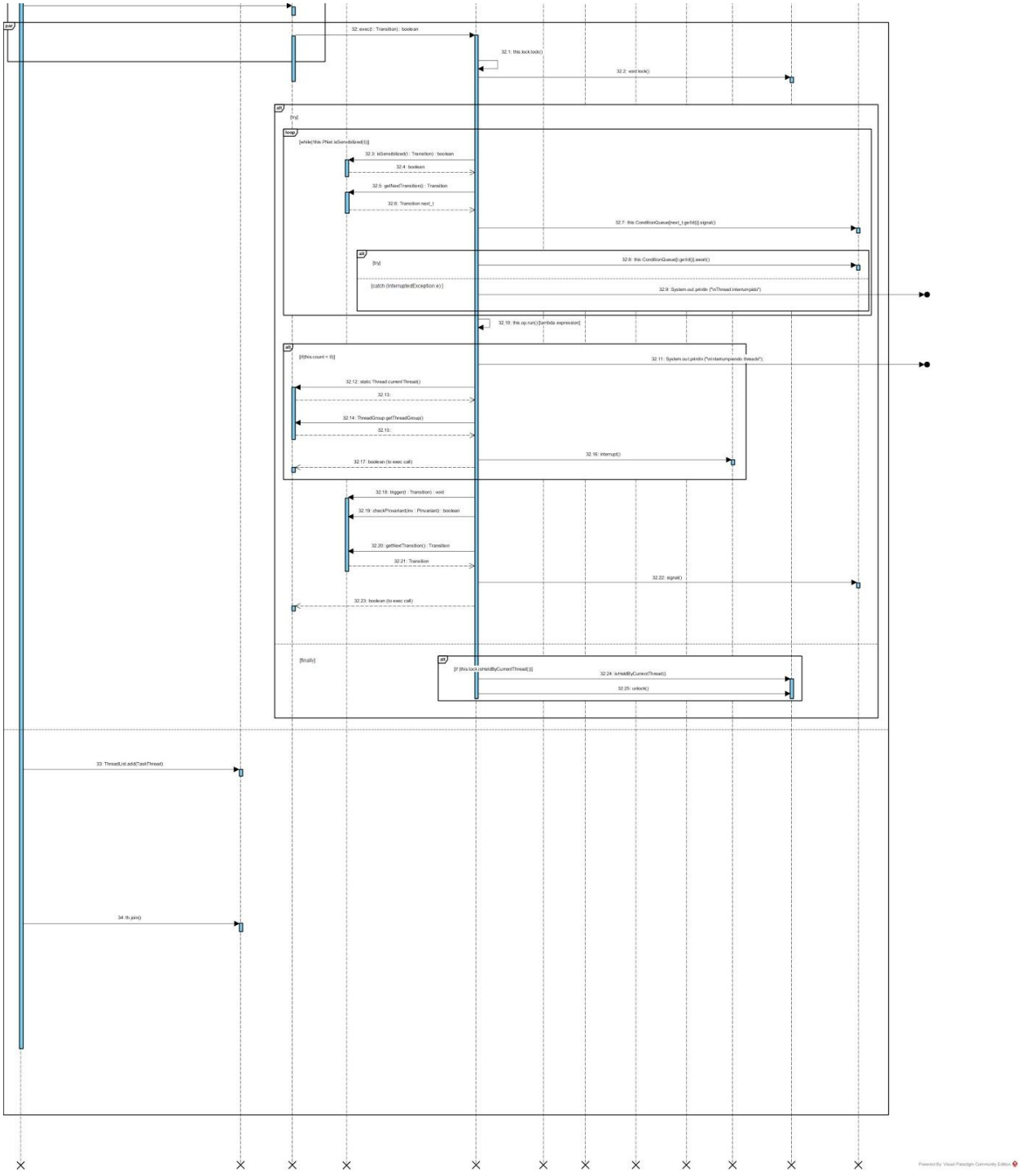
## Diagrama de clases



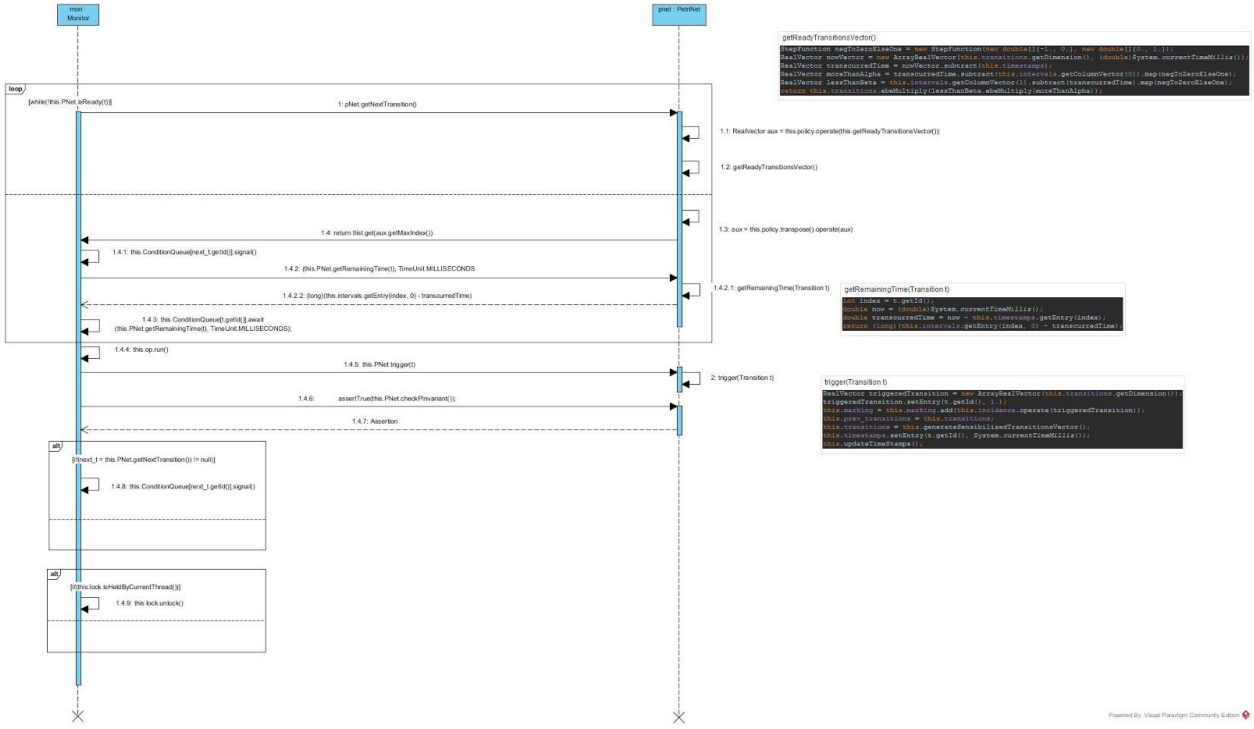

Para visualizar el diagrama de clases con mayor detalle, se encuentra disponible en: https://github.com/SorianoJuan/ProgConcurrente-UNC

## Diagrama de secuencias completo




par
32: exec(t : Transition) : boolean
32.1: this.lock.lock()
32.2: void lock()
alt
[try]
loop
[while(!this.PNet.isSensibilized(t))]
32.3: isSensibilized(t : Transition) : boolean
32.4: boolean
32.5: getNextTransition() : Transition
32.6: Transition next_t
32.7: this.ConditionQueue[next_t.getId()].signal()
alt
[try]
32.8: this.ConditionQueue[t.getId()].await()
[catch (InterruptedException e) ]
32.9: System.out.println ("\nThread interrumpida")
32.10: this.op.run() [lambda expression]
alt
[if(this.count < 0)]
32.11: System.out.println ("\n\tInterrumpiendo threads");
32.12: static Thread currentThread()
32.13:
32.14: ThreadGroup getThreadGroup()
32.15:
32.16: interrupt()
32.17: boolean (to exec call)
32.18: trigger(t : Transition) : void
32.19 :checkPrivariantInv : Privariant) : boolean
32.20: getNextTransition() : Transition
32.21: Transition
32.22: signalt()
32.23: boolean (to exec call)
[finally]
alt
[if (this.lock.isHeldByCurrentThread())]
32.24 : isHeldByCurrentThread()
32.25: unlock()
33: ThreadList.add(TaskThread)
34: th.join()

## Diagrama de secuencia (interacción Monitor - PetriNet)




Para visualizar el diagrama de secuencias con mayor detalle, se encuentra disponible en: https://github.com/SorianoJuan/ProgConcurrente-UNC

## Código y testing

Se analizaron tanto los P-Invariantes como los T-Invariantes. En cada ejecución de un thread en el monitor, se chequean los P-Invariantes. Utilizando el log de salida del programa, en python, se parsea la salida y se validan los órdenes de ejecución de las transiciones.

El código con su testing correspondiente se encuentra disponible en: https://github.com/SorianoJuan/ProgConcurrente-UNC

# Conclusiones

Durante la resolución del problema planteado, se fueron tomando distintos enfoques para solucionar inconvenientes como el del interbloqueo o evitar que la gente que subía en una estación, se bajara en otras. Para evitar esto último, se introdujeron plazas nuevas de gente que se subió en esa estación. Cuando el tren abandonaba dicha estación, la gente pasaba a la máquina o al vagón según correspondía.

Se le dió prioridad al tren por sobre los autos a la hora de utilizar la barrera para ajustarse a un problema del mundo real, en el cual cuando un tren cruza un paso nivel, la barrera siempre le da prioridad al mismo.

Por otra parte, se introdujeron plazas para limitar el marcado de la red y evitar utilizar una construcción del tipo “generador de tokens”. Esto permitió que el Pipe pudiera realizar el análisis de estados que de otra manera hubiera sido imposible porque los generadores de tokens introducen infinitos estados a la red.

En este trabajo no se le dio un trato distinto a las transiciones que son decisiones con respecto a las que son acciones, todas fueron tratadas de la misma forma en el monitor, cada una con su cola de condiciones.

A modo de cierre, se puede decir que la abstracción con la que se implementó el código permite, en caso de cambiar el programa, simplemente dibujar una nueva red de petri, exportar los archivos correspondientes y la ejecución será la misma. Por otra parte, la red de petri se tornó bastante compleja, con lo que hubo que reducirla para analizarla con el Pipe. En caso de querer resolver un problema más complejo, se debe utilizar otra herramienta.

# Bibliografía


- Programación Concurrente - José Tomás Palma Méndez
- Java 7 Concurrency Cookbook - Javier Fernández González
- Sistemas operativos - William Stallings
- Petri Nets Fundamental Models, Verification and Applications