

MODELADO Y DISEÑO DEL SOFTWARE

UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

Alba de la Torre Segato
Álvaro Valencia Villalón
Pablo Astudillo Fraga
Carla Serracant Guevara
Pablo Alarcón Carrión
INGENIERÍA DEL SOFTWARE

P2 - TRENES

CONTEXTO

Para empezar empezaremos describiendo un poco el concepto de la práctica y su finalidad. Para esta práctica debemos modelar un sistema de trenes, en el que tendremos las respectivas líneas, con sus vías y sus trenes. Cada par de vías compondrán un segmento que será la unión entre dos estaciones. Más tarde deberemos añadir a nuestro modelo el transcurso del tiempo y para ello deberemos crear una clase reloj que será representada además por una clase abstracta llamada ActiveObject, de esta manera podremos hacer que los trenes se muevan automáticamente con el paso del tiempo.

DIAGRAMAS

USE

DIAGRAMA DE CLASES

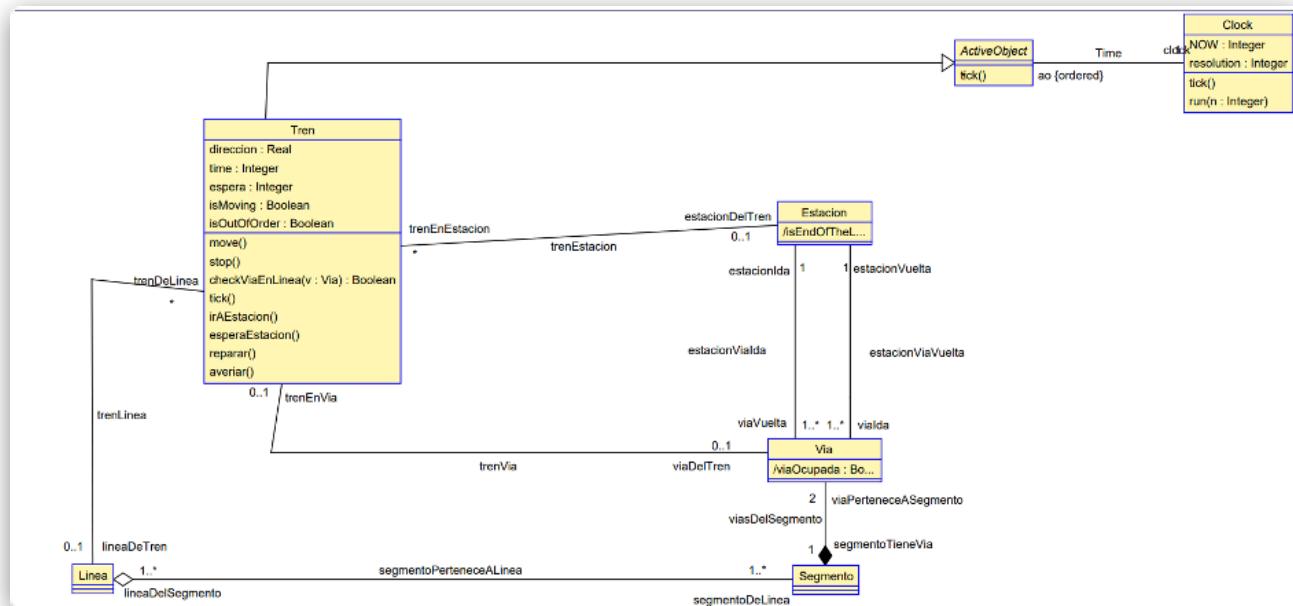
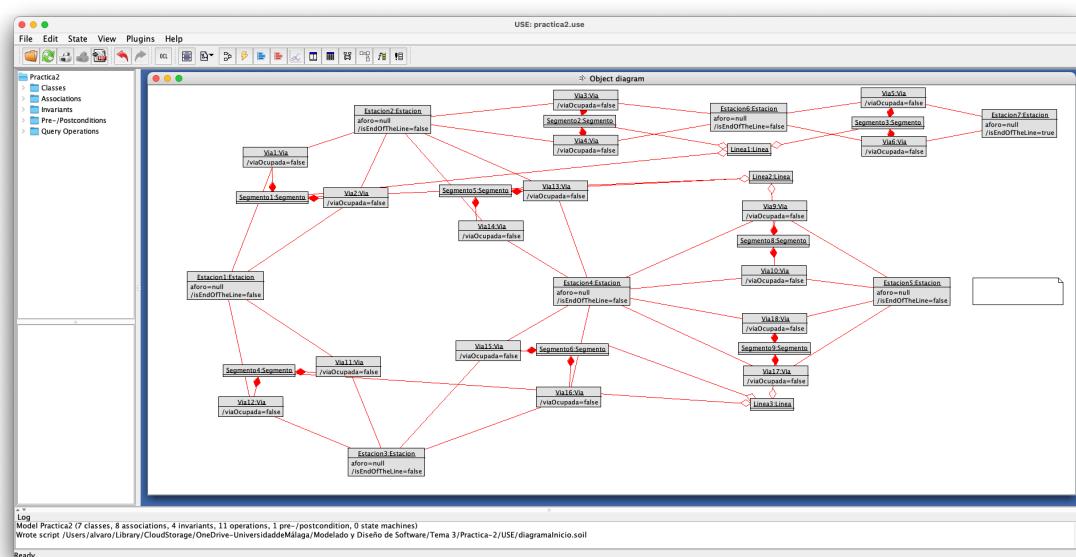
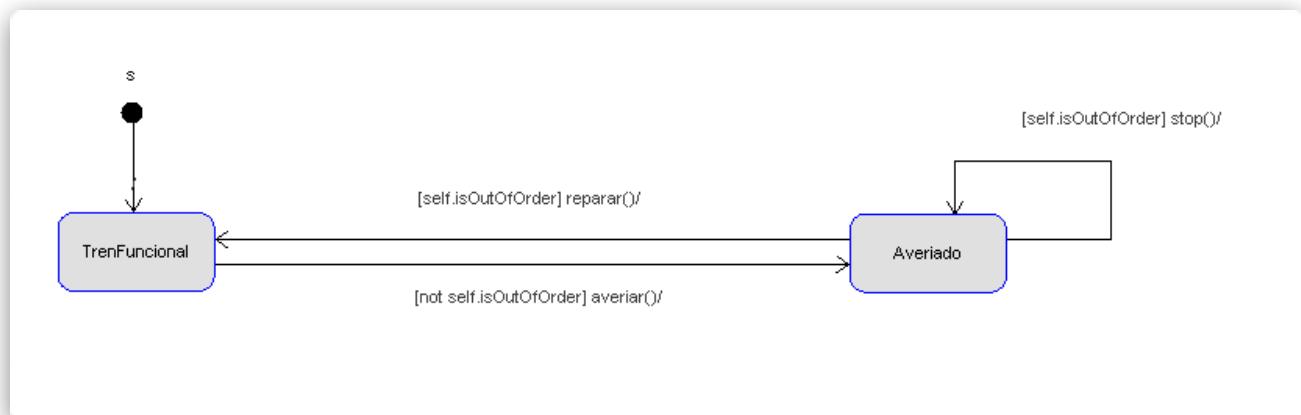


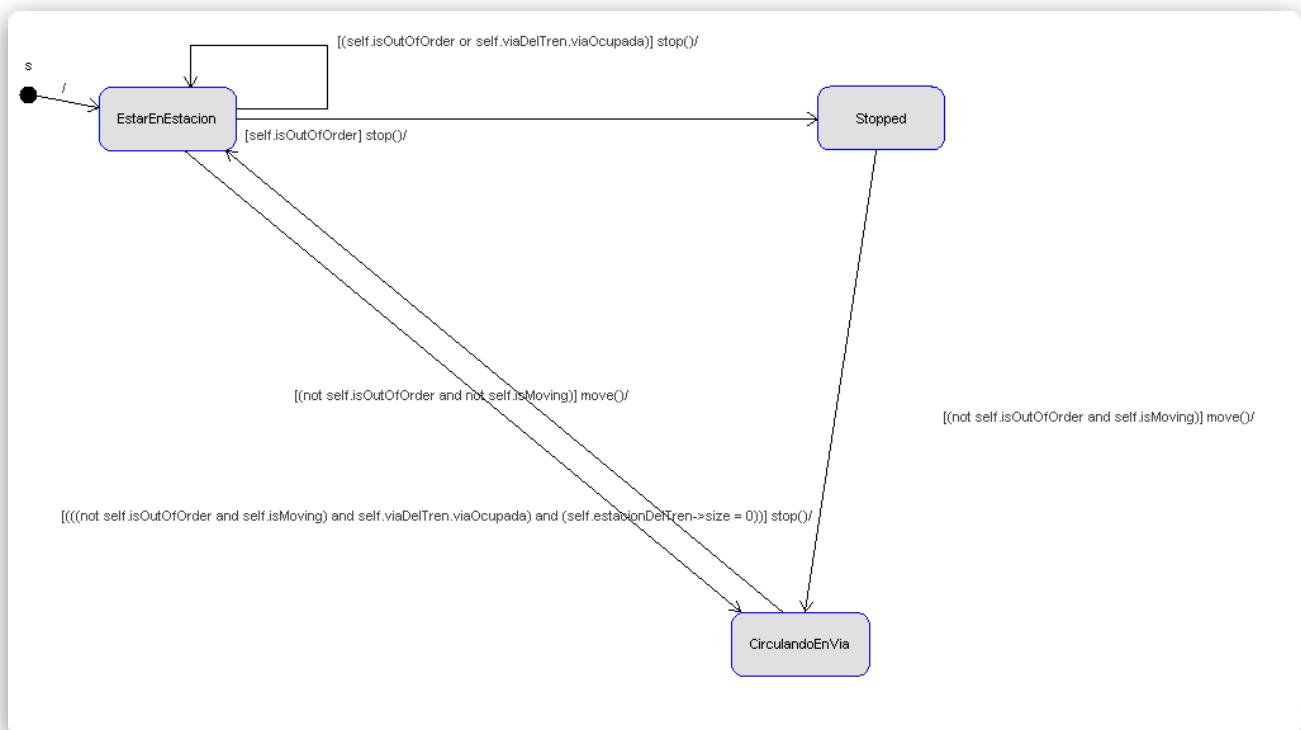
DIAGRAMA DE OBJETOS



STATEMACHINE



Tren averiado



Tren funcionando

VISUAL PARADIGM

DIAGRAMA DE CLASES

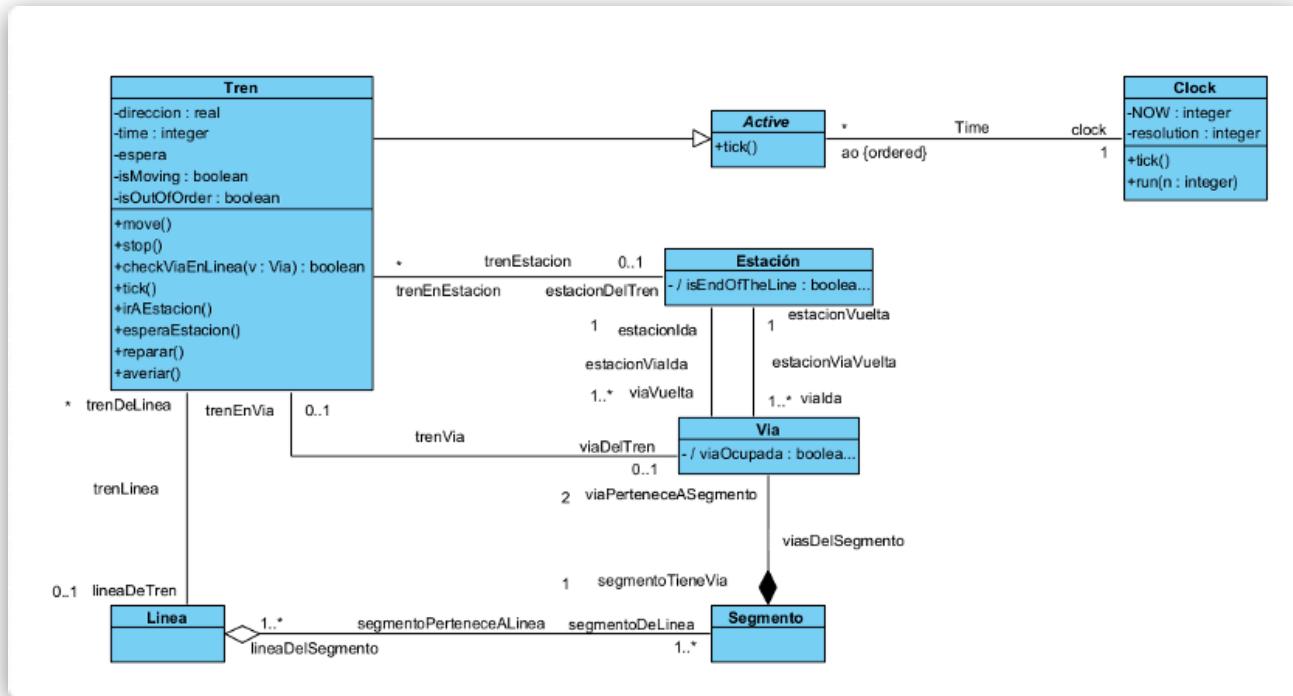
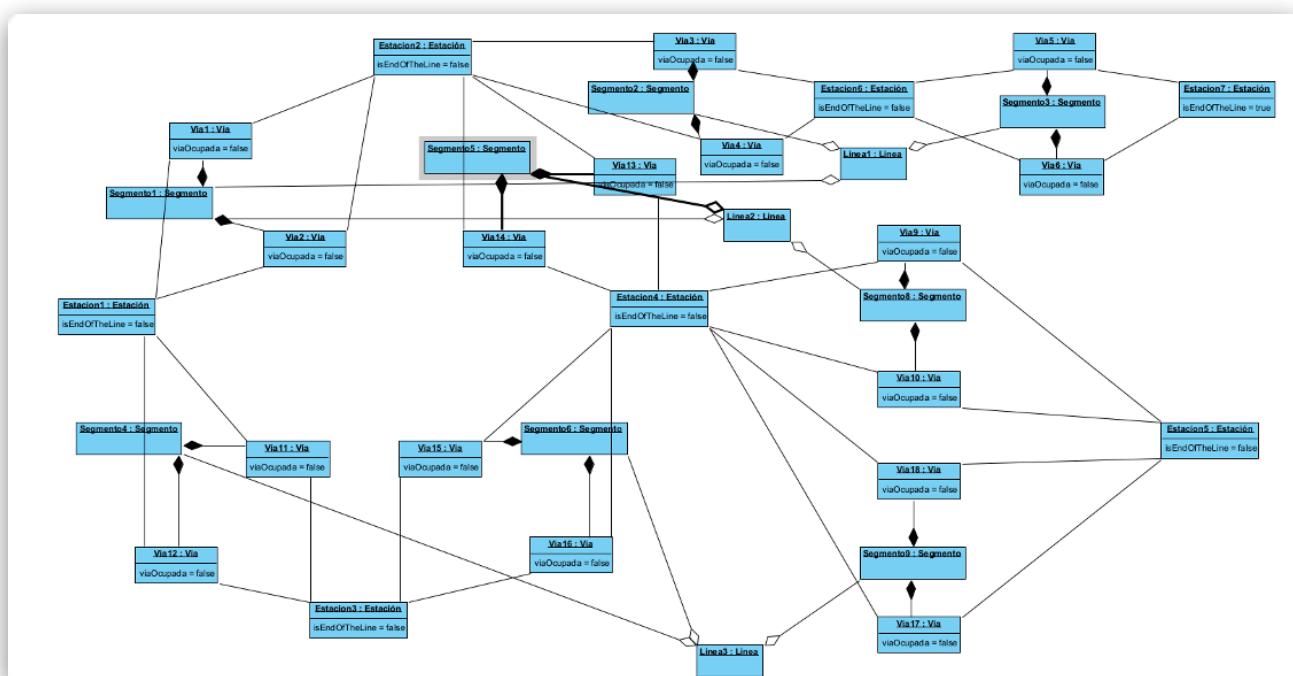
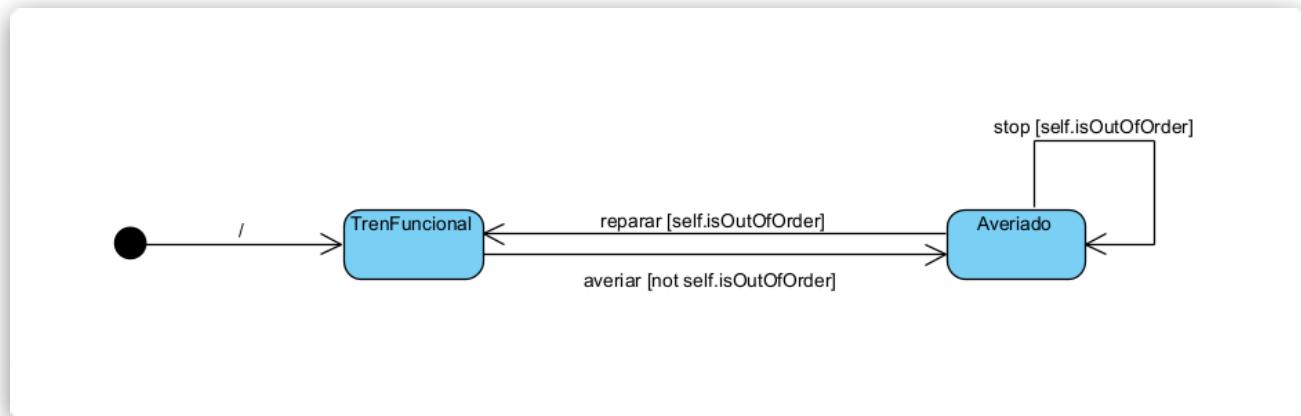


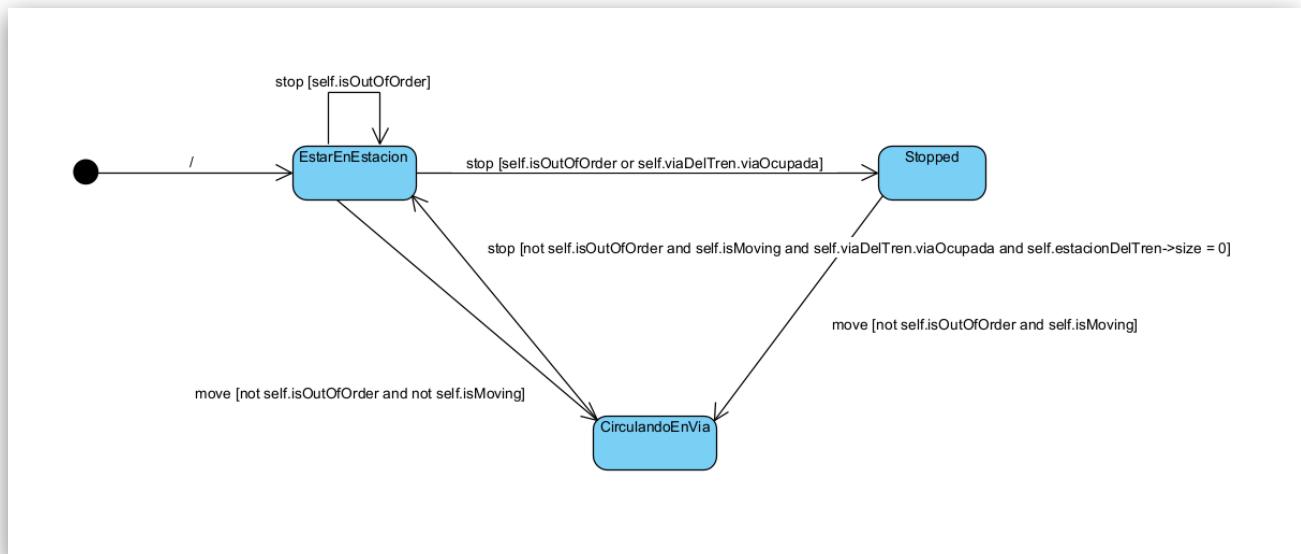
DIAGRAMA DE OBJETOS



STATEMACHINE



Tren averiado



Tren funcionando

PAPYRUS

DIAGRAMA DE CLASES

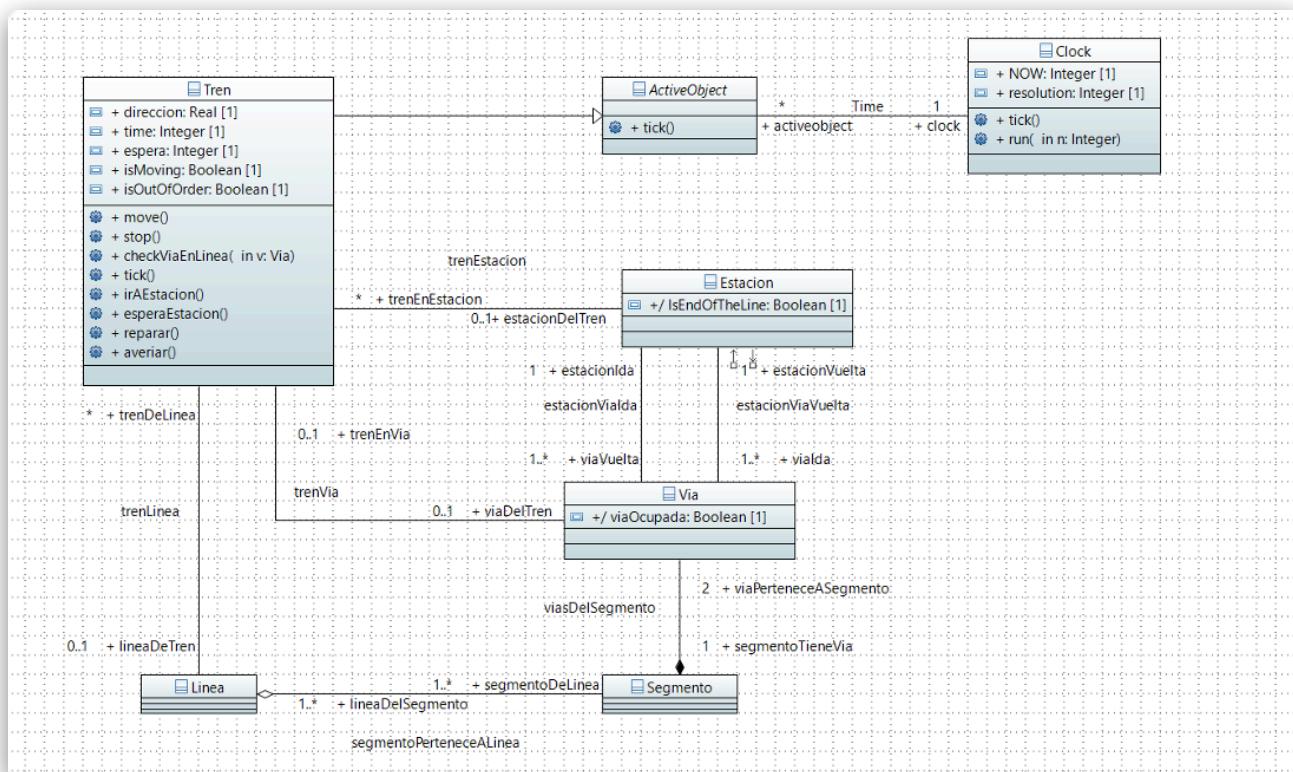
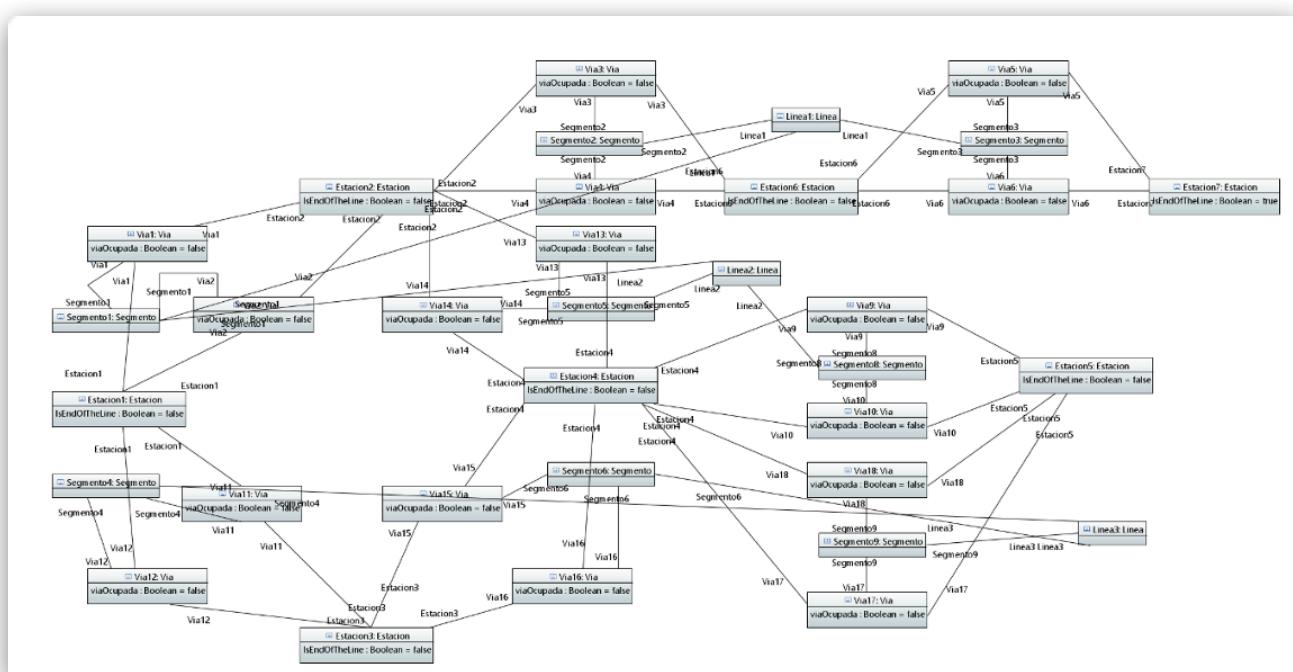
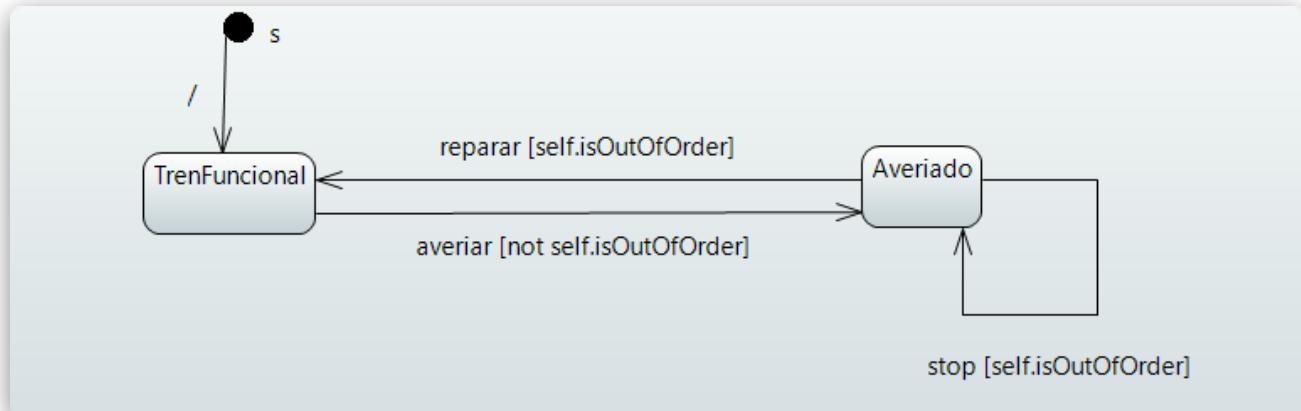


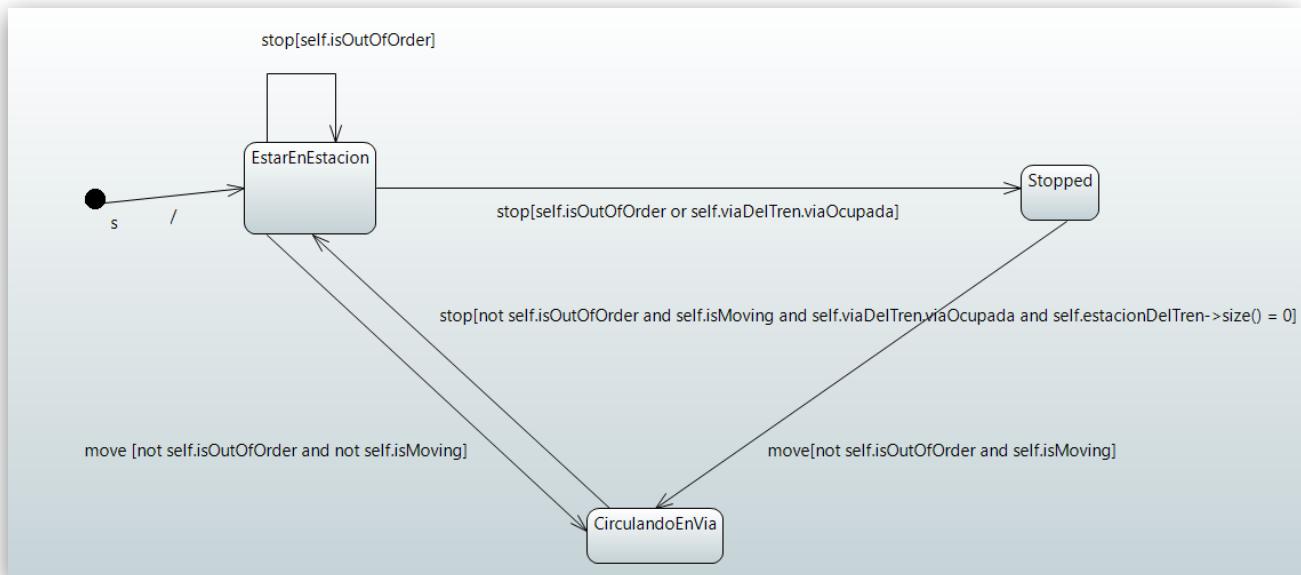
DIAGRAMA DE OBJETOS



STATEMACHINE



Tren averiado



Tren funcionando

ALGUNAS CONSIDERACIONES SOBRE EL DIAGRAMA DE CLASES Y A NIVEL CONCEPTUAL DEL PROBLEMA

- Las líneas son independientes de los trenes, por lo que no tiene sentido hacer composiciones entre líneas y trenes. De hecho podemos tener 0 trenes en una línea en algunos casos, y los trenes pueden cambiar de linea, aunque en su funcionamiento siempre siguen una.
- Líneas y segmento son agregación porque un segmento puede pertenecer a varias líneas
- Una estación se conecta a otra mediante un dos vías, las cuales forman un segmento. De esta forma, como no hay estaciones aisladas, una estación siempre tiene dos vías y un segmento. Siempre habrá un número par de vías en una estación.
- No tiene sentido una línea sin segmento ni un segmento que no pertenezca a una linea.
- Un segmento puede pertenecer a más de una línea, y una línea está formada por segmentos.
- Debido a las multiplicidades, dos trenes no van a estar en la misma vía.

EXPLICACIÓN

CLASES

Ahora vamos a seguir describiendo las diferentes clases de nuestro modelo junto con sus atributos y operaciones.

- Tren: es una clase que representará a un tren, que irá desplazándose de estación a estación y creando nuevas relaciones con estas para representarlo. Sus atributos son
 - Dirección: nos indica la dirección en la que irá el tren, cuando este llegue al final de la línea su dirección se invertirá.
 - Time: es un atributo de tipo integer que nos mostrará el tiempo que lleva moviéndose o mejor dicho “trabajando”.
 - isMoving: es un atributo de tipo boolean que nos indicará si el tren está moviéndose o no.
 - isOutOfOrder: este atributo de tipo boolean nos indicará que el tren está fuera de servicio. Este se activará cuando nuestro tren se averie.
 - move(): operación utilizada para mover el tren de una posición a otra.
 - stop(): operación utilizada para parar el tren, utilizada para cuando llegue a una estación se pare un minuto.
 - checkViaEnLinea(v:Via): operación utilizada para comprobar que la siguiente vía pertenezca a la línea a la que pertenece el tren y no acceda a una vía que no pertenezca a dicha linea.
 - tick(): operación utilizada al heredar de la clase abstracta del reloj, esta es utilizada para aumentar los ticks de reloj o segundos en nuestro modelo de tiempo.
 - irAEstacion(): operación utilizada para indicar la siguiente vía que debe coger el tren.

- `esperaEstacion()`: operación utilizada para indicar al tren que debe esperar 1 minuto en la estación.
 - `reparar()`: operación utilizada para reparar el tren .
 - `averiar()`: operación utilizada para averiar el tren.
- `Clock`: es una clase utilizada para modelar el tiempo, en nuestra implementación cada tick será un segundo por lo que toda nuestra implementación de instantes y tiempos se basan en minuto, por lo que para cada minuto hacemos 60 ticks. Sus atributos son:
- `NOW`: es un atributo de tipo integer que nos indica el momento de tiempo en el que estamos.
 - `resolution`: es un atributo de tipo integer que indica cuantas unidades aumenta cada tick.
 - `tick()`: operación utilizada para aumentar los segundos/ticks del tiempo.
 - `run(n : integer)`: operación que hará el número de ticks que se le pase por la entrada.
- `ActiveObject`: clase abstracta de la clase `Clock`.
- `Estación`: clase que representa las estaciones, esta tendrá un único atributo que además será derivado y nos indicará cuando una estación es el final de una linea.
- `Via`: clase que representa las vias, esta tendrá tambien un único atributo derivado que nos dirá cuando la via esta ocupada y cuando no.
- `Segmento`: clase que representa el segmento que hay entre dos estaciones, esta estará compuesta por dos vias.
- `Línea`: clase que modela la linea de tren.

RELACIONES

Ahora vamos a exponer nuestras relaciones y explicar que representan y sus multiplicidades.

- Time: es una relación que relaciona una clase abstracta con la clase. Un reloj puede tener varios objetos activos. Y varios objetos activos pueden pertenecer a un solo reloj.
- Tren-ActiveObject: esta relación de herencia solo nos indica que el tren es un objeto activo y por tanto un usuario del tiempo.
- trenEstacion: esta relación representa cuando un tren está en una estación. A medida que el tren vaya moviéndose irá creando y borrando estas relaciones con determinadas instancias. Según nos indica el enunciado puede haber un número indefinido de trenes en una estación, pero un tren solo puede pertenecer en cada instante a una sola estación.
- Via-Estacion: en este caso hay dos relaciones entre estas dos clases.
 - estacionVialda: en este caso esta relación nos indica el sentido que tiene la vía para esa estación, ya que entre dos estaciones habrá dos vías, una para cada sentido.
 - estacionViaVuelta: de nuevo, esta relación nos indica el sentido que tiene dicha vía para dicha estación. Como una estación puede ser un nodo entre varias líneas y vías podemos decir que una estación puede tener de una a muchas vías de cada sentido. Pero solo una vía puede pertenecer a una estación.
- trenVia: esta relación es utilizada para indicarnos cuando un tren está en una vía. Un tren puede estar o no en una vía. Y una vía puede tener o no un tren.
- viasDelSegmento: como ya hemos dicho antes un segmento es aquello que une dos estaciones entre sí, dicho segmento se compone de dos y solo dos vías.
- segmentoPerteneceALinea: esta relación es utilizada para saber que segmentos tiene una línea. Cada línea puede tener uno o varios segmentos y viceversa, un mismo segmento puede pertenecer a una sola línea o a varias.

- trenLinea: con esta relación sabemos a que linea pertenece cada tren. Un tren puede pertenecer o no a una sola linea, es decir, un tren no puede pertenecer a más vias. Y una linea puede disponer de varios trenes.

CÓDIGO DE LA PRÁCTICA (.USE)

```

model Practica2

-- Reloj para apartado C --


class Clock
  attributes
    NOW : Integer init = 0
    resolution: Integer init = 1
  operations
    tick()
    begin
      self.NOW:=self.NOW + self.resolution;
      for o in self.ao do
        o.tick()
      end;
    end
    run (n:Integer)
    begin
      for i in Sequence{1..n} do
        self.tick()
      end
    end
  end

abstract class ActiveObject
  operations
    tick() begin end
end

association Time between
  Clock [1]
  ActiveObject [*] role ao ordered
end

-----
class Estacion
  attributes
    isEndOfTheLine : Boolean derive:
      self.viaIda->iterate(viaIda ; suma : Integer = 0 | suma+1) + self.viaVuelta-
>iterate(viaVuelta ; suma : Integer = 0 | suma+1) = 2
  end

class Tren < ActiveObject
  attributes
    direccion : Real init: 1
    time : Integer init : 0
    espera : Integer init : 0
    isMoving : Boolean init: false
    isOutOfOrder : Boolean init: false
  operations
    move()
    begin
      self.isMoving:=true;
    end

    stop()
    begin
      self.isMoving:=false;
    end

    checkViaEnLinea(v : Via):
      Boolean = v.segmentoTieneVia.lineaDelSegmento->exists(m|m=self.lineaDeTren) -- Para ahorrar
      codigo
      tick()

```

```

begin
    if (self.isMoving) then
        self.irAEstacion();
    end;

    if (self.isMoving=false) then
        self.esperaEstacion();
    end;
end
-- Operación para buscar la próxima vía a coger. Además los trenes deben tardar 2 min entre
estación y estación
irAEstacion()
begin
    if not self.isOutOfOrder then
        declare viaAnterior : Via;
        viaAnterior := self.viaDelTren;
        self.time := self.time + self.clock.resolution;

        if (self.time) = 180 then
            self.time := 0;
        if(self.viaDelTren >> null) then
            delete (self,self.estacionDelTren) from trenEstacion;
            insert (self,self.viaDelTren.estacionVuelta) into trenEstacion; --
self,nextStation
        delete (self,self.viaDelTren) from trenVia;
    end;

    if (self.estacionDelTren.isEndOfTheLine) then
        self.direccion:=self.direccion*-1;
        if(not self.estacionDelTren.viaIda->asSequence()->first().viaOcupada) then
            insert (self,self.estacionDelTren.viaIda->asSequence()->first()) into
trenVia; --no va a tener mas xq es fin de linea, pero USE lo coge como un Set de 1 elemento
        end;
    else
        if (self.direccion=1) then
            if(not self.estacionDelTren.viaIda->select(v|v>>viaAnterior and
self.checkViaEnLinea(v))->asSequence()->last().viaOcupada) then
                insert (self, self.estacionDelTren.viaIda->select(v|v>>viaAnterior and
self.checkViaEnLinea(v))->asSequence()->last()) into trenVia;
            end;
        else
            if (self.direccion=-1) then
                if(not viaAnterior.estacionVuelta.viaIda->select(v|
v>>viaAnterior.segmentoTieneVia.viaPerteneceASegmento->select(viaAux|
viaAux>>viaAnterior)->asSequence()->first() and
self.checkViaEnLinea(v))->asSequence()->first().viaOcupada) then
                    insert (self, viaAnterior.estacionVuelta.viaIda->select(v|
v>>viaAnterior.segmentoTieneVia.viaPerteneceASegmento->select(viaAux|
viaAux>>viaAnterior)->asSequence()->first() and
self.checkViaEnLinea(v))->asSequence()->first()) into trenVia;
                end;
            end;
        end;
    end;
end

-- Los trenes deben esperar 1 min en las estaciones
esperaEstacion()
begin
    self.time:=self.time + self.clock.resolution;
    if self.time = 60 then
        self.move();
    end;
end

-- Operación para reparar el tren
reparar()
begin
    self.isOutOfOrder := false;
end
pre outOfOrder : self.isOutOfOrder

-- Operación para averiar el tren

```

```

averiar()
begin
    self.isOutOfOrder := true;
end
pre notOutOfOrder: not self.isOutOfOrder

statemachines
psm TrenFuncionando
states
    s : initial
    EstarEnEstacion
    CirculandoEnVia
    Stopped
transitions
    s->EstarEnEstacion
    EstarEnEstacion->Stopped { [self.isOutOfOrder or self.viaDelTren.viaOcupada] stop()}
    Stopped->CirculandoEnVia { [not self.isOutOfOrder and self.isMoving] move()}
    EstarEnEstacion->CirculandoEnVia { [not self.isOutOfOrder and not self.isMoving]
move()}
    CirculandoEnVia->EstarEnEstacion { [not self.isOutOfOrder and self.isMoving and
self.viaDelTren.viaOcupada and self.estacionDelTren->size() = 0] stop()}
    EstarEnEstacion->EstarEnEstacion { [self.isOutOfOrder] stop()}
end

psm TrenAveriado
states
    s : initial
    Averiado
    TrenFuncional
transitions
    s->TrenFuncional
    TrenFuncional->Averiado { [not self.isOutOfOrder] averiar()}
    Averiado->Averiado { [self.isOutOfOrder] stop() }
    Averiado->TrenFuncional { [self.isOutOfOrder] reparar() }
end
end

class Linea
end

class Via
    attributes
        viaOcupada : Boolean derive = self.trenEnVia->size()>0
end

class Segmento
end
-- association trenVia between
Tren[0..1] role trenEnVia
Via[0..1] role viaDelTren
end

association trenEstacion between
Tren[0..*] role trenEnEstacion
Estacion[0..1] role estacionDelTren
end

association trenLinea between
Tren[0..*] role trenDeLinea
Linea[0..1] role lineaDeTren
end

aggregation segmentoPerteneceALinea between
Linea[1..*] role lineaDelSegmento
Segmento[1..*] role segmentoDeLinea
end

composition viasDelSegmento between
Segmento[1] role segmentoTieneVia
Via[2] role viaPerteneceASegmento
end

association estacionVialda between
Estacion[1] role estacionIda

```

```

Via[1..*] role viaIda
end

association estacionViaVuelta between
Estacion[1] role estacionVuelta
Via[1..*] role viaVuelta
end

-- constraints
-- Un segmento debe de estar compuesto por dos vias
context Segmento inv segmentoTieneDosVias:
    self.viaPerteneceASegmento->iterate(vias ; sum : Integer = 0 | sum+1) = 2

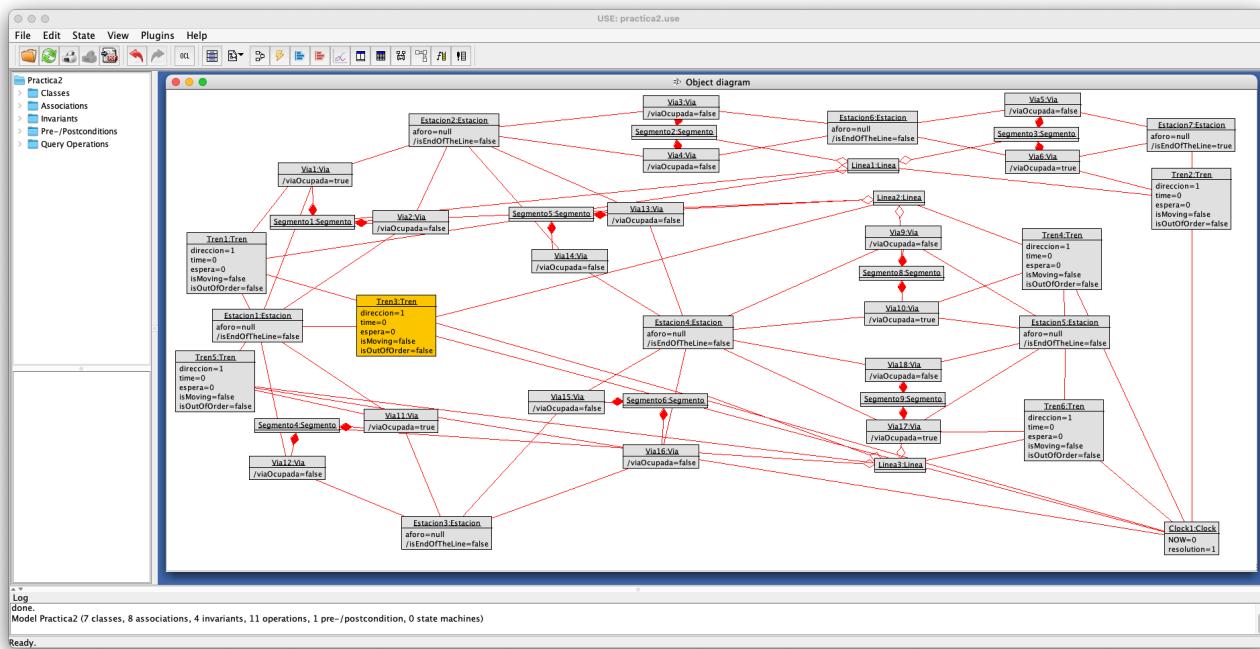
-- Numero de lineas mayor que cero
context Linea inv lineasMayorQueCero:
    Linea.allInstances()->size() > 0

-- Todas las lineas deben de tener al menos un segmento asociado
context Linea inv lineasTienenAlMenosUnSegmentoAsociado:
    Linea.allInstances().segmentoDeLinea <> null

-- Las estaciones tienen un numero par de vias
context Estacion inv estacionTieneNumeroParDeVias:
    (self.viaVuelta->iterate(vias ; sum : Integer = 0 | sum+1) + self.viaIda->iterate(vias ; sum :
Integer = 0 | sum+1)).mod(2) = 0

```

APARTADO D



0 segundos

Nuestro reloj tiene como unidad de tiempo los segundos. Se nos pide que avancemos dos minutos así que tras ejecutar !Clock1.run(120) la situación es la siguiente:

Tren1: Ha esperado 1 minuto en la Estacion1 y ahora se dirige a Estacion2 mediante Via1

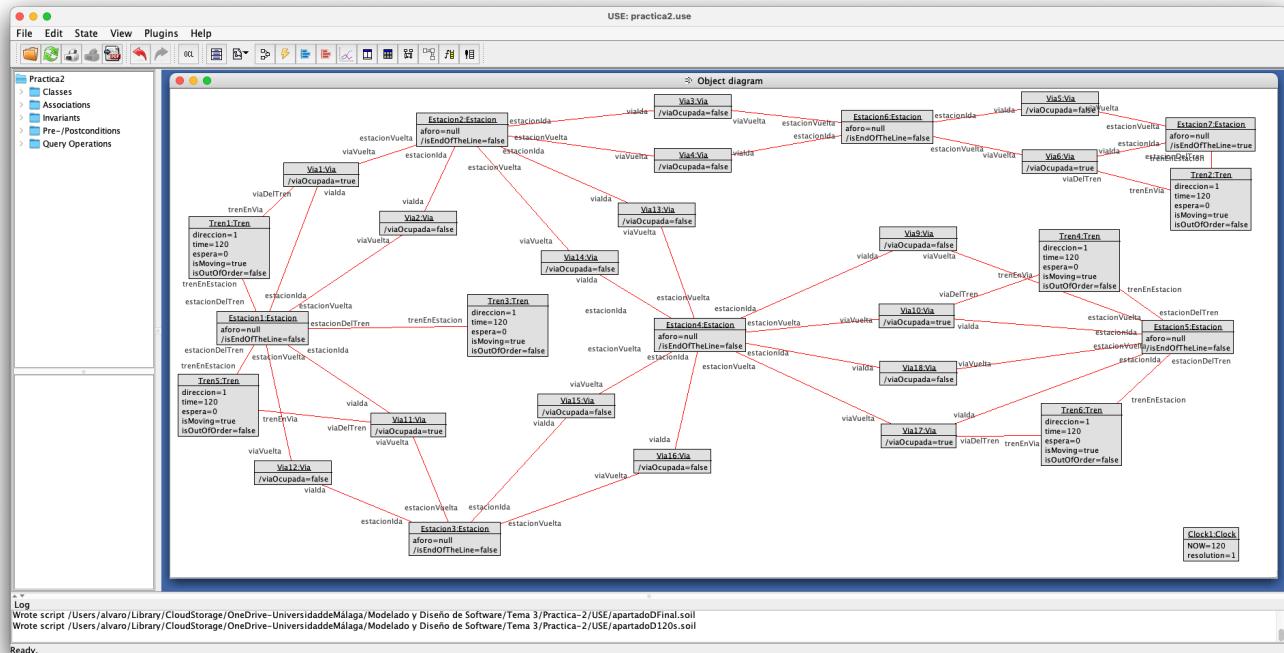
Tren2: Ha esperado 1 minuto en la Estacion7 y ahora se dirige a Estacion6 mediante Via6

Tren3: Está esperando en Estacion1 ha que se libere la Via1 para dirigirse a Estacion2

Tren4: Ha esperado 1 minuto en la Estacion5 y ahora se dirige a Estacion4 mediante Via9

Tren5: Ha esperado 1 minuto en la Estacion1 y ahora se dirige a Estacion3 mediante Via11

Tren6: Ha esperado 1 minuto en la Estacion5 y ahora se dirige a Estacion4 mediante Via17



120 segundos

Añadimos el .soil:

```

!insert (Estacion4,Via10) into estacionViaVuelta
!insert (Estacion5,Via9) into estacionVialda
!insert (Estacion5,Via10) into estacionViaVuelta
!insert (Estacion2,Via14) into estacionViaVuelta
!insert (Estacion2,Via13) into estacionVialda
!insert (Estacion4,Via13) into estacionVialda
!insert (Estacion4,Via14) into estacionViaVuelta
!insert (Estacion1,Via11) into estacionVialda
!insert (Estacion1,Via12) into estacionViaVuelta
!insert (Estacion3,Via11) into estacionVialda
!insert (Estacion3,Via12) into estacionViaVuelta
!insert (Estacion3,Via15) into estacionVialda
!insert (Estacion4,Via15) into estacionVialda
!insert (Estacion4,Via16) into estacionViaVuelta
!insert (Estacion3,Via16) into estacionViaVuelta
!new Segmento
!insert (Segmento1,Via1) into viasDelSegmento
!insert (Segmento1,Via2) into viasDelSegmento
!new Segmento
!insert (Segmento2,Via3) into viasDelSegmento
!insert (Segmento2,Via4) into viasDelSegmento
!new Segmento
!insert (Segmento3,Via5) into viasDelSegmento
!insert (Segmento3,Via6) into viasDelSegmento
!new Segmento
!insert (Segmento4,Via11) into viasDelSegmento
!insert (Segmento4,Via12) into viasDelSegmento
!new Segmento
!insert (Segmento5,Via13) into viasDelSegmento
!insert (Segmento5,Via14) into viasDelSegmento
!new Segmento
!insert (Segmento6,Via15) into viasDelSegmento
!insert (Segmento6,Via16) into viasDelSegmento
!new Segmento
!insert (Segmento7,Via7) into viasDelSegmento
!insert (Segmento7,Via8) into viasDelSegmento
!new Segmento
!insert (Segmento8,Via9) into viasDelSegmento
!insert (Segmento8,Via10) into viasDelSegmento
!new Linea
!insert (Linea1, Segmento1) into segmentoPerteneceALinea

```

```

!insert (Linea1, Segmento2) into segmentoPerteneceALinea
!insert (Linea1, Segmento3) into segmentoPerteneceALinea
!new Linea
!insert (Linea2, Segmento1) into segmentoPerteneceALinea
!insert (Linea2, Segmento5) into segmentoPerteneceALinea
!insert (Linea2, Segmento8) into segmentoPerteneceALinea
!new Linea
!insert (Linea3, Segmento4) into segmentoPerteneceALinea
!insert (Linea3, Segmento6) into segmentoPerteneceALinea
!insert (Linea3, Segmento8) into segmentoPerteneceALinea
!new Tren
!insert (Tren1,Linea1) into trenLinea
!insert (Tren1,Estacion1) into trenEstacion
!new Tren
!insert (Tren2,Linea1) into trenLinea
!insert (Tren2,Estacion7) into trenEstacion
!new Tren
!insert (Tren3,Linea2) into trenLinea
!insert (Tren3,Estacion1) into trenEstacion
!new Tren
!insert (Tren4,Linea2) into trenLinea
!insert (Tren4,Estacion5) into trenEstacion
!new Tren
!insert (Tren5,Estacion1) into trenEstacion
!insert (Tren5,Linea3) into trenLinea
!new Tren
!insert (Tren6,Estacion5) into trenEstacion
!insert (Tren6,Linea3) into trenLinea
!new Clock
!insert (Clock1,Tren1) into Time
!insert (Clock1,Tren5) into Time
!insert (Clock1,Tren6) into Time
!insert (Clock1,Tren4) into Time
!insert (Clock1,Tren2) into Time
!insert (Clock1,Tren3) into Time

```

```

!Clock1.run(20)
!insert (Tren1,Via1) into trenVia
!insert (Tren2,Via6) into trenVia
!insert (Tren5,Via11) into trenVia
!insert (Tren4,Via9) into trenVia
!insert (Tren6,Via10) into trenVia
!insert (Tren3,Via7) into trenVia
!destroy Segmento7; destroy Via7; destroy Via8
!insert (Tren3,Via1) into trenVia
!delete (Tren6,Via10) from trenVia
!insert (Tren6,Via9) into trenVia
!delete (Tren4,Via9) from trenVia
!delete (Tren6,Via9) from trenVia
!insert (Tren6,Via10) into trenVia
!insert (Tren4,Via10) into trenVia
!delete (Linea3 , Segmento8 ) from
segmentoPerteneceALinea
!new Via
!new Segmento
!new Via
!insert (Estacion4,Via18) into estacionVialda
!insert (Estacion4,Via17) into estacionViaVuelta
!insert (Estacion5,Via18) into estacionVialda
!insert (Estacion5,Via17) into estacionViaVuelta
!insert (Segmento9,Via18) into viasDelSegmento
!insert (Segmento9,Via17) into viasDelSegmento
!insert (Linea3 , Segmento9 ) into
segmentoPerteneceALinea
!delete (Tren6,Via10) from trenVia
!insert (Tren6,Via17) into trenVia
!delete (Estacion1,Via1) from estacionVialda
!insert (Estacion1,Via1) into estacionViaVuelta
!delete (Estacion1,Via2) from estacionViaVuelta
!insert (Estacion1,Via2) into estacionVialda
!delete (Estacion1,Via1) from estacionViaVuelta
!insert (Estacion1,Via1) into estacionVialda
!delete (Estacion2,Via1) from estacionVialda
!insert (Estacion2,Via1) into estacionViaVuelta
!delete (Estacion1,Via2) from estacionVialda
!insert (Estacion1,Via2) into estacionViaVuelta
!delete (Estacion2,Via2) from estacionViaVuelta
!insert (Estacion2,Via2) into estacionVialda

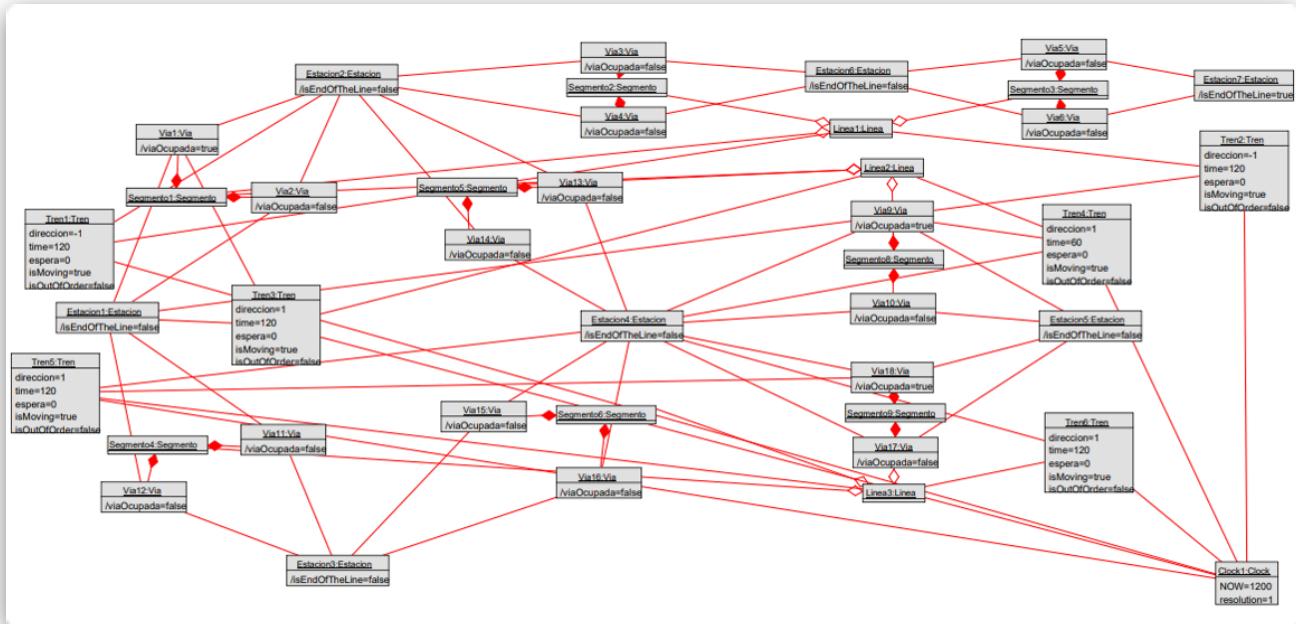
```

```

!delete (Estacion6,Via3) from estacionVialda
!insert (Estacion6,Via3) into estacionViaVuelta
!delete (Estacion6,Via4) from estacionViaVuelta
!insert (Estacion6,Via4) into estacionVialda
!delete (Estacion7,Via5) from estacionVialda
!insert (Estacion7,Via5) into estacionViaVuelta
!delete (Estacion6,Via6) from estacionVialda
!insert (Estacion6,Via6) into estacionViaVuelta
!delete (Estacion7,Via6) from estacionViaVuelta
!insert (Estacion7,Via6) into estacionVialda
!delete (Estacion4,Via13) from estacionVialda
!insert (Estacion4,Via13) into estacionViaVuelta
!delete (Estacion4,Via14) from estacionViaVuelta
!insert (Estacion4,Via14) into estacionVialda
!delete (Estacion5,Via9) from estacionVialda
!insert (Estacion5,Via9) into estacionViaVuelta
!delete (Estacion5,Via10) from estacionViaVuelta
!insert (Estacion5,Via10) into estacionVialda
!delete (Estacion5,Via18) from estacionVialda
!insert (Estacion5,Via18) into estacionViaVuelta
!delete (Estacion5,Via17) from estacionViaVuelta
!insert (Estacion5,Via17) into estacionVialda
!delete (Estacion3,Via11) from estacionVialda
!insert (Estacion3,Via11) into estacionViaVuelta
!delete (Estacion3,Via12) from estacionViaVuelta
!insert (Estacion3,Via12) into estacionVialda
!delete (Estacion4,Via15) from estacionVialda
!insert (Estacion4,Via15) into estacionViaVuelta
!delete (Estacion4,Via16) from estacionViaVuelta
!insert (Estacion4,Via16) into estacionVialda
!delete (Tren3,Via1) from trenVia
!Clock1.NOW := 0
!Tren1.time := 0
!Tren2.time := 0
!Tren3.time := 0
!Tren4.time := 0
!Tren5.time := 0
!Tren6.time := 0
!Clock1.run(12)
!Clock1.run(108)

```

APARTADO E



Para esta ejecución debemos hacer 20 instancias que nosotros interpretaremos como 20 min, como nuestro reloj va con segundos haremos !Clock1.run(60) por cada min que queramos.

- Ejecutamos el !Clock1.run(180) para hacer transcurrir el tiempo 3 minutos, y cuando llegue a ese instante averiamos un tren con el comando !Tren1.averiar(), esta orden dejara fuera de servicio al Tren1 y se le sacará de la vía en la que este.
- Volvemos a ejecutar el !Clock1.run(180), y esta vez repararemos el Tren1 como nos pide el enunciado con el comando !Tren1.reparar(), esta orden indicará que ya no esta fuera de servicio y por lo tanto que puede empezar a "funcionar".
- Esta vez haremos un !Clock1.run(120) para hacer que pasen 2 instantes o minutos, para averiar otro tren con el comando !Tren4.averiar().
- Como este tren se reparará en el instante 12 hacemos !Clock1.run(240), y posteriormente !Tren4.reparar().
- Por último para acabar con la ejecución debemos llegar al instante 0 min 20, por lo que solo nos queda hacer un !Clock1.run(480).

Añadimos también el .soil:


```

!insert(Segmento8,Via10) into viasDelSegmento
!new Linea
!insert(Linea1, Segmento1) into segmentoPerteneceALinea
!insert(Linea1, Segmento2) into segmentoPerteneceALinea
!insert(Linea1, Segmento3) into segmentoPerteneceALinea
!new Linea
!insert(Linea2, Segmento1) into segmentoPerteneceALinea
!insert(Linea2, Segmento5) into segmentoPerteneceALinea
!insert(Linea2, Segmento8) into segmentoPerteneceALinea
!new Linea
!insert(Linea3, Segmento4) into segmentoPerteneceALinea
!insert(Linea3, Segmento6) into segmentoPerteneceALinea
!insert(Linea3, Segmento8) into segmentoPerteneceALinea
!new Tren
!insert(Tren1,Linea1) into trenLinea
!insert(Tren1,Estacion1) into trenEstacion
!new Tren
!insert(Tren2,Linea1) into trenLinea
!insert(Tren2,Estacion7) into trenEstacion
!new Tren
!insert(Tren3,Linea2) into trenLinea
!insert(Tren3,Estacion1) into trenEstacion
!new Tren
!insert(Tren4,Linea2) into trenLinea
!insert(Tren4,Estacion5) into trenEstacion
!new Tren
!insert(Tren5,Estacion1) into trenEstacion
!insert(Tren5,Linea3) into trenLinea
!new Tren
!insert(Tren6,Estacion5) into trenEstacion
!insert(Tren6,Linea3) into trenLinea

!new Clock
!insert(Clock1,Tren1) into Time
!insert(Clock1,Tren5) into Time
!insert(Clock1,Tren6) into Time
!insert(Clock1,Tren4) into Time
!insert(Clock1,Tren2) into Time
!insert(Clock1,Tren3) into Time
!Clock1.run(20)
!insert(Tren1,Via1) into trenVia
!insert(Tren2,Via6) into trenVia
!insert(Tren5,Via11) into trenVia
!insert(Tren4,Via9) into trenVia
!insert(Tren6,Via10) into trenVia
!insert(Tren3,Via7) into trenVia
!destroy Segmento7; destroy Via7; destroy Via8
!insert(Tren3,Via1) into trenVia
!delete(Tren6,Via10) from trenVia
!insert(Tren6,Via9) into trenVia
!delete(Tren4,Via9) from trenVia
!delete(Tren6,Via9) from trenVia
!insert(Tren6,Via10) into trenVia
!insert(Tren4,Via10) into trenVia
!delete(Linea3, Segmento8) from segmentoPerteneceALinea
!new Via
!new Segmento
!new Via
!insert(Estacion4,Via18) into estacionVialda
!insert(Estacion4,Via17) into estacionViaVuelta
!insert(Estacion5,Via18) into estacionVialda
!insert(Estacion5,Via17) into estacionViaVuelta
!insert(Segmento9,Via18) into viasDelSegmento
!insert(Segmento9,Via17) into viasDelSegmento
!insert(Linea3, Segmento9) into segmentoPerteneceALinea
!delete(Tren6,Via10) from trenVia
!insert(Tren6,Via17) into trenVia
!delete(Estacion1,Via1) from estacionVialda
!insert(Estacion1,Via1) into estacionViaVuelta

```

```

!delete (Estacion1,Via2) from estacionViaVuelta
!insert (Estacion1,Via2) into estacionVialda
!delete (Estacion1,Via1) from estacionViaVuelta
!insert (Estacion1,Via1) into estacionVialda
!delete (Estacion2,Via1) from estacionVialda
!insert (Estacion2,Via1) into estacionViaVuelta
!delete (Estacion1,Via2) from estacionVialda
!insert (Estacion1,Via2) into estacionViaVuelta
!delete (Estacion2,Via2) from estacionViaVuelta
!insert (Estacion2,Via2) into estacionVialda
!delete (Estacion6,Via3) from estacionVialda
!insert (Estacion6,Via3) into estacionViaVuelta
!delete (Estacion6,Via4) from estacionViaVuelta
!insert (Estacion6,Via4) into estacionVialda
!delete (Estacion7,Via5) from estacionVialda
!insert (Estacion7,Via5) into estacionViaVuelta
!delete (Estacion6,Via6) from estacionVialda
!insert (Estacion6,Via6) into estacionViaVuelta
!delete (Estacion7,Via6) from estacionViaVuelta
!insert (Estacion7,Via6) into estacionVialda
!delete (Estacion4,Via13) from estacionVialda
!insert (Estacion4,Via13) into estacionViaVuelta
!delete (Estacion4,Via14) from estacionViaVuelta
!insert (Estacion4,Via14) into estacionVialda
!delete (Estacion5,Via9) from estacionVialda
!insert (Estacion5,Via9) into estacionViaVuelta
!delete (Estacion5,Via10) from estacionViaVuelta
!insert (Estacion5,Via10) into estacionVialda
!delete (Estacion5,Via18) from estacionVialda
!insert (Estacion5,Via18) into estacionViaVuelta
!delete (Estacion5,Via17) from estacionViaVuelta
!insert (Estacion5,Via17) into estacionVialda
!delete (Estacion3,Via11) from estacionVialda
!insert (Estacion3,Via11) into estacionViaVuelta
!delete (Estacion3,Via12) from estacionViaVuelta
!insert (Estacion3,Via12) into estacionVialda
!delete (Estacion4,Via15) from estacionVialda
!insert (Estacion4,Via15) into estacionViaVuelta
!delete (Estacion4,Via16) from estacionViaVuelta
!insert (Estacion4,Via16) into estacionVialda
!Clock1.NOW := 0
!Tren1.time := 0
!Tren2.time := 0
!Tren3.time := 0
!Tren4.time := 0
!Tren5.time := 0
!Tren6.time := 0
!Clock1.run(180)
!Tren1.averiar()
!Clock1.run(180)
!Tren1.reparar()
!Clock1.run(120)
!Tren4.averiar()
!Clock1.run(240)
!Tren4.reparar()
!Clock1.run(480)

```