

PiNet

Manual de Usuario

Mauricio Holguín Londoño

Universidad Tecnológica de Pereira

Juan Felipe Grajales Gonzalez

Universidad Tecnológica de Pereira

Andrés Escobar Mejía

Universidad Tecnológica de Pereira

14 de noviembre de 2017

Índice

1. INTRODUCCIÓN	1
2. INSTALACIÓN	3
2.1. REQUERIMIENTOS	3
2.2. PROCEDIMIENTO PARA LA EJECUCIÓN	3
3. PRESENTACIÓN DE LA INTERFAZ	5
3.1. INTERFAZ PRINCIPAL	5
3.1.1. Panel de objetos	6
3.1.2. Panel de acciones	8
3.1.3. Panel de propiedades	12
3.1.4. Área de trabajo	15
4. USO DE LA INTERFAZ Y EJEMPLOS	17
4.1. EJEMPLO 1	17
4.2. EJEMPLO 2	27
4.3. EJEMPLO 3	31
4.4. EJEMPLO 4	37
4.5. CARPETA DE EJEMPLOS	46

5. MATERIALES Y MÉTODOS EMPLEADOS	47
5.1. CARACTERÍSTICAS DE PiNet	47
5.2. INTRODUCCIÓN A LAS REDES DE PETRI	48
5.2.1. Matrices de Incidencia	51
5.2.2. Vector de marcado	52
5.2.3. Propiedades de las redes	53
5.2.4. Invariantes de marcado y de disparo	53
5.3. MODELOS UML	54
6. EVALUACIÓN DE LA CALIDAD DEL SOFTWARE	59
6.1. INTRODUCCIÓN	59
6.2. ANÁLISIS CALIDAD DE SOFTWARE PiNet	66
6.2.1. Robustez	66
6.2.2. Extendibilidad	67
6.2.3. Desempeño	67
6.2.4. Usabilidad o amigable al usuario	68
6.2.5. Integridad	68
6.2.6. Portabilidad	69
6.2.7. Compatibilidad	69
6.2.8. Mantenibilidad	69
6.2.9. Documentación	69

6.2.10. Grado de invención o de innovación	70
--	----

Índice de figuras

1.	Ejecutable PiNet.	4
2.	Interfaz principal	5
3.	Panel de objetos	6
4.	Lugar	7
5.	Transición	7
6.	Arco	8
7.	Marca	8
8.	Panel de acciones	9
9.	Botón de ejecutar.	9
10.	Botón de eliminar.	9
11.	Botón de configuración.	10
12.	Panel de configuración.	10
13.	Opción disponible para los lugares (No-Temp y T-Temp).	11
14.	Opción disponible para las transiciones (T-Temp).	11
15.	Opción disponible para los lugares (P-Temp).	11
16.	Botón de ayuda.	11
17.	Parámetros de red.	12
18.	Botón de parámetros.	12

19.	Ventana de parámetros.	13
20.	Botón de incidencia.	13
21.	Ventana de incidencias.	14
22.	Botón de transformación.	14
23.	Visualización de los vectores anuladores.	14
24.	Área de trabajo.	15
25.	Botones para almacenar y cargar redes.	15
26.	Lugar posicionado en el área de trabajo.	18
27.	Etiqueta de elementos.	19
28.	Transiciones posicionadas dentro del área de trabajo.	20
29.	Selección de elemento a ser conectado.	21
30.	Conexión de elementos.	22
31.	Red de Petri disponible.	23
32.	Evolución de una red.	24
33.	Propiedades de la red.	24
34.	Matrices de la red.	25
35.	Anuladores de la red.	25
36.	Ventana de diálogo para almacenamiento de una red.	25
37.	Dirección para la red almacenada dentro del equipo anfitrión.	26
38.	Elección de red T-Temporizada.	26

39.	Definición de tiempo a una transición.	27
40.	Ejemplo de recurso compartido.	27
41.	Elección de ruta realizada por el sistema.	28
42.	Habilitación de transiciones para un segundo disparo.	29
43.	Resultado del segundo disparo de la red.	29
44.	Elección del modo manual.	30
45.	Habilitación de la red para realizar un disparo en modo manual.	30
46.	Ejecución de un disparo manual sobre la red.	31
47.	Arquitectura de Recurso Compartido.	32
48.	Transiciones para alimentación habilitadas.	33
49.	Alimentación de actividad desde T0.	34
50.	Transiciones 2 y 3 disponibles.	35
51.	Actividad en el lugar 2.	36
52.	Recurso compartido nuevamente disponible.	37
53.	RdP Conservativa.	38
54.	RdP conservativa en PiNet.	38
55.	Incidencia Previa y Posterior obtenidas en [3].	39
56.	Incidencia Previa y Posterior obtenidas en PiNet.	39
57.	Matriz de incidencia obtenida en [3].	40
58.	Matriz de incidencia obtenida en PiNet.	40

59.	Vectores anuladores obtenidos en PiNet.	41
60.	Elección modo P-Temporizado.	41
61.	Transición no sensibilizada.	42
62.	Red sensibilizada.	42
63.	Adición de tiempo al lugar 2.	43
64.	Disparo de T0.	44
65.	Disparo de T2.	44
66.	Disparo de T1.	44
67.	Disparo de T3.	45
68.	Disparo de T4.	45
69.	Segundo disparo de T4.	45
70.	Diagrama de clases.	55
71.	Diagrama de componentes.	56
72.	Diagrama de paquetes.	57
73.	Diagrama de actividades.	58
74.	Diagrama de secuencias.	59
75.	Certificación del impacto del producto PiNet.	71

Índice de cuadros

1.	Cuadro comparativo de plataformas	49
2.	Cuadro comparativo de estándares.	65

1. INTRODUCCIÓN

Las redes de Petri (RdP) constituyen una familia de formalismos bien conocidos dentro de las aplicaciones de ingeniería para el modelado, análisis y síntesis de sistemas discretos [1]. En los sistemas discretos se asume que las variables de estado, o memorización del pasado dinámico, se pueden modelar como discretas al codificarlas, según los formalismos, con diversos alfabetos o numéricamente, por ejemplo.

Las RdP se presentan como una poderosa herramienta capaz de modelar de forma gráfica y matemática todos estos sistemas de diferentes naturalezas. Su representación gráfica permite una visualización clara de los sistemas, además de facilitar su posterior descripción mediante otras metodologías tales como: máquinas de estados, diagramas de flujo, gráficos marcados, diagramas de bloques, diagramas escalera, diagramas de descripción secuencial, lógicas temporales, lenguajes naturales, etc.

Para emplear una RdP en el modelamiento de una clase de aplicaciones, se procede a dotarla de una interpretación. Es decir, asociar una significación física a las condiciones de evolución de la red, así como definir las acciones generadas por dicha evolución. Para cumplir con tal fin se presenta el software PiNet, el cual permite modelar y analizar sistemas de control basados en sistemas discretos que exhiben evoluciones concurrentes, todo ello por medio de las Redes de Petri. PiNet permite diseñar y analizar distintos aspectos de una red, tal como su diseño e interpretación gráfica, análisis y síntesis matricial, análisis de propiedades distintivas y simulación de una red con base en tanto sistemas No temporizados como en sistemas Temporizados desde una perspectiva de los lugares y/o las transiciones.

PiNet es un software portable multiplataforma desarrollado en el lenguaje de programación Python, bajo el paradigma de la programación orientada a objetos. Este lenguaje, además de ofrecer módulos que permiten el tratamiento de arreglos y cálculos matemáticos, posee la librería Pygame, la cual permite el desarrollo de software que se basa en módulos de videojuegos en 2D, y siendo ideal para la creación de interfaces que requieren de una mayor interacción entre el usuario y el sistema, como es el caso de PiNet. Dentro de este módulo se encuentran características de especial interés, como lo son los eventos por ratón y teclado, los cuales son piezas esenciales en el desarrollo de entornos gráficos interactivos.

Este manual no tiene como finalidad ser un documento exhaustivo acerca de las operaciones realizadas internamente por la interfaz, como tampoco pretende ser un documento guía para el estudio de las Redes de Petri. En su lugar, se describe el uso de la interfaz de usuario, la teoría de fundamento utilizada, los modelos comportamentales que rigen el diseño y el respectivo análisis de calidad del software.

2. INSTALACIÓN

2.1. REQUERIMIENTOS

Antes de iniciar con la instalación de **PiNet**, compruebe los siguientes requisitos:

- Sistema Operativo Windows 7/8/10 (64 bits) o Sistema Operativo Ubuntu 14 o superior.
- Intel Core 2 Duo @ 2.0Ghz (o equivalente).
- Memoria RAM de 4GB mínimo.
- 500MB de espacio libre en el disco de almacenamiento.
- Visor de archivos PDF.

2.2. PROCEDIMIENTO PARA LA EJECUCIÓN

Para ejecutar **PiNet** se debe insertar el CD de distribución en la unidad CD-ROM. Se recomienda iniciar verificando los contenidos de los archivos *leame.txt* y *licencia.txt* disponibles en el CD de distribución.

PiNet es un software portable, esto quiere decir que no es intrusivo para el sistema operativo, por tanto no se hace necesaria la instalación de éste dentro del sistema de almacenamiento interno. Para la ejecución basta con copiar los elementos del CD de distribución dentro de un lugar deseado en el equipo anfitrión. Este paso es indiferente del sistema operativo en que se desee ejecutar.

A continuación, se definen los pasos a seguir dependiendo del sistema operativo en que se desee ejecutar.

- **Windows:** Para sistemas operativos Windows (7/8.1/10), basta con presionar doble clic sobre el archivo **PiNet.exe**, similar al que se observa en la Figura 1. Para comodidad del usuario, es posible añadir este ejecutable como un acceso directo, o bien sea dentro de la barra de inicio.



Figura 1: Ejecutable PiNet.

- **Ubuntu:** Para distribuciones del GNU Linux (Ubuntu, Devian, entre otras), se hace necesaria la identificación de la ruta absoluta en que se encuentra el archivo, esto con el fin de ejecutar **PiNet** desde la terminal. El siguiente es un ejemplo de ejecución del software desde la terminal, para el caso en que la carpeta contenedora se almacene dentro de la sección **Home** del sistema operativo.

```
cd/Home/PiNet
```

```
sudo/.PiNet
```

3. PRESENTACIÓN DE LA INTERFAZ

Verificados los requerimientos del sistema, es posible acceder a la interfaz por medio del icono del ejecutable de **PiNet** ya sea desde el equipo anfitrión o desde el CD de distribución, el cual se observa en la Figura 1. Una vez ejecutado el software se despliega la interfaz principal del sistema, tal como se observa en la Figura 2.

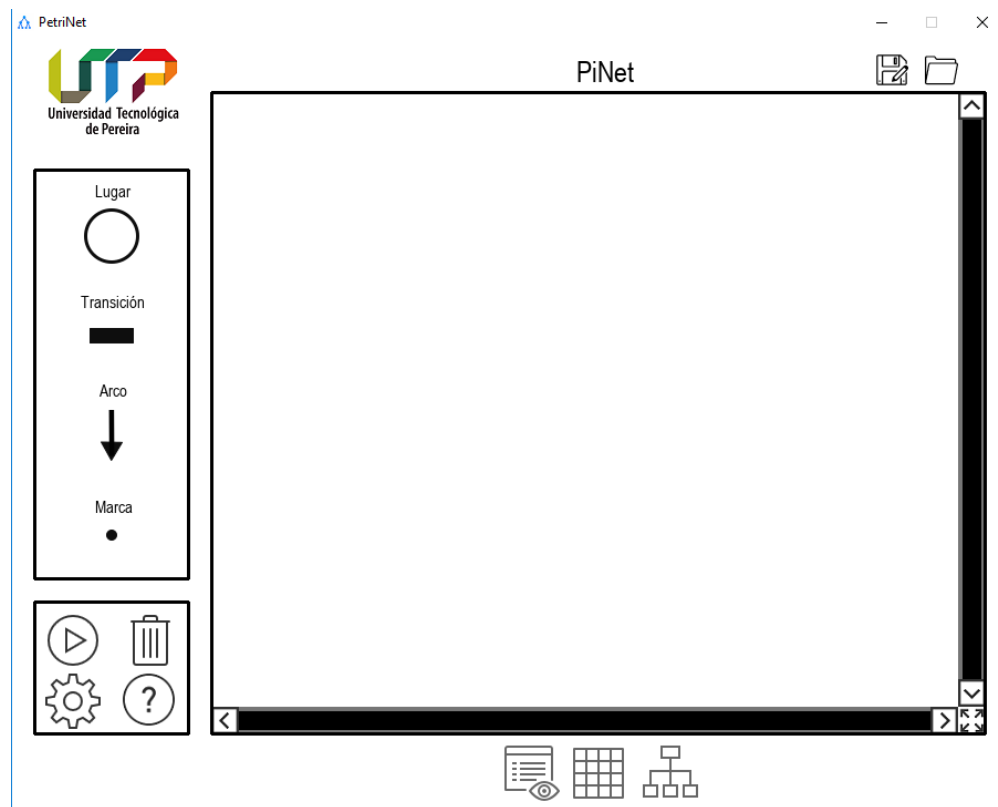


Figura 2: Interfaz principal

3.1. INTERFAZ PRINCIPAL

La interfaz principal se compone de cuatro elementos:

1. Panel de objetos

2. Panel de acciones
3. Panel de propiedades
4. Área de trabajo

3.1.1. Panel de objetos

La Figura 3 enseña el panel de objetos, dentro del cual se encuentran las siguientes opciones:

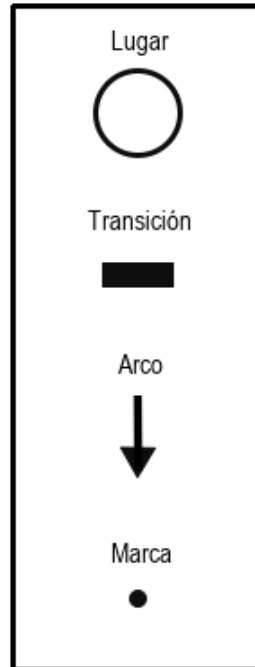


Figura 3: Panel de objetos

Lugar: Con el objeto **Lugar** es posible adicionar lugares dentro del área de trabajo. En la Figura 4 se observa dicho objeto.



Figura 4: Lugar

Transición: Con el objeto **Transición** es posible adicionar transiciones dentro del área de trabajo. Éstas por defecto se presentan en forma horizontal pero, por conveniencia y facilidad del usuario, es posible rotarlas a una posición vertical, lo cual se logra seleccionando el objeto y haciendo clic izquierdo sobre este. En caso de desear volver a la posición horizontal, se debe accionar nuevamente el botón izquierdo del ratón. En la Figura 5 se observa dicho objeto.

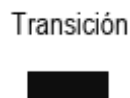


Figura 5: Transición

Arco: Con el objeto **Arco** es posible realizar conexiones entre lugares y transiciones que se encuentren dentro del área de trabajo. Las conexiones solo son posibles desde transición a lugar y viceversa, pero nunca entre objetos de la misma naturaleza (lugar con lugar o transición con transición). En caso de volver a dibujar una conexión que ya exista entre dos objetos dentro del área de trabajo, se toma como una adición de peso a la conexión ya existente. En la Figura 6 se observa dicho objeto.



Figura 6: Arco

Marca: Con el objeto **Marca** es posible adicionar marcado, o *tokens*, a los lugares y además es posible adicionar peso a las conexiones (arcos) existentes dentro del área de trabajo. Para el caso de los lugares basta con seleccionar el objeto marca y hacer click en el lugar respectivo, para los arcos es necesario realizar la misma acción, pero esta vez sobre la cabeza de flecha del arco. Mediante el botón izquierdo del ratón es posible adicionar marcas (en los lugares) o pesos (en los arcos), mediante el botón derecho es posible restar marcas o pesos. En la Figura 7 se observa dicho objeto.



Figura 7: Marca

Para dejar de usar cada uno de los objetos seleccionados basta con presionar la tecla *Escape*.

3.1.2. Panel de acciones

Esta sección de la interfaz permite realizar las siguientes operaciones: Ejecutar una red, eliminar elementos posicionados previamente sobre el área de trabajo, configurar el modo y tiempos de ejecución del sistema y visualizar el manual de usuario. La Figura 8 enseña el panel de acciones.

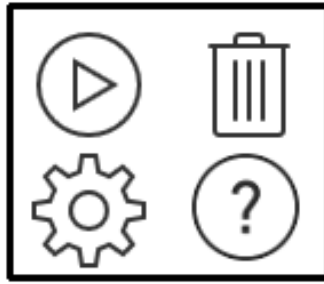


Figura 8: Panel de acciones

Ejecutar: Esta opción permite simular el comportamiento de la Red que se encuentre dentro del área de trabajo. En la Figura 9 se observa dicho botón.



Figura 9: Botón de ejecutar.

Eliminar: Esta herramienta permite borrar objetos que se encuentren sobre el área de trabajo. Una vez seleccionada esta opción, para borrar transiciones o lugares basta con posicionar el cursor sobre estos y hacer click, en el caso de los arcos es necesario realizar esta acción sobre la cabeza del arco. En la Figura 10 se observa dicho botón.



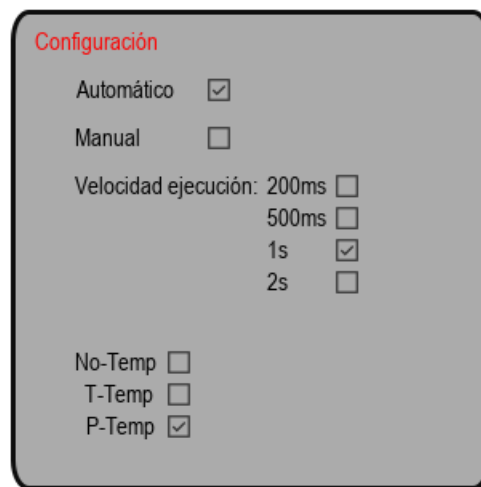
Figura 10: Botón de eliminar.

Configuración: Esta opción despliega una ventana emergente la cual permite elegir el modo en que se desea ejecutar la red, bien sea Manual o Automática, además de la duración en los tiempos de ejecución. Para el caso de la opción automática, es posible elegir el comportamiento del sistema ya sea como una red No-Temp

(No temporizada, que es la opción por defecto), T-Temp (Transiciones temporizadas), o P-Temp (lugares temporizados). En la Figura 11 se observa la opción de configuración, y en la Figura 12 el panel que se despliega por medio de esta acción.



Figura 11: Botón de configuración.



Configuración

Automático ☒

Manual ☐

Velocidad ejecución: 200ms ☐

500ms ☐

1s ☒

2s ☐

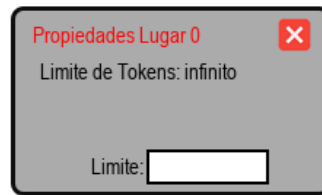
No-Temp ☐

T-Temp ☐

P-Temp ☒

Figura 12: Panel de configuración.

Dependiendo del tipo de temporización elegida, se habilita el parámetro de tiempo, ya sea para los lugares o para las transiciones. El tipo de temporización por defecto es el No-Temporizado, el cual omite las opciones de tiempos individuales, tanto para lugares como para transiciones. Para ingresar el parámetro individual de tiempo, basta con presionar el botón derecho del ratón sobre el elemento deseado (ya ubicado dentro del área de trabajo). Por defecto, los lugares siempre tienen activo el ingreso del límite de tokens. Las Figuras 13, 14 y 15 enseñan las opciones visibles dependiendo del tipo de Red temporizada que se encuentre activa.

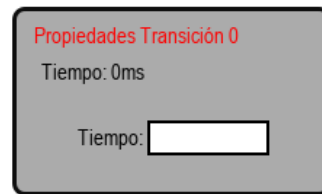


Propiedades Lugar 0

Limite de Tokens: infinito

Limite:

Figura 13: Opción disponible para los lugares (No-Temp y T-Temp).

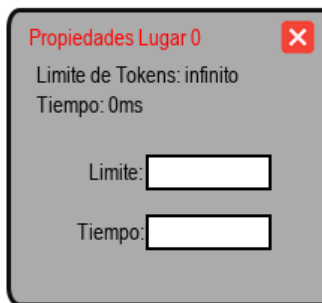


Propiedades Transición 0

Tiempo: 0ms

Tiempo:

Figura 14: Opción disponible para las transiciones (T-Temp).



Propiedades Lugar 0

Limite de Tokens: infinito

Tiempo: 0ms

Limite:

Tiempo:

Figura 15: Opción disponible para los lugares (P-Temp).

Ayuda: Esta opción despliega el presente manual de usuario en versión PDF. En la Figura 16 se observa dicho botón.



Figura 16: Botón de ayuda.

3.1.3. Panel de propiedades

Dentro de esta sección de la interfaz se visualizan las propiedades de cada Red, como lo son sus matrices de incidencia, propiedades intrínsecas de la red y transformaciones a la que puede ser sometida. En la Figura 17 se observa este panel.



Figura 17: Parámetros de red.

Propiedades: Mediante esta opción se visualiza el cumplimiento, o no, de las propiedades de una red, como lo son: Red de Petri Ordinaria, Red de Petri Reversible, Red de Petri Viva, Red de Petri Pura, Red de Petri Limitada. En la Figura 18 se observa el botón respectivo y en la Figura 19 se enseña la ventana en que se detallan cada una de estas propiedades.



Figura 18: Botón de parámetros.

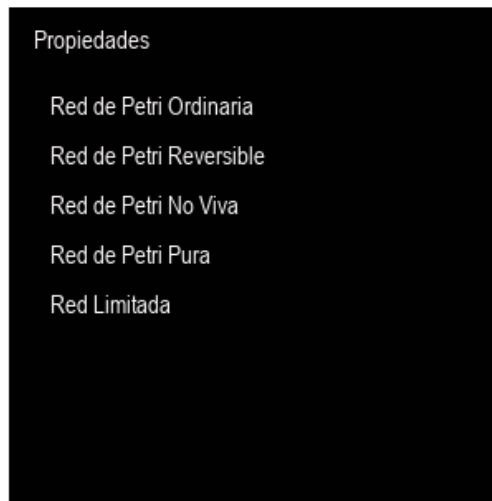


Figura 19: Ventana de parámetros.

Incidencia: Esta opción permite obtener, la matriz de incidencia, la matriz de incidencia previa, la matriz de incidencia posterior y la matriz dual. En la Figura 20 se enseña el botón referente a esta opción, y en la Figura 21 se observan las distintas matrices, así como la visualización de la matriz seleccionada.



Figura 20: Botón de incidencia.

☒ Incidencia +
 ☐ Incidencia -
 ☐ Dual

Lugares	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
P0	0	0	-1	0	0	0	0	0	0	0	0	1	0	0	0
P1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	-2	0	0	0	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	6	-1	0	-1	0
P4	0	0	0	0	-4	0	0	0	0	0	0	0	0	0	0
P5	0	0	0	-1	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	0	0
P7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P8	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
P9	0	0	0	0	0	-1	-1	0	0	0	0	0	0	0	0
P10	0	0	0	0	0	-1	-4	0	0	0	0	0	0	-1	-1
P11	0	0	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	0	0
P13	-4	0	0	0	0	0	0	0	0	0	0	0	-1	0	0
P14	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0
P15	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0

Figura 21: Ventana de incidencias.

Transformaciones: Esta acción permite obtener los valores para el anulador derecho y anulador izquierdo. En la Figura 22 se observa el botón que permite acceder a dicha opción y en la Figura 23 se observan los anuladores para una red previamente diseñada.



Figura 22: Botón de transformación.

Transformaciones

Anulador derecho:

1	1
---	---

Anulador izquierdo:

1	1
---	---

Figura 23: Visualización de los vectores anuladores.

3.1.4. Área de trabajo

Es dentro de esta sección de la interfaz en la que se realiza el posicionamiento de cada uno de los objetos, dando paso así al diseño de la red deseada por el usuario. Esta sección posee cinco botones, uno de los cuales permite expandir el área de trabajo (ubicado en la esquina inferior derecha de la ventana), siendo así posible la adición de más elementos. Los otros cuatro botones se encargan del desplazamiento de la sección de trabajo visible (movimiento de las barras de desplazamiento de forma vertical y horizontal).

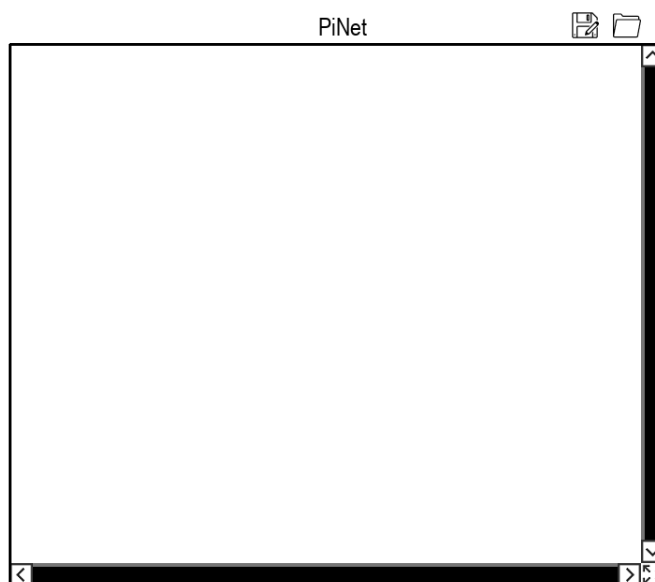


Figura 24: Área de trabajo.

Esta sección también ofrece la posibilidad de almacenar y cargar diseños a petición del usuario mediante los botones **Almacenar** y **Cargar**, los cuales se observan en la Figura 25.



Figura 25: Botones para almacenar y cargar redes.

Una vez se selecciona alguna de estas opciones, se presenta una ventana emergente que permite almacenar o cargar una red, según lo deseado por el usuario.

4. USO DE LA INTERFAZ Y EJEMPLOS

4.1. EJEMPLO 1

Una vez se ejecuta el programa y se presenta la interfaz principal, es posible proceder a adicionar los elementos necesarios para componer una red. Para el presente ejemplo se inicia insertando el número total de lugares dentro del área de trabajo (cabe resaltar que el orden en que se agreguen los lugares o las transiciones no es de relevancia) mediante el accionamiento del botón derecho del ratón sobre el objeto **Lugar**. Ya con el elemento debidamente seleccionado, este se hará visible en todo momento siguiendo el desplazamiento del cursor dentro de la interfaz. El elemento solo puede ser posicionado dentro del área de trabajo, en caso de intentar algún otro movimiento este será obviado. Para ubicar el elemento basta con presionar nuevamente el botón derecho del mouse, luego de esto es posible seguir adicionando lugares dentro del área de trabajo y hasta que se elija algún nuevo elemento o se presione la tecla *Escape*. En la Figura 26 es posible observar un objeto debidamente posicionado dentro del área de trabajo y el cursor aún disponible para la adición de nuevos lugares (círculo parcialmente en el área de trabajo).

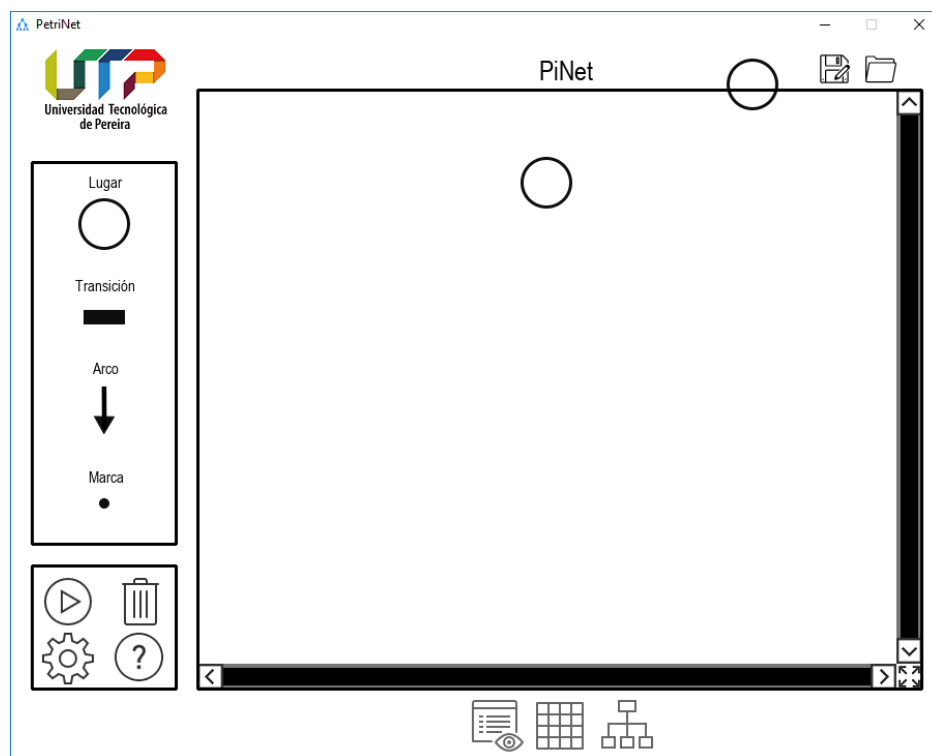


Figura 26: Lugar posicionado en el área de trabajo.

Para la adición de un segundo lugar sólo basta con presionar nuevamente el botón derecho del ratón sobre el área de trabajo. A medida que se agreguen nuevos elementos, cada uno de ellos incrementará en pasos de 1 el valor del identificador grupal, esto es, una etiqueta que permite diferenciar cada elemento por medio de una identidad única. Esta etiqueta se observa como un elemento tipo *tooltip* (descripción emergente), que aparece al pasar el cursor sobre algún elemento ubicado dentro del área de trabajo. En la Figura 27 se observa dicho caso.

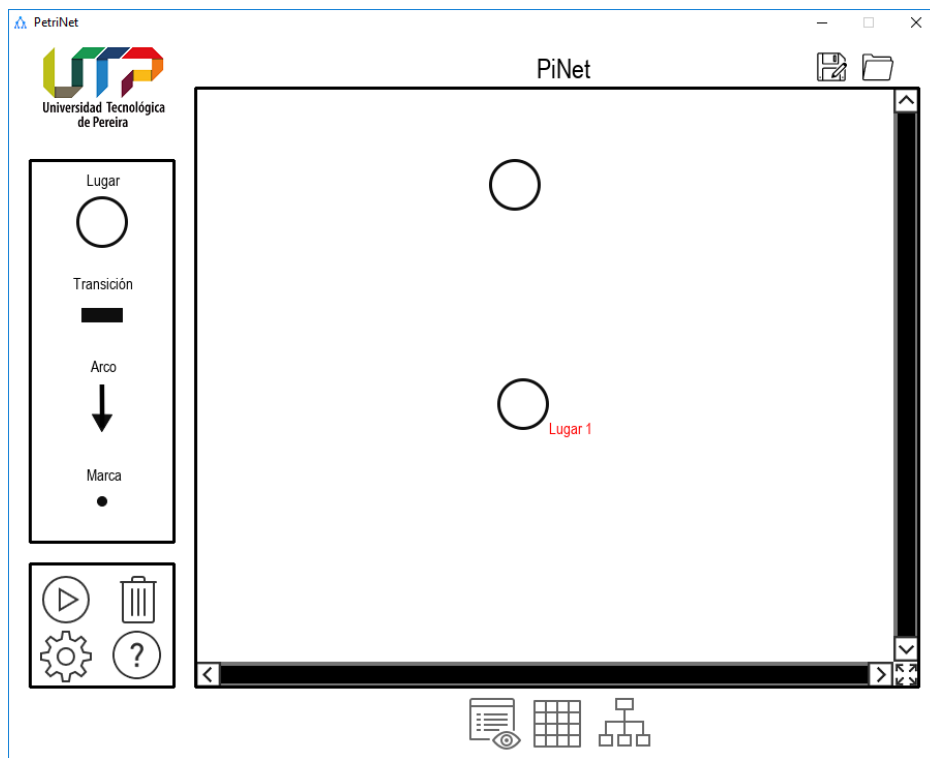


Figura 27: Etiqueta de elementos.

La mecánica para agregar transiciones es similar a la mencionada previamente para los lugares, aunque esta posee la particularidad de permitir la rotación del elemento, tal como se describe en la Sección 3.1.1. En la Figura 28 se observa la adición de dos transiciones.

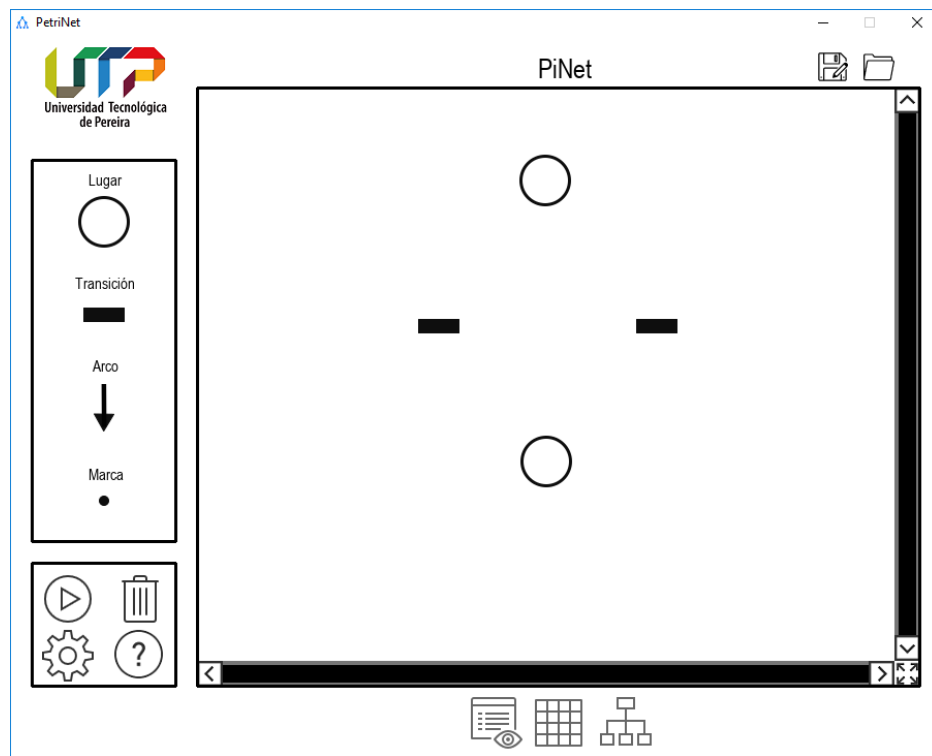


Figura 28: Transiciones posicionadas dentro del área de trabajo.

Con los elementos necesarios dentro del área de trabajo, ahora resta unirlos por medio de la herramienta **Arco**. Luego de seleccionar esta herramienta, se elige el elemento inicial a conectar por medio del botón derecho del ratón, lo cual da lugar a un cambio de color del elemento seleccionado (a modo de señalar) y a la creación del arco, tal como se observa en la Figura 29.

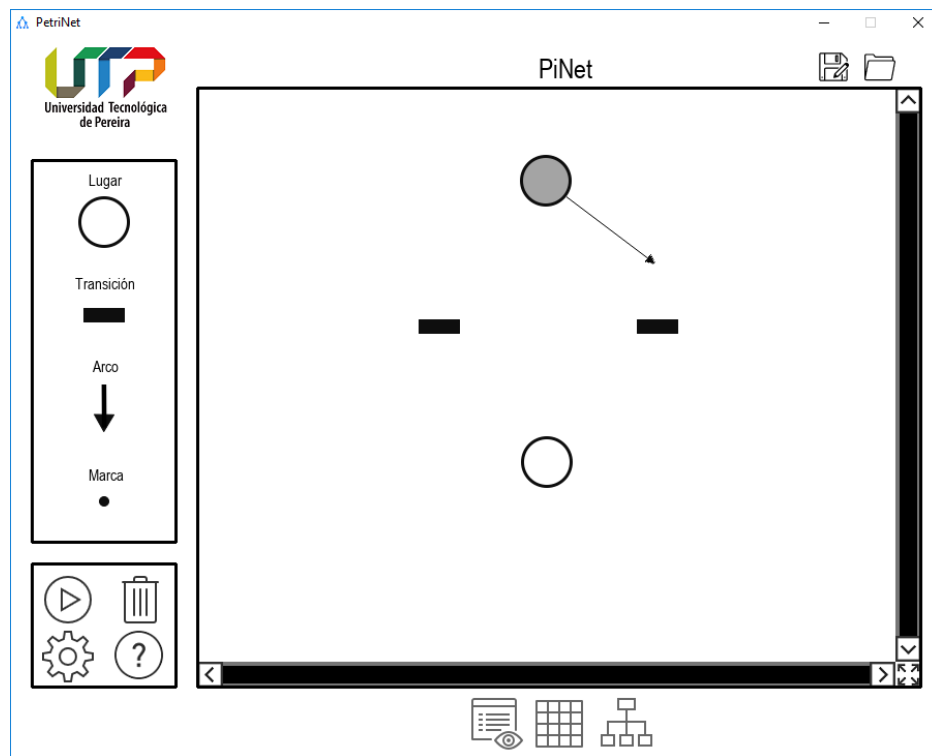


Figura 29: Selección de elemento a ser conectado.

La cabeza del arco se desplaza a medida que el puntero también lo hace, es señal de espera a que el usuario elija el elemento final de la conexión, o hasta que se presione la tecla de *Escape*, con lo cual se cancela la conexión. En la Figura 30 se observan los elementos debidamente conectados, dando lugar a una red.

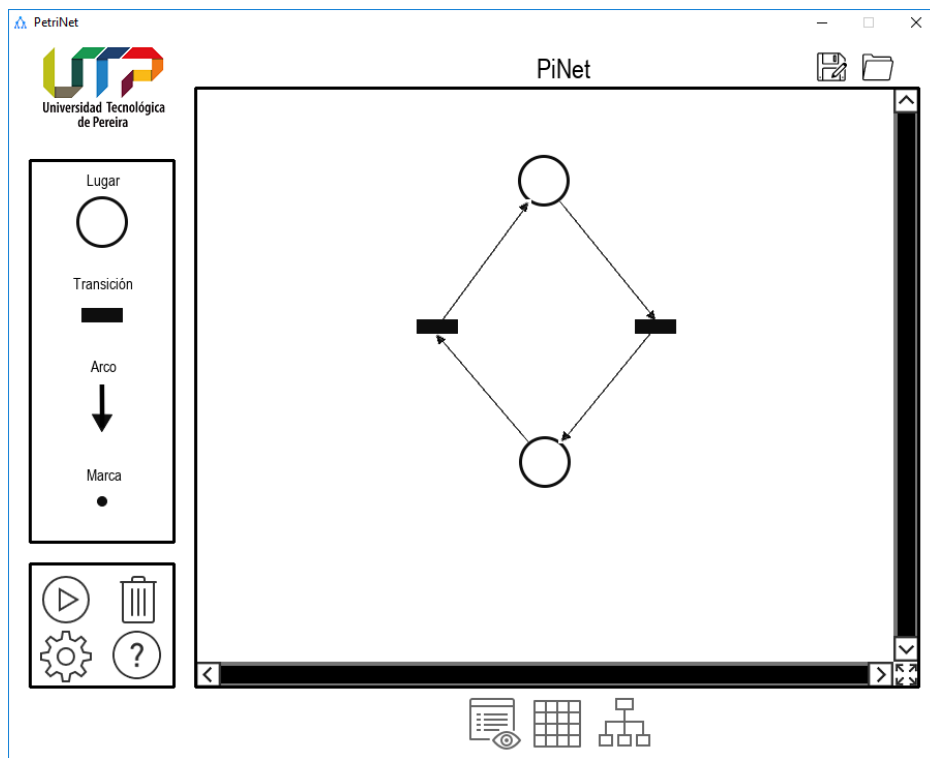


Figura 30: Conexión de elementos.

Ya con la red disponible, basta indicar el marcado de los lugares (**Tokens**) y el peso de los arcos por medio de la herramienta **Marca**. Para el presente ejemplo se realiza la adición de solo una marca al lugar con la etiqueta *Lugar 0*. En la Figura 31 es posible visualizar la adición de dicho token y el cambio de color en *transición 0*, indicando así la habilitación de dicho evento.

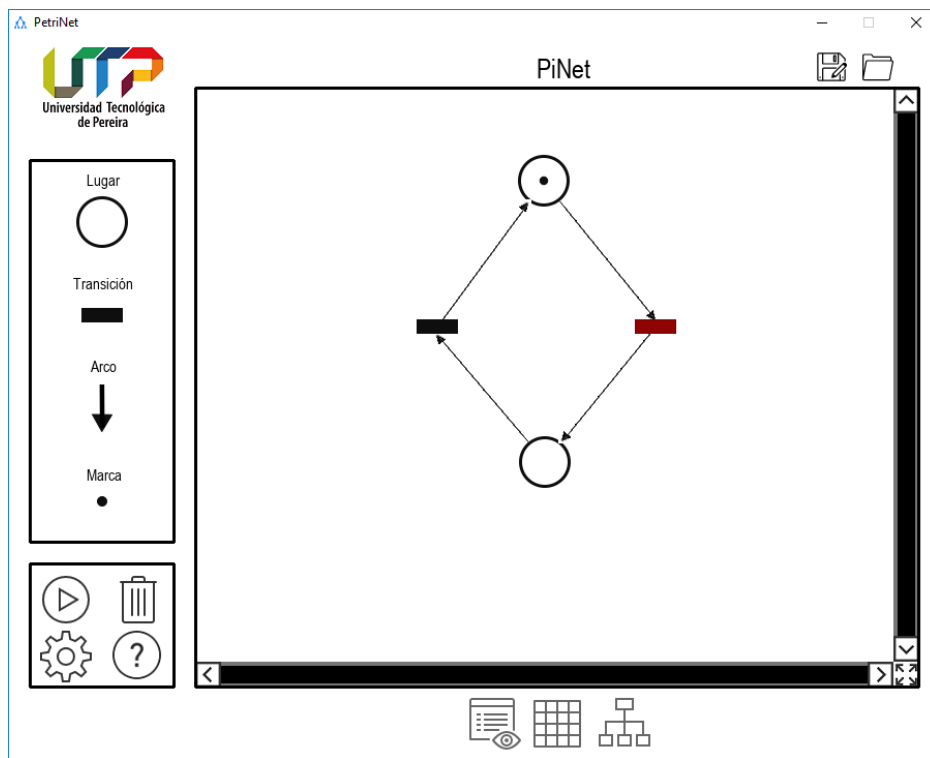


Figura 31: Red de Petri disponible.

Con la transición disponible, se procede a ejecutar una simulación comportamental de la red por medio del botón **Ejecutar**, ubicado en la barra de acciones. Antes de la ejecución de la red, es posible realizar cambios en cuanto al comportamiento que tendrá dicha red, ya sea por medio del modo en que se realizará el disparo (Automático o Manual), el intervalo de simulación que tendrá la operación no temporizada, o el comportamiento que asumirá la temporización en el modo automático. Estos cambios son posibles mediante el botón de configuración. Para el presente ejemplo se eligen los parámetros por defecto, esto es, Modo Automático, Tiempo de ejecución 1 segundo y modo no temporizado. Una vez se confirman los parámetros deseados, se dispara la red, dando lugar al comportamiento que se observa en la Figura 32.

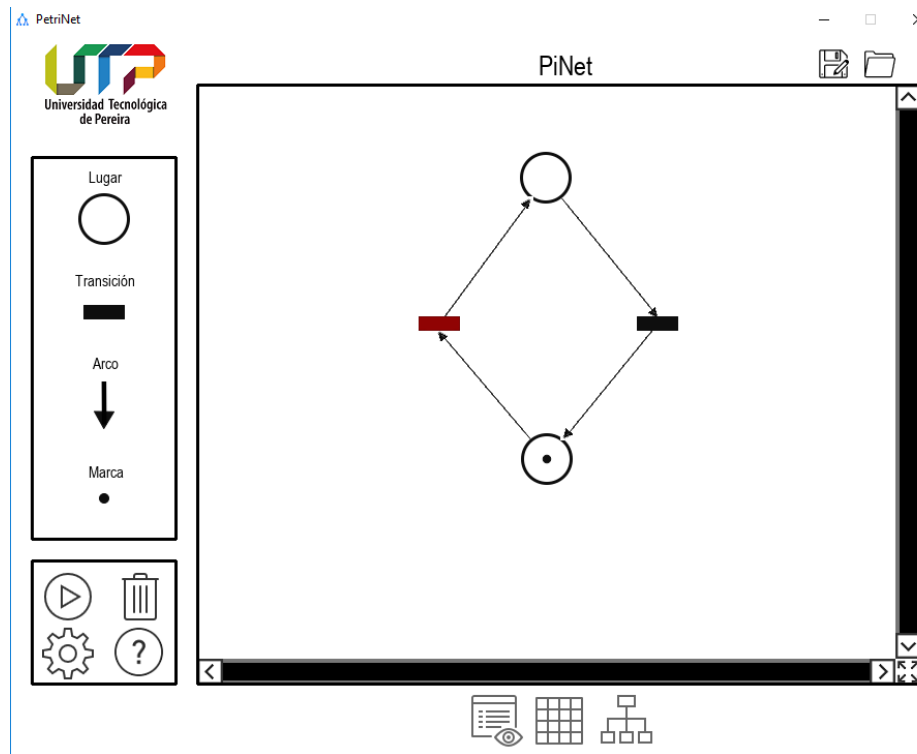


Figura 32: Evolución de una red.

Mediante el panel de propiedades se visualizan distintos aspectos de la Red diseñada, como lo son sus propiedades intrínsecas, sus matrices de incidencia y los vectores anuladores disponibles. Las Figuras 33, 34 y 35 muestran estos aspectos.



Figura 33: Propiedades de la red.

Incidencia <input checked="" type="checkbox"/> Incidencia + <input type="checkbox"/> Incidencia - <input type="checkbox"/> Dual <input type="checkbox"/>		
Lugares	Transiciones	
	T0	T1
P0	-1	1
P1	1	-1

Figura 34: Matrices de la red.

Transformaciones	
Anulador derecho:	
1	1
Anulador izquierdo:	
1	1

Figura 35: Anuladores de la red.

Una vez se verifican cada una de estas propiedades, es posible almacenar el diseño para futuras intervenciones por medio del botón **Guardar**. La Figura 36 enseña la ventana emergente al momento de seleccionar esta opción.

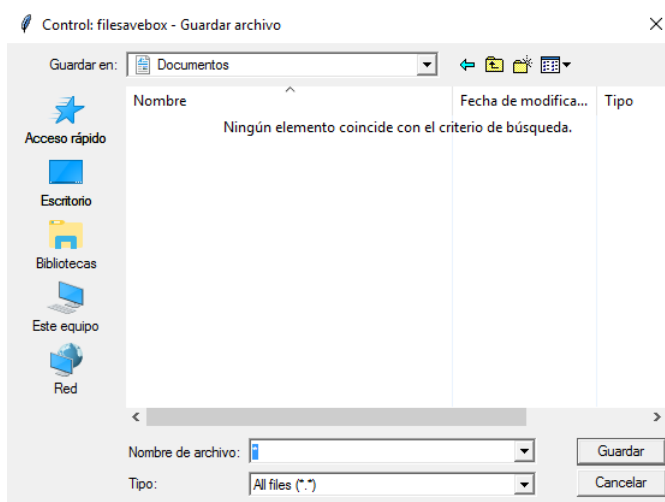


Figura 36: Ventana de diálogo para almacenamiento de una red.

La red a almacenar requiere de un nombre cuyo carácter inicial sea de índole alfabético, luego de este carácter inicial los demás restantes pueden ser una combinación alfanumérica. La Figura 37 enseña una red almacenada con el nombre **red_1**.

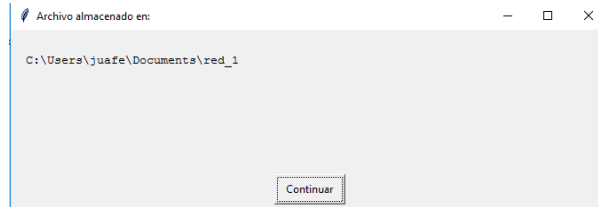


Figura 37: Dirección para la red almacenada dentro del equipo anfitrión.

Ahora se procede a realizar una simulación con la misma red, pero esta vez con una clase de temporización diferente, en este caso una red de Petri T-Temporizada (T-Temp). En la Figura 38 se observa la elección de dicho parámetro.

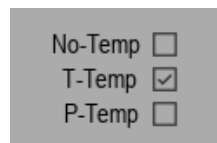


Figura 38: Elección de red T-Temporizada.

Una vez se elige el tipo de temporización que se desea, (T-Temporizada en este caso), se habilita la posibilidad de accionar el botón izquierdo sobre las transiciones ya ubicadas dentro del área de trabajo con la finalidad de definir los valores de tiempo de dichos elementos. En la Figura 39 se observa la propiedad de una transición a la que se le adiciona un tiempo de 1000 ms. Para el ingreso de este tiempo basta con ingresar el número deseado y presionar la tecla **Enter**. El sistema limita al usuario a ingresar solo valores numéricos, en caso de intentar adicionar otro tipo de carácter, éste será omitido. Una vez se adicionan los respectivos tiempos, es posible simular la red bajo la opción **Ejecutar**, como se realizó previamente.

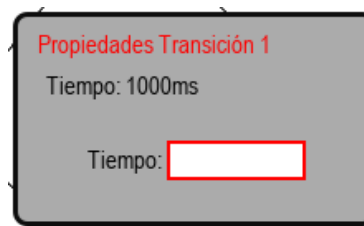


Figura 39: Definición de tiempo a una transición.

4.2. EJEMPLO 2

En el ejemplo siguiente se expone un caso que posee en su estructura una exclusión mutua, característica muy común en sistemas discretos. Una aplicación típica de elementos con exclusión mutua es el modelado de recursos compartidos por múltiples usuarios. La Figura 40 enseña la implementación de dicho caso.

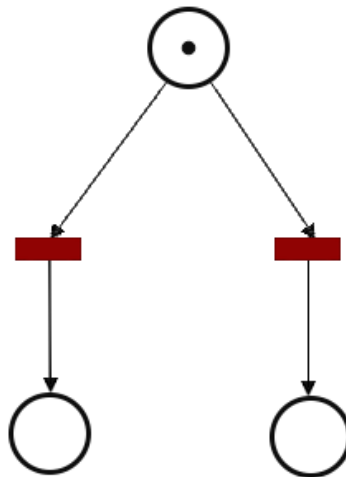


Figura 40: Ejemplo de recurso compartido.

En el modo de simulación Automático, para la elección de entre las transiciones sensibilizadas (habilitada para disparar) se dispone de un algoritmo que determina aleatoriamente

el elemento objetivo bajo una función de distribución uniforme. Como se observa en la Figura 41, para el caso presente, el disparo inicial se realiza por medio de la transición situada a la derecha de la red.

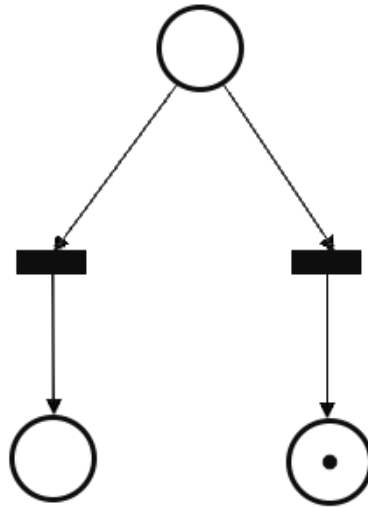


Figura 41: Elección de ruta realizada por el sistema.

Si se adiciona una nueva marca en el lugar inicial (superior), ahora se cuenta con dos transiciones habilitadas, implicando esto una probabilidad de disparo del 50 % para cada una de ellas. Lo descrito se observa en la Figura 42, dando lugar nuevamente a la activación de ambas transiciones.

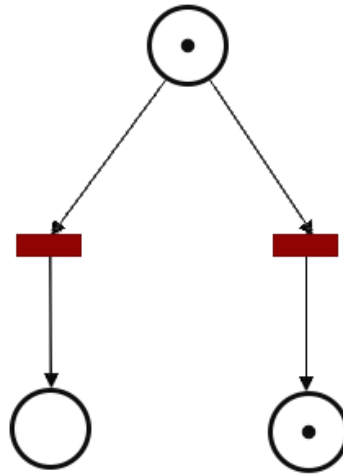


Figura 42: Habilitación de transiciones para un segundo disparo.

Realizando un nuevo disparo se obtiene el resultado que se observa en la Figura 43. Como se aprecia en dicha figura, esta vez el disparo se realiza sobre la transición que se encuentra a la izquierda de la red.

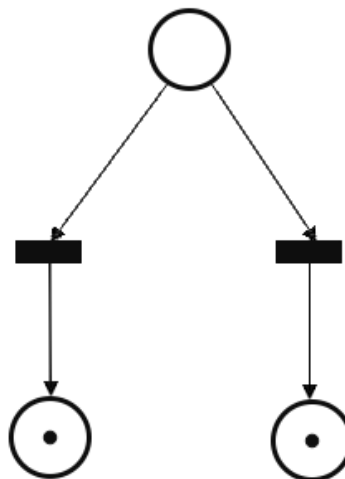


Figura 43: Resultado del segundo disparo de la red.

Si el modo de operación se fija en Manual, es el usuario quien elige autónomamente la transición a disparar (de entre las sensibilizadas) en un instante deseado. La Figura 44

enseña la elección de dicha opción.

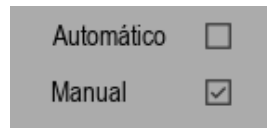


Figura 44: Elección del modo manual.

Si nuevamente se ubica una nueva marca en el lugar de inicio (superior), la red queda como se muestra en la Figura 45.

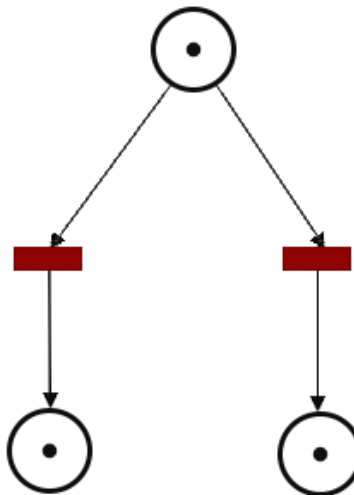


Figura 45: Habilidad de la red para realizar un disparo en modo manual.

Una vez habilitadas las transiciones por medio del token previamente adicionado, y operando bajo el modo manual, se realiza el disparo de la transición que se desee. Para el caso de disparar la transición izquierda se da lugar a la red que se observa en la Figura 46.

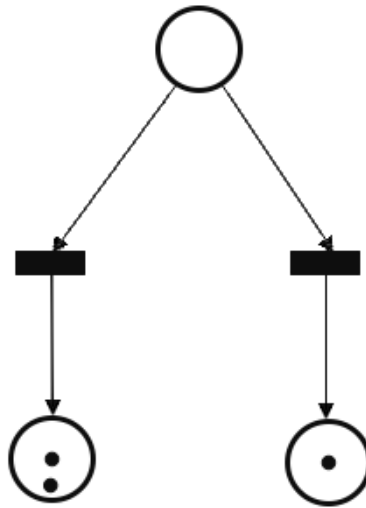


Figura 46: Ejecución de un disparo manual sobre la red.

4.3. EJEMPLO 3

Una de las concepciones más simples de un recurso compartido se presenta cuando una parte de algún sistema debe ser empleada en varios procesos que deben acceder en forma controlada a su uso y notificar su liberación, para permitir su posterior empleo en otro proceso. Otra noción de recurso compartido, normalmente presente en los sistemas industriales, es la necesidad de compartir brazos robóticos o bandas transportadoras en la realización de varias funciones. En la Figura 47 se propone una red en la que se presenta la realización de múltiples tareas, donde P4 posee una marca inicial, notificando que el recurso se encuentra disponible, con lo cual tanto T2 como T3 son transiciones candidatas a hacer uso de este recurso, pasando P4 a ser un recurso ocupado. Para facilidad al momento de reconocer cada uno de los elementos, se dispone de etiquetas fijas sobre la imagen.

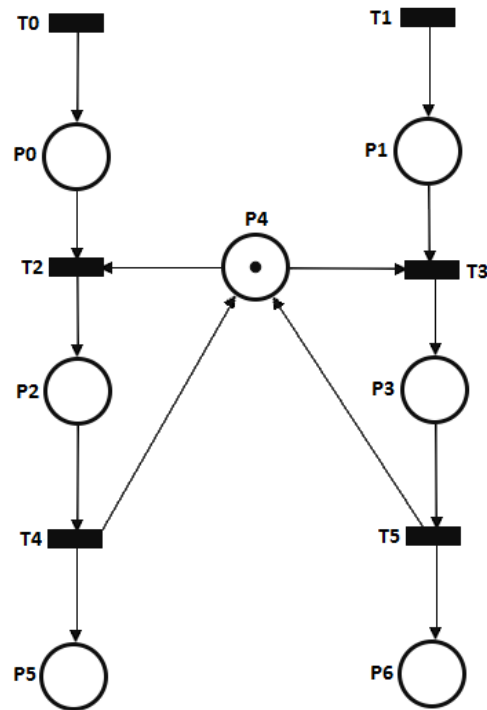


Figura 47: Arquitectura de Recurso Compartido.

Las transiciones T4 y T5 indican la liberación del recurso, regresando así una marca al lugar P4, lo cual crea la notificación de recurso disponible y listo para ser usado nuevamente. Las transiciones T0 y T1 indican la llegada de una nueva tarea o solicitud al sistema desde dos puntos diferentes, actuando como transiciones fuente, es decir, no requieren un lugar de entrada para su disparo. P5 y P6 actúan como estados sumideros, los cuales permiten identificar de manera visual el número de recursos consumidos en cada tarea. Para poseer control sobre el comportamiento de la red y verificar la idoneidad del sistema, se hace simulación bajo el modo de operación Manual, facilitando así la visualización de los disparos realizados dentro del sistema. En la Figura 48 se observa la habilitación de las transiciones T0 y T1, una vez se elige el modo de operación manual y se presiona el botón de **Ejecutar**.

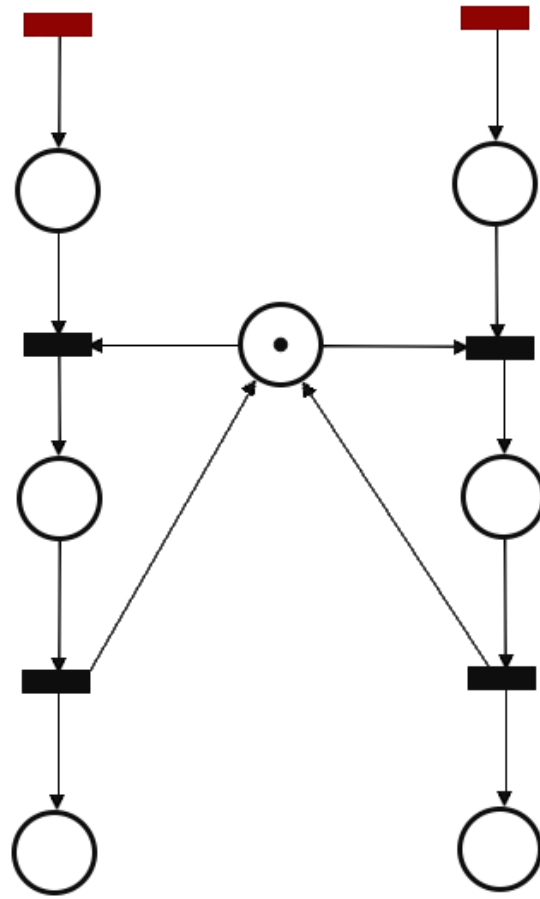


Figura 48: Transiciones para alimentación habilitadas.

Estas dos transiciones, al ser de tipo fuente, se encontrarán disponibles en todo instante de tiempo. Para proceder a dar paso a la evolución del sistema, se realiza la adición de una tarea por medio de la activación de T0. En la Figura 49 se observa el ingreso de una marca al lugar P0.

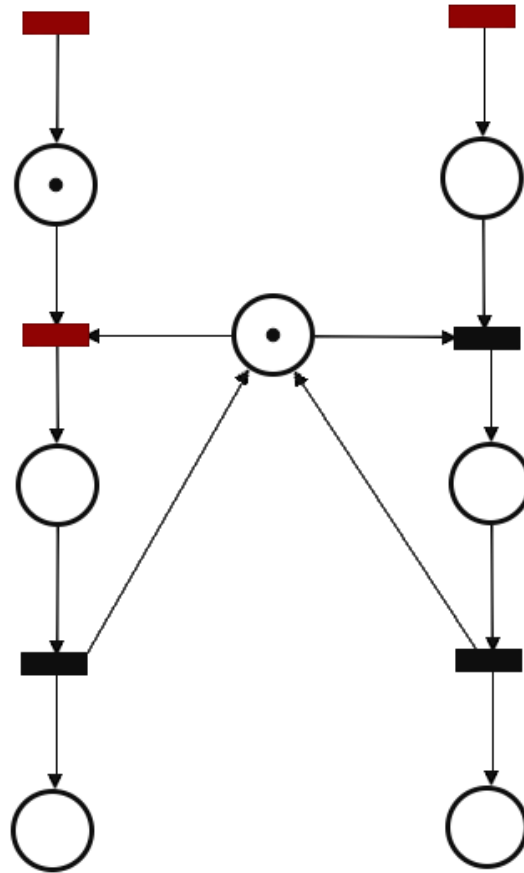


Figura 49: Alimentación de actividad desde T0.

Si luego se activa la transición T1, se pasa una marca al lugar P1 y estará activas las transiciones T2 y T3, ya que el recurso compartido aún se encuentra disponible. En La Figura 50 se observa la disponibilidad de ambas transiciones.

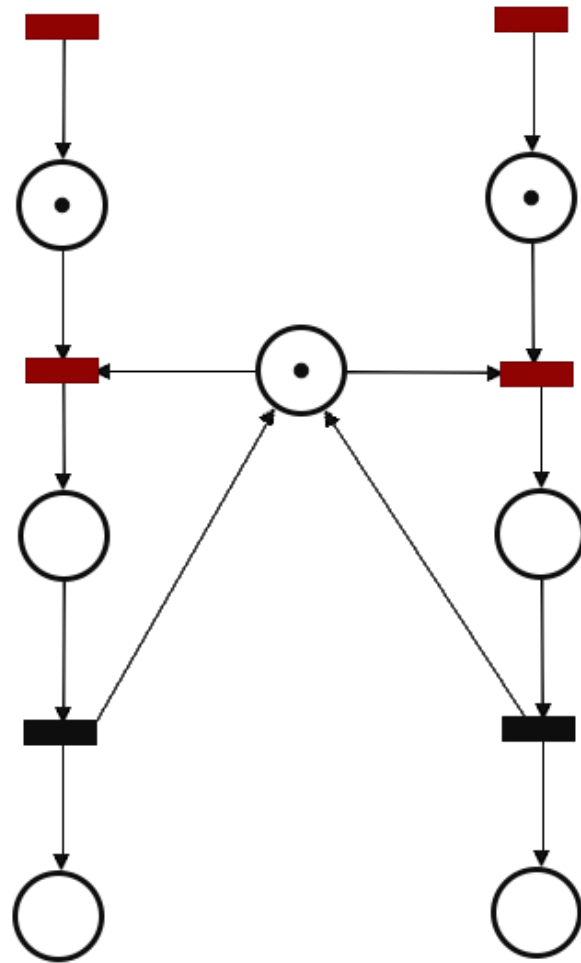


Figura 50: Transiciones 2 y 3 disponibles.

Al instante de disparar ya sea T2 o T3, inmediatamente la otra no estará disponible, puesto que el recurso compartido pasa a un estado de actividad. En La Figura 51 se observa la activación de T2, inhabilitando así a T3.

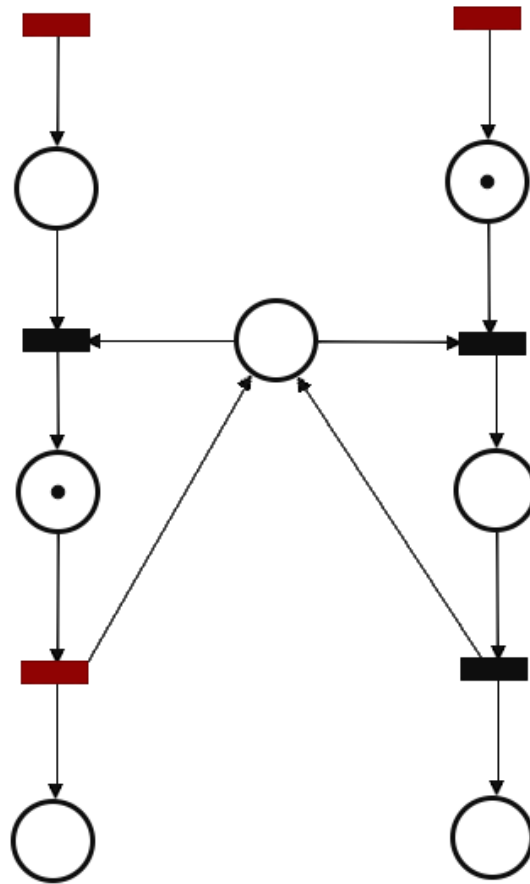


Figura 51: Actividad en el lugar 2.

Para continuar con la evolución de la red, se vuelve obligatorio el disparo de T4, con el fin de indicar que el recurso compartido está nuevamente disponible. En la Figura 52 se observa nuevamente el recurso disponible, además del registro de fin de actividad por medio del lugar P5.

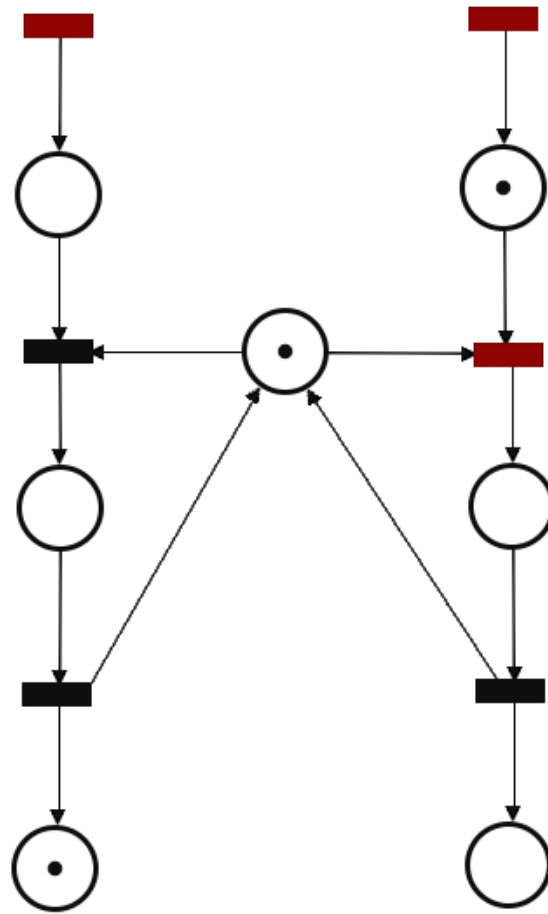


Figura 52: Recurso compartido nuevamente disponible.

4.4. EJEMPLO 4

El presente ejemplo busca evidenciar la validación e integridad de PiNet al mostrar resultados sobre una de las varias redes empleadas para este propósito. El ejemplo se ha tomado de la sección 5.5.7 de la referencia [3]. En éste se expone una red conservativa, la cual cuenta dentro de sus elementos con arcos que poseen pesos mayores a uno, tal como se observa en la Figura 53.

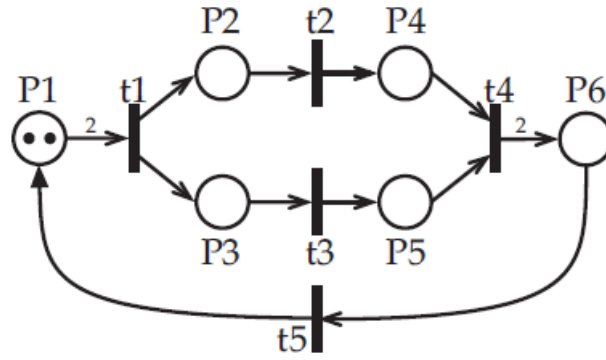


Figura 53: RdP Conservativa.

Como se observa en la estructura de la Red, los arcos que conectan al lugar $P1$ con la transición $T1$ y a la transición $T4$ con el lugar $P6$ poseen un peso igual a dos. La Figura 54 muestra la red desplegada dentro del área de trabajo de PiNet.

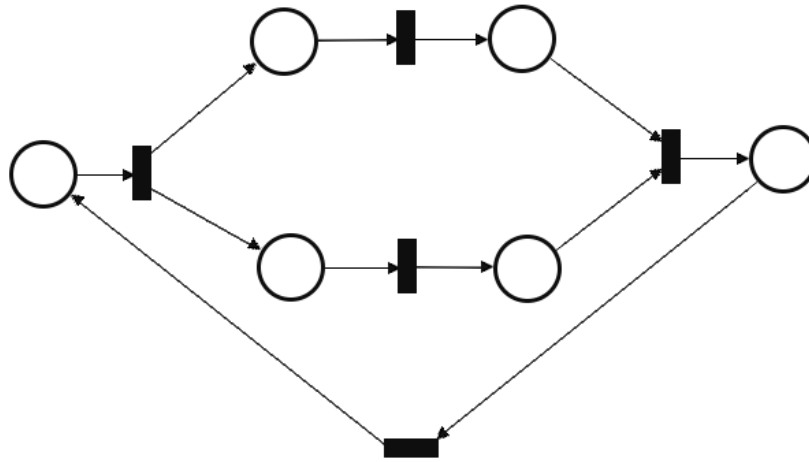


Figura 54: RdP conservativa en PiNet.

Para la red generada en PiNet, los índices de los elementos inician desde el valor cero, a diferencia de la red de la Figura 53 en la cual se inicia a indexar desde el número 1, por tanto, las conexiones previamente descritas con peso igual a dos son dentro de PiNet las que unen el lugar 0 con la transición 0 y la transición 3 con el lugar 5. Para la adición de

estos pesos, basta con adicionar marcas en la cabeza de flecha de la conexión deseada, o bien realizando una nueva conexión entre los elementos deseados que ya poseen una (lo cual suma uno al peso ya existente).

Con los elementos debidamente conectados, se procede a verificar las matrices y anuladores obtenidos dentro de PiNet. En las Figuras 55 y 56 se aprecian las matrices de incidencia posterior (\mathbf{C}^+) y previa (\mathbf{C}^-), tanto de la red base, como de la red dibujada en PiNet, y de las cuales se puede constatar que los resultados coinciden.

$$\mathbf{C}^+ = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \end{bmatrix} \quad \mathbf{C}^- = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de Incidencia Posterior Matriz de Incidencia Previa

Figura 55: Incidencia Previa y Posterior obtenidas en [3].

	T0	T1	T2	T3	T4		T0	T1	T2	T3	T4
P0	0	0	0	0	1	P0	2	0	0	0	0
P1	1	0	0	0	0	P1	0	1	0	0	0
P2	1	0	0	0	0	P2	0	0	1	0	0
P3	0	1	0	0	0	P3	0	0	0	1	0
P4	0	0	1	0	0	P4	0	0	0	1	0
P5	0	0	0	2	0	P5	0	0	0	0	1
Matriz de Incidencia Posterior						Matriz de Incidencia Previa					

Figura 56: Incidencia Previa y Posterior obtenidas en PiNet.

Realizando la operación $\mathbf{C}^+ - \mathbf{C}^-$ se obtiene la matriz de incidencia. Las Figuras 57 y 58 permiten confirmar nuevamente la integridad de los resultados arrojados por PiNet.

$$\begin{bmatrix} -2 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 2 & -1 \end{bmatrix}$$

Figura 57: Matriz de incidencia obtenida en [3].

Incidencia ☒ Incidencia + ☐ Incidencia - ☐ Dual ☐

Lugares \ Transiciones					
	T0	T1	T2	T3	T4
P0	-2	0	0	0	1
P1	1	-1	0	0	0
P2	1	0	-1	0	0
P3	0	1	0	-1	0
P4	0	0	1	-1	0
P5	0	0	0	2	-1

Figura 58: Matriz de incidencia obtenida en PiNet.

En las secciones 5.11.3.5 y 5.11.3.6 de [3] se enseñan los anuladores de red obtenidos para la RdP conservativa de la Figura 53, los cuales son $\gamma = [1, 1, 1, 1, 2]^T$ (Anulador derecho) y $\Delta = [1, 1, 1, 1, 1, 1]^T$ (Anulador izquierdo). La Figura 59 enseña los anuladores obtenidos en PiNet, que vuelven a coincidir con los datos de referencia.

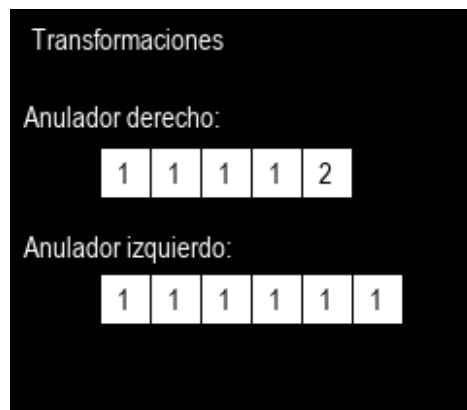


Figura 59: Vectores anuladores obtenidos en PiNet.

Ahora se procede a seleccionar una simulación en modo P-Temporizada (P-Temp), tal como se observa en la Figura 60.

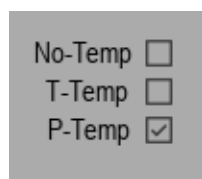


Figura 60: Elección modo P-Temporizado.

Ya con el modo de temporización P-Temp elegido, se realiza la adición de una marca al lugar con la etiqueta cero (en PiNet), tal como se muestra en la Figura 61. En este ejemplo, y desde este punto, siempre se hará referencia a lugares y transiciones según la numeración que realiza PiNet.

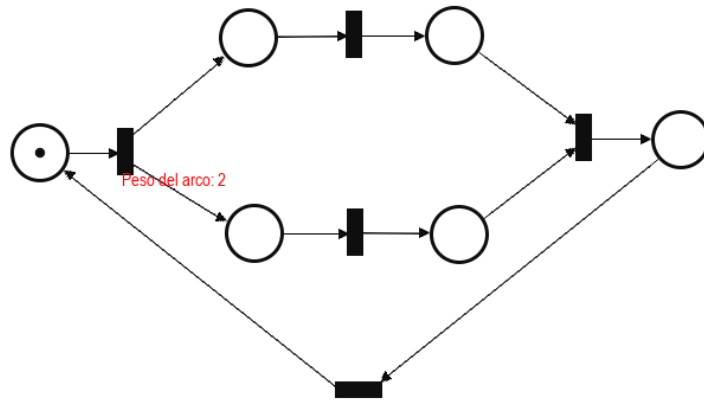


Figura 61: Transición no sensibilizada.

Tal como se aprecia en la figura anterior, la transición 0 no se encuentra sensibilizada debido a que el peso del arco de entrada es dos, por tanto se hace necesaria la adición de una nueva marca al lugar 0. En la Figura 62 se observa la sensibilización de la transición 0 al añadir una nueva marca al lugar 0.

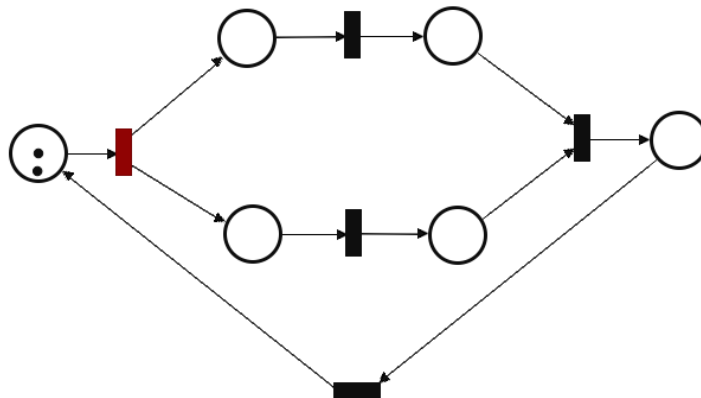


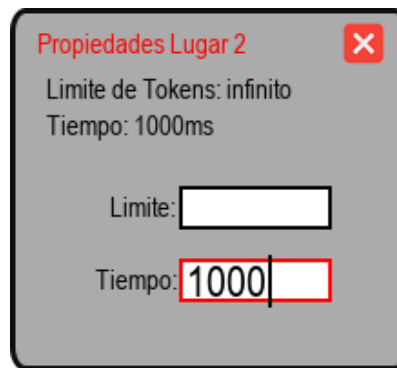
Figura 62: Red sensibilizada.

Ahora se añaden tiempos a los lugares deseados, en caso de no realizar dicha acción sobre algún lugar, se asignan los valores de tiempo por defecto, el cual es igual a cero milisegundos.

En este ejemplo, los tiempos asignados a cada lugar son los siguientes:

- Lugar 0 = 0ms
- Lugar 1 = 2500ms
- Lugar 2 = 1000ms
- Lugar 3 = 1500ms
- Lugar 4 = 0ms
- Lugar 5 = 3000ms

En la Figura 63 se enseña un ejemplo de adición de tiempo, en este caso para el lugar 2.



The image shows a dialog box with a grey background and a red close button in the top right corner. The title bar reads 'Propiedades Lugar 2'. Inside the dialog, the text 'Limite de Tokens: infinito' and 'Tiempo: 1000ms' is displayed. Below this, there are two input fields. The first is labeled 'Limite:' and is empty. The second is labeled 'Tiempo:' and contains the value '1000'. The 'Tiempo:' label and the '1000' value are highlighted with a red rectangular border.

Figura 63: Adición de tiempo al lugar 2.

Una vez se fijan los tiempos sugeridos, se procede a ejecutar la evolución de la red por medio del botón **Ejecutar**, para lo cual se obtiene que el orden de disparo en las transiciones y de esperas en los lugares produce un flujo de marcado por los lugares en el orden $0 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 3 \rightarrow 5 \rightarrow 0$. En las Figuras 64, 65, 66, 67, 68 y 69 es posible apreciar la evolución de la red luego de las esperas fijadas para cada lugar.

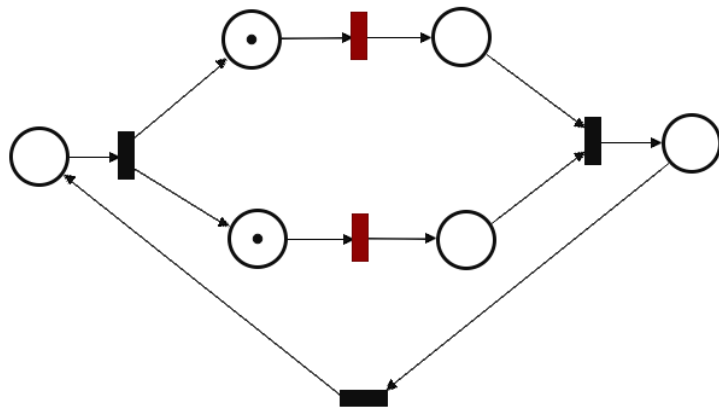


Figura 64: Disparo de T0.

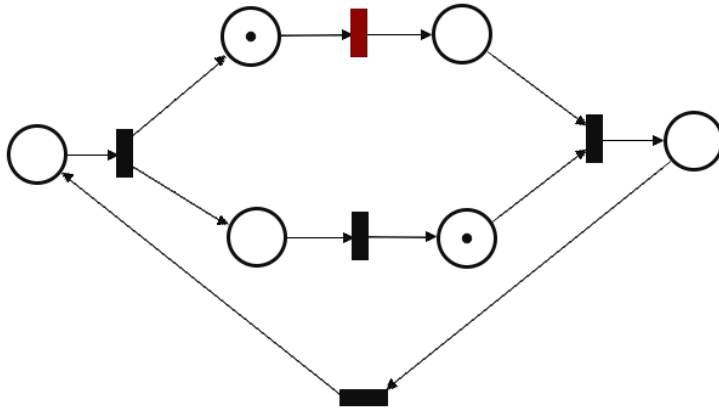


Figura 65: Disparo de T2.

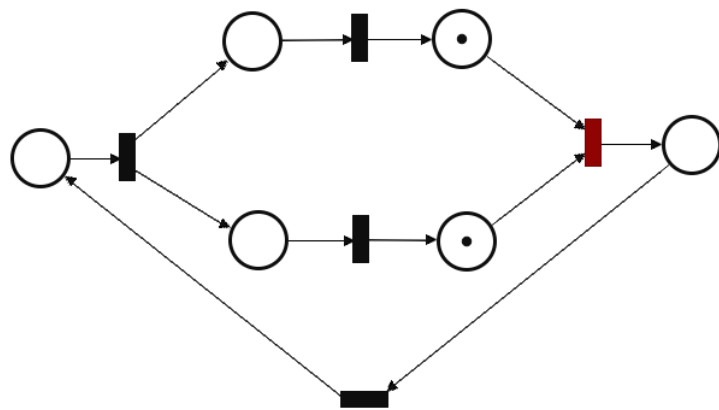


Figura 66: Disparo de T1.

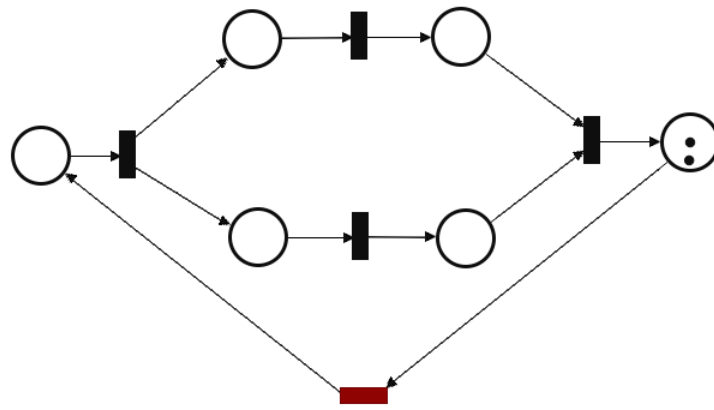


Figura 67: Disparo de T3.

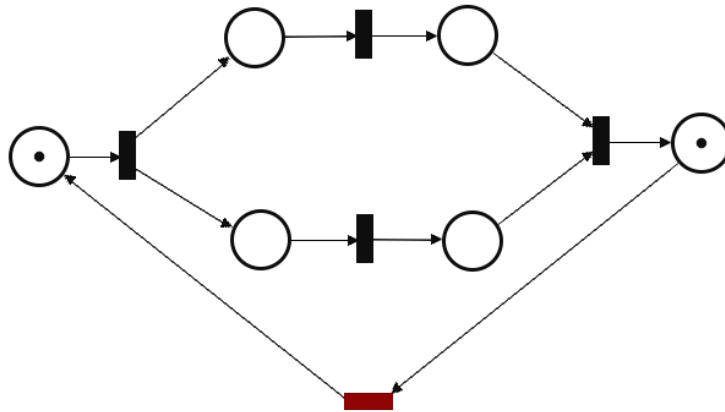


Figura 68: Disparo de T4.

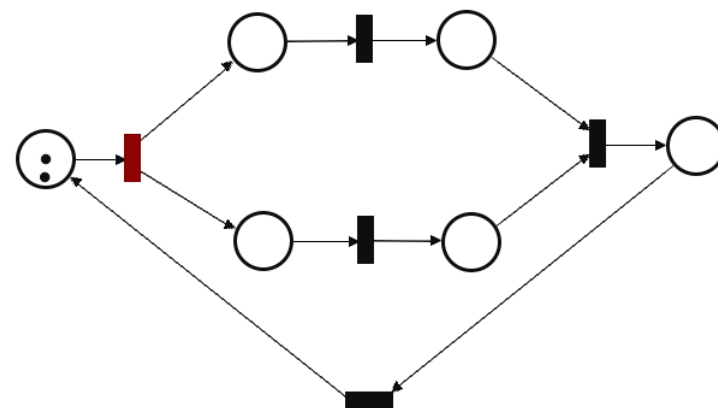


Figura 69: Segundo disparo de T4.

4.5. CARPETA DE EJEMPLOS

Para facilidad del usuario, se disponen de múltiples redes previamente realizadas y almacenadas dentro de la carpeta **Ejemplos**, con el fin de permitir interactuar con redes que cubren diferentes aplicaciones y siendo fundamentales para la correcta comprensión del uso del software. Para cargar dichas redes se procede a seleccionar el botón **Cargar**, opción que despliega una ventana de diálogo desde donde se selecciona el archivo deseado.

5. MATERIALES Y MÉTODOS EMPLEADOS

En esta sección se hace una breve descripción de los principales métodos (marco teórico) y materiales (elementos de soporte) que fundamentan teóricamente la implementación de **PiNet**.

5.1. CARACTERÍSTICAS DE PiNet

PiNet es un software multiplataforma desarrollado en el lenguaje de programación Python, por medio de la librería Pygame, la cual permite el desarrollo de software que se basa en módulos de videojuegos en 2D, siendo ideal para la creación de interfaces que requieren de una mayor interacción entre el usuario y el sistema. Dentro de este módulo se encuentran características de especial interés, como lo son los eventos por ratón y teclado, los cuales son piezas esenciales en el desarrollo del entorno gráfico interactivo de **PiNet**.

PiNet es una herramienta para el diseño, simulación y análisis de sistemas de eventos discretos por medio de Redes de Petri. Se permite una simulación del sistema tanto de forma manual como automática, siendo esta última opción posible por medio de tres métodos: sin temporización, con transiciones temporizadas o con lugares temporizados. Cada una de las redes representadas puede ser almacenada dentro del equipo anfitrión, esto con la finalidad de permitir futuras evaluaciones y correcciones a un sistema deseado. Dentro de las características que se incluyen en el presente software se encuentran las siguientes:

- Visualización de los distintos objetos al momento de buscar su posicionamiento.

- Posibilidad de situar las cabezas de flecha de los arcos dentro de la porción de área de los distintos objetos (lugares, transiciones) deseada.
- Rotación de transiciones para permitir mayor orden en las conexiones.
- Adición y sustracción de marcas y/o pesos por distintos medios.
- Ampliación del área de trabajo por medio de “Scrolling” o desplazamiento del área.
- Señalización de transiciones sensibilizadas (susceptibles de disparo).
- Señalización de los objetos que se desean conectar por medio de un arco.
- Operación del sistema, tanto en modo automático como en modo manual.
- Inclusión de redes No-Temporizadas, P-Temporizadas y T-Temporizadas dentro del modo automático.
- Obtención de parámetros gráficos y analíticos, tales como matrices de incidencia, anuladores de matrices y propiedades de las redes.
- Inclusión de hasta aproximadamente 150 elementos dentro del área de trabajo.

A continuación, el Cuadro 1 enseña un comparativo entre distintas plataformas de software para simular redes de Petri:

5.2. INTRODUCCIÓN A LAS REDES DE PETRI

Las Redes de Petri fueron introducidas inicialmente por el Dr. Carl Adam Petri en el año de 1962 para su disertación doctoral en la facultad de Matemáticas y Física del

Atributo	Pipe	PetriTool	JARP	Analizador Petri [5]	PiNet
Componente visual	x	x	x	x	x
Simulación manual	x	x	x	x	x
Simulación automática	x	x			x
Verificación de propiedades	x	x	x	x	x
Resultados gráficos	x			x	x
Simulación automática	x	x	x	x	x

Cuadro 1: Cuadro comparativo de plataformas

Technical University of Darmstadt, en Alemania Occidental. El éxito de las Redes de Petri radica en la amplitud de diferentes sistemas que se pueden modelar bajo esta misma técnica, entre los cuales se pueden incluir: sistemas asíncronos, concurrentes, paralelos, no determinísticos, secuenciales, de eventos discretos, distribuidos, estocásticos, entre otros [3].

Las RdP constan de tres componentes denominados como Lugares (elementos pasivos), Transiciones (elementos activos) y Arcos (elementos conectivos), donde los lugares están relacionados con estados, condiciones, recursos, esperas, etc. y en conjunto reciben la denominación P y se representan por círculos; las transiciones están relacionadas con eventos, acciones, ejecución de sentencias, etc. y en conjunto reciben la denominación T y se representan por rectángulos o segmentos de línea; y los arcos que unen lugares con transiciones y transiciones con lugares, más no dos lugares o dos transiciones entre sí, y se representan por segmentos orientados de línea a los cuales se les asocia de forma individual un peso (k) que es un valor entero positivo y el cual se puede interpretar como un conjunto de k arcos en paralelo. El conjunto de todos los arcos que unen lugares con transiciones y transiciones con lugares se designa por F y a la función de peso asignada a cada arco se le denomina W . La representación matemática de las Redes de Petri se

basa en un modelo matricial-vectorial que además de describirlas permite su estudio, análisis y abstracción de sistemas complejos.

Formalmente, una Red de Petri se define como una quintupla:

$$PN = \{P, T, F, W, M_0\}$$

Donde:

- $P = \{p_1, p_2, \dots, p_m\}$ es el conjunto finito de lugares.
- $T = \{t_1, t_2, \dots, t_m\}$ es el conjunto finito de transiciones.
- $F \subseteq (P \times T) \cup (T \times P)$ es el conjunto de arcos que definen el flujo de la red.
- $W : F \rightarrow \{1, 2, 3, \dots\}$ es la función de peso.
- $M_0 : P \rightarrow \{0, 1, 2, 3, \dots\}$ es el marcado inicial de la red.

En toda red de Petri la función lógica que expresa la condición para que una transición esté sensibilizada (pre condición) se debe al cumplimiento de tener cada uno de los lugares previos (de entrada) a la transición con mínimo número de marcas como peso tiene el arco que une cada lugar con la transición. Así, si $\{p_1, p_2, p_3\}$ es el conjunto de lugares de entrada de una transición, se dispara primero la transición que estado sensibilizada tenga su condición de disparo en uno lógico. Al disparar una transición, la evolución del marcado es tal que, de cada lugar de entrada a la transición se retira el número de marcas indicadas por el peso del arco que une lugar con transición, y se adicionan marcas en cada lugar de salida de la transición como peso tiene el arco que une la transición con el lugar respectivo.

Para una simulación no temporizadas, se asume que todas las transiciones tienen igual prioridad y por tanto, siempre se activa la primera transición para la cual el sistema evalúe en uno lógico su condición de disparo, bajo la premisa de estar sensibilizada. Para una simulación T-temporizada, fuera de la condición de estar sensibilizada, se hace una espera indicada por la temporización antes de evaluar su respectivo disparo. La simulación P-temporizada es similar a la T-temporizada, con la diferencia que la espera se realiza en los lugares y no en las transiciones. Con fines prácticos, para una simulación No temporizada se puede realizar una temporización con valores iguales en todas las transiciones, con el único fin de permitir al usuario ver una evolución en tiempo lento.

5.2.1. Matrices de Incidencia

En las redes de Petri, la representación matricial se puede obtener mediante dos matrices, la de incidencia previa (\mathbf{C}^-) que representa las conexiones existentes entre lugares de entrada a transiciones y la de incidencia posterior (\mathbf{C}^+) que presenta las conexiones existentes entre transiciones y sus lugares de salida. Cuando una red de Petri no posee arcos de entrada y salida a la vez entre un lugar y una transición, se dice que es pura, y su representación se puede simplificar en una única matriz denominada Matriz de Incidencia, \mathbf{C} , la cual se define como: $\mathbf{C} = \mathbf{C}^+ - \mathbf{C}^-$. Cada uno de los elementos que compone la matriz de incidencia, c_{ij} , es positivo para indicar la presencia de incidencia posterior, negativo para indicar la presencia de incidencia previa, o cero para indicar la no conexión entre el lugar p_i y la transición t_j .

5.2.2. Vector de marcado

En una red de Petri interesa conocer el marcado que se alcanza luego de realizar una cierta secuencia de disparo partiendo de un marcado inicial conocido. El vector de disparo, μ_k , es un vector con tantos elementos como transiciones tiene la red y con un valor de 1 en la posición de la transición que se dispara y cero en el resto de componentes. En las redes de Petri puras y con un marcado inicial conocido, existe una ecuación que determina el marcado alcanzado desde un marcado inicial dado un vector de disparo y se conoce como ecuación de estado, la cual está dada por:

$$M_k^T = M_{k-1}^T + C\mu_k$$

Como esta ecuación entrega una forma de determinar un marcado posterior a partir de uno previo, entonces se puede decir sucesivamente que:

$$M_k^T = M_{k-2}^T + C\mu_{k-1} + C\mu_k = M_0^T + C(\mu_1 + \dots + \mu_k)$$

De donde se define el vector secuencia de disparo $\sigma = \mu_1 + \dots + \mu_k$, el cual al ser reemplazado en la ecuación anterior se obtiene finalmente:

$$M_k^T = M_0^T + C\sigma$$

De la anterior ecuación, es importante resaltar el hecho que el vector σ sólo entrega datos sobre las transiciones disparadas para llegar hasta el marcado deseado, más no brinda información sobre la secuencia en la cual se realizaron dichos disparos.

5.2.3. Propiedades de las redes

Para determinar las propiedades de una red de Petri mediante el uso del análisis por representación estructural se debe presentar su aplicación individual a cada una de ellas, como lo son los conceptos de Reversibilidad, Vivacidad, Pureza y Limitación, para lo cual se realiza a continuación la descripción y método de obtención de cada una de éstas.

Una RdP $PN = \{N, M_0\}$ es reversible si para todo marcado alcanzable M_k , es posible alcanzar nuevamente el marcado inicial M_0 . Para determinar la existencia de la reversibilidad deberá existir un vector anulador derecho, γ , con todos sus elementos positivos para la matriz de incidencia de la red.

Si para una red de Petri, $PN = \{N, M_0\}$, existe el anulador izquierdo, Δ , tal que $\Delta^T C = 0$, la red es conservativa y limitada.

La determinación de la vivacidad de una red de Petri no es directa, al no existir una propiedad general que la pueda determinar, ya que como se ha expuesto, el vector secuencia de disparo no entrega la información completa sobre el orden de disparo de las transiciones. La propiedad de vivacidad establece que el sistema debe alcanzar todos los estados para los que fue diseñado, por tal motivo, esta propiedad sirve para caracterizar la ausencia de bloqueos mutuos.

5.2.4. Invariantes de marcado y de disparo

Se denomina invariante de disparo a cualquier relación satisfecha por todas las secuencias de disparo que se pueden realizar desde el marcado inicial. De igual forma, se denomina Invariante de marcado a cualquier relación satisfecha por todos los marcados alcanzables

desde el marcado inicial. La obtención de estos invariantes se basa en el vector anulador derecho, γ , y el vector anulador izquierdo, Δ .

5.3. MODELOS UML

Para la correcta visualización e interpretación del software, se hace uso de las herramientas del lenguaje de modelado “UML”, el cual es un estándar aprobado por ISO como un modelo general para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. Estos se dividen en las siguientes tres categorías: clasificación estructural, comportamiento dinámico, y la gestión del modelo. Dentro de cada una se exponen modelos distintos, esto con la finalidad de captar y enumerar exhaustivamente los requisitos y el dominio de conocimiento, de forma que todos los implicados puedan entenderlos y estar de acuerdo con ellos. Los diversos modelos de un sistema de software pueden capturar requisitos sobre su dominio de aplicación, las formas en que los usuarios harán uso de éste, su división en módulos, los patrones comunes usados en un sistema, entre otras [6]. A continuación, se presentan los modelos implementados dentro de cada una de las clasificaciones previamente mencionadas.

Modelos Estructurales: Dentro de esta clasificación se describe los elementos del sistema y sus relaciones con otros elementos. Para la presente clasificación se enseñan el diagrama de clases, el diagrama de componentes y el diagrama de paquetes para **PiNet**.

- Diagrama de clases: Permite visualizar la relación existente entre las clases que involucran el sistema, las cuales pueden ser de naturaleza asociativa, de herencia, de uso y de contenido. Dentro del presente modelo, ver Figura 70, se expone la inter-

acción existente entre el encapsulamiento de cada uno de los objetos involucrados en el desarrollo de la interfaz.

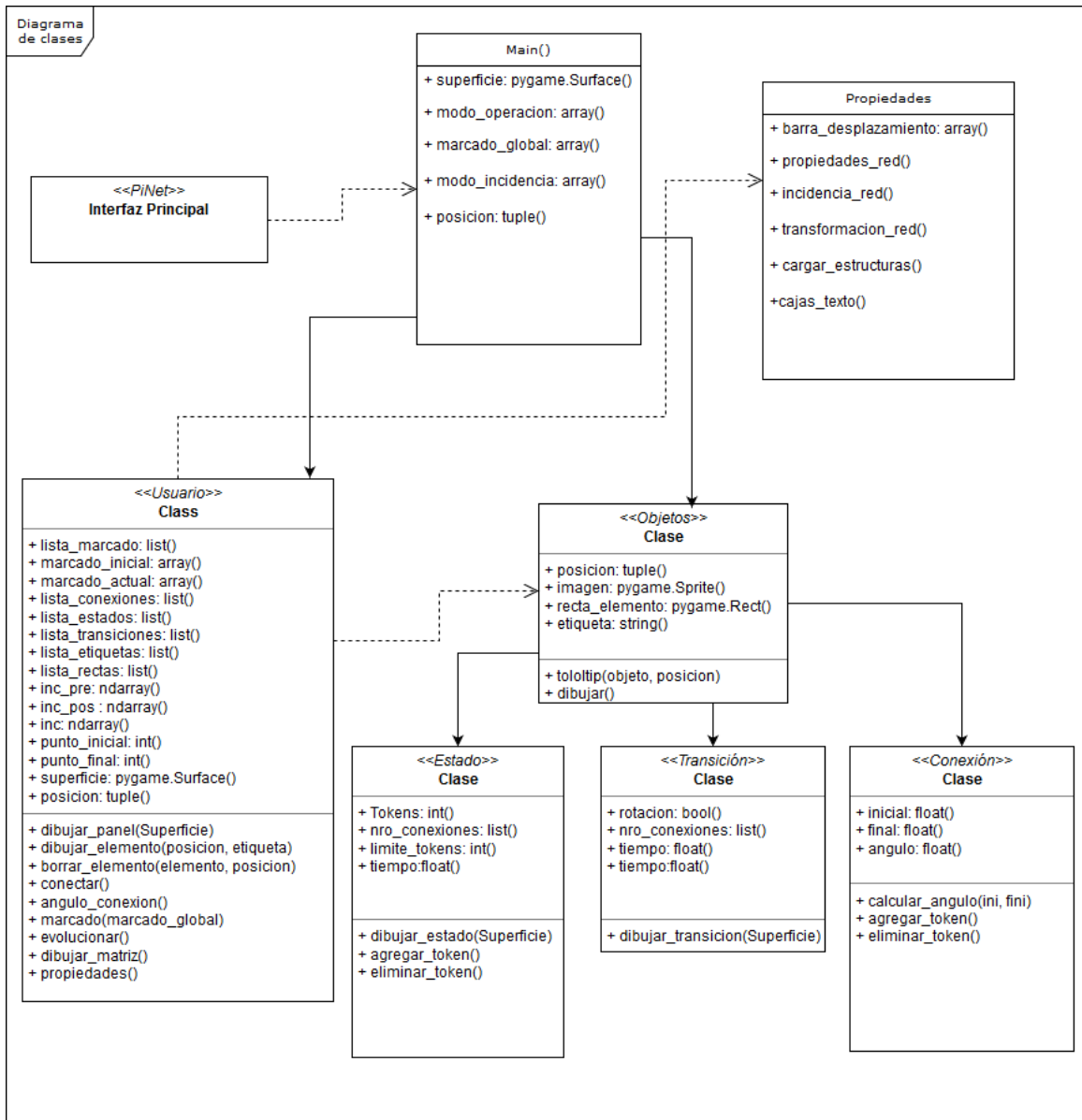


Figura 70: Diagrama de clases.

- Diagrama de componentes: Permite visualizar la estructura de alto nivel del sistema y el comportamiento del servicio que estos componentes proporcionan y usan a través de la interfaz. Dentro de este modelo se observa al usuario como principal actor dentro de los módulos de la interfaz, siendo este quien tiene acceso a cada

uno de los demás módulos restantes, ver Figura 71.

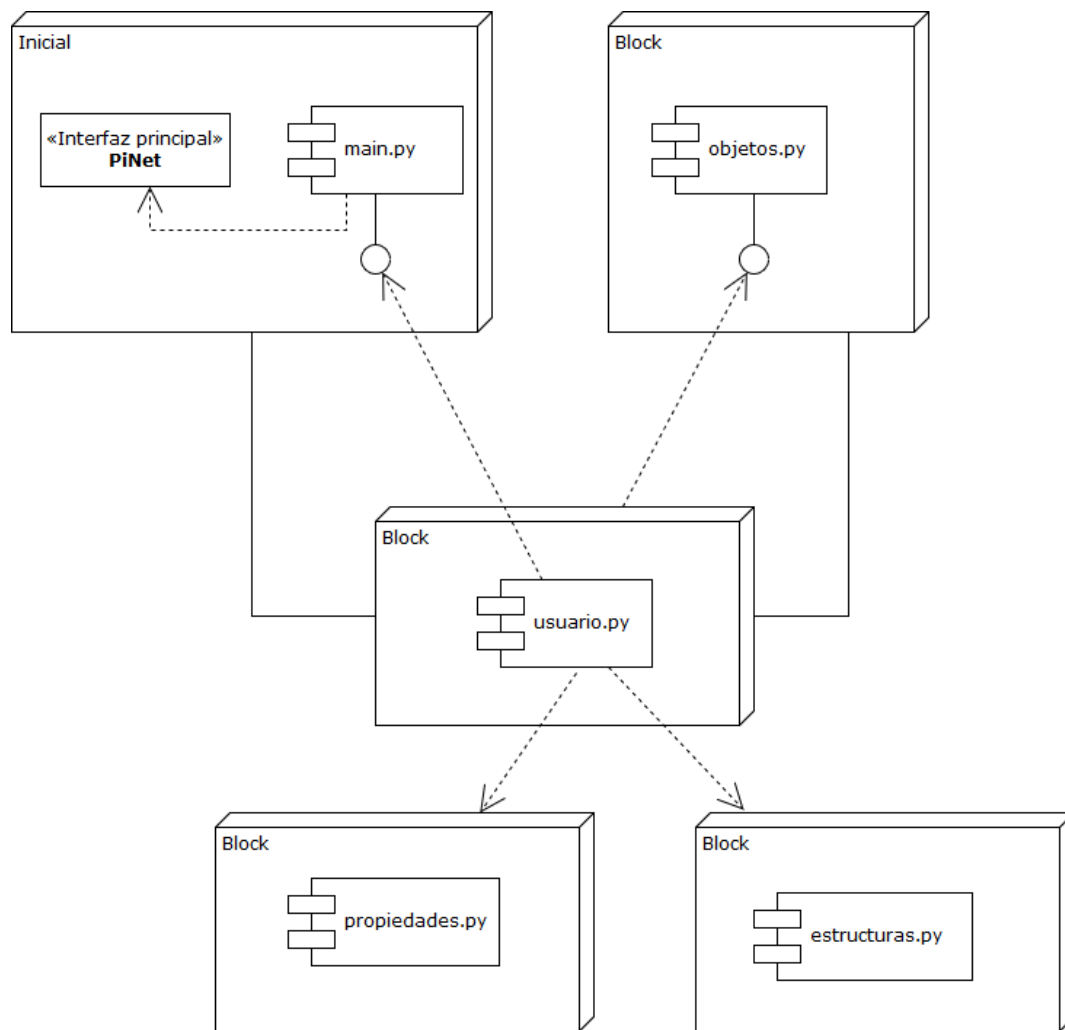


Figura 71: Diagrama de componentes.

- Diagrama de paquetes: Permite dividir el modelo para agrupar y encapsular sus elementos en unidades lógicas individuales. Por medio del presente diagrama se visualiza la relación existente entre cada uno de los módulos de la interfaz, incluyendo el módulo encargado de la carga de los eventos y renderización de gráficos, tal como se aprecia en la Figura 72.

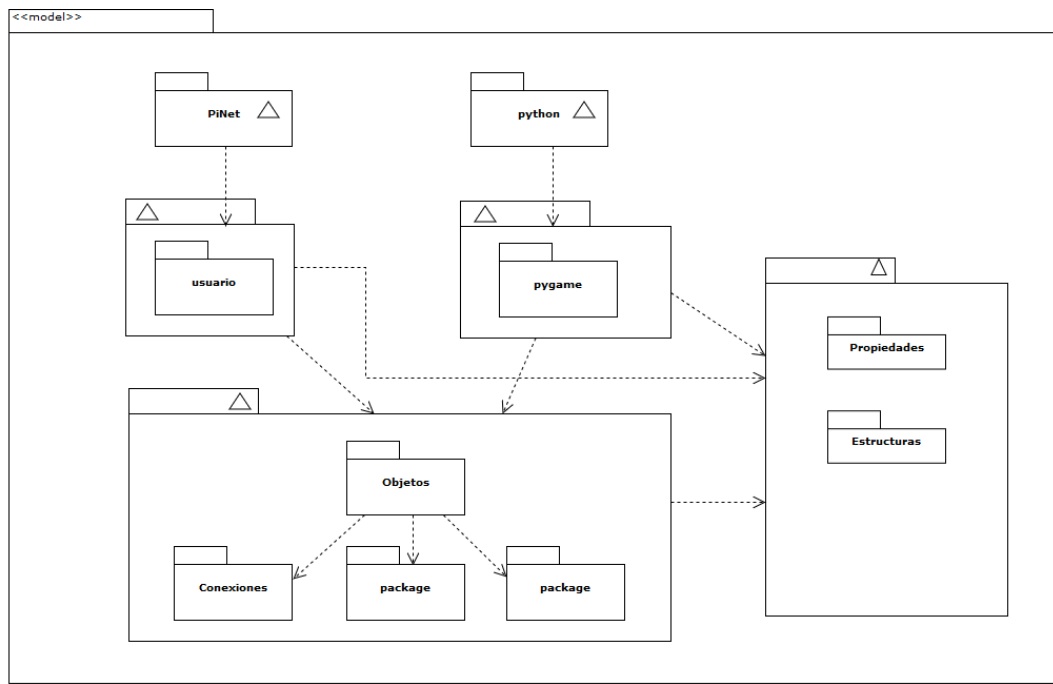


Figura 72: Diagrama de paquetes.

Modelo de Comportamiento: Enseñan el comportamiento dinámico de los objetos en el sistema. Se presenta por medio del diagrama de actividades.

- Diagrama de actividades: Permite describir las actividades que realiza cada módulo del programa, así como la relación que tiene con los distintos módulos presentes. Dentro de este se observa la interacción entre las distintas acciones presentes en la interfaz, aclarando cuál es el inicio del programa y cuál puede ser considerado como el punto final de éste, ver Figura 73.

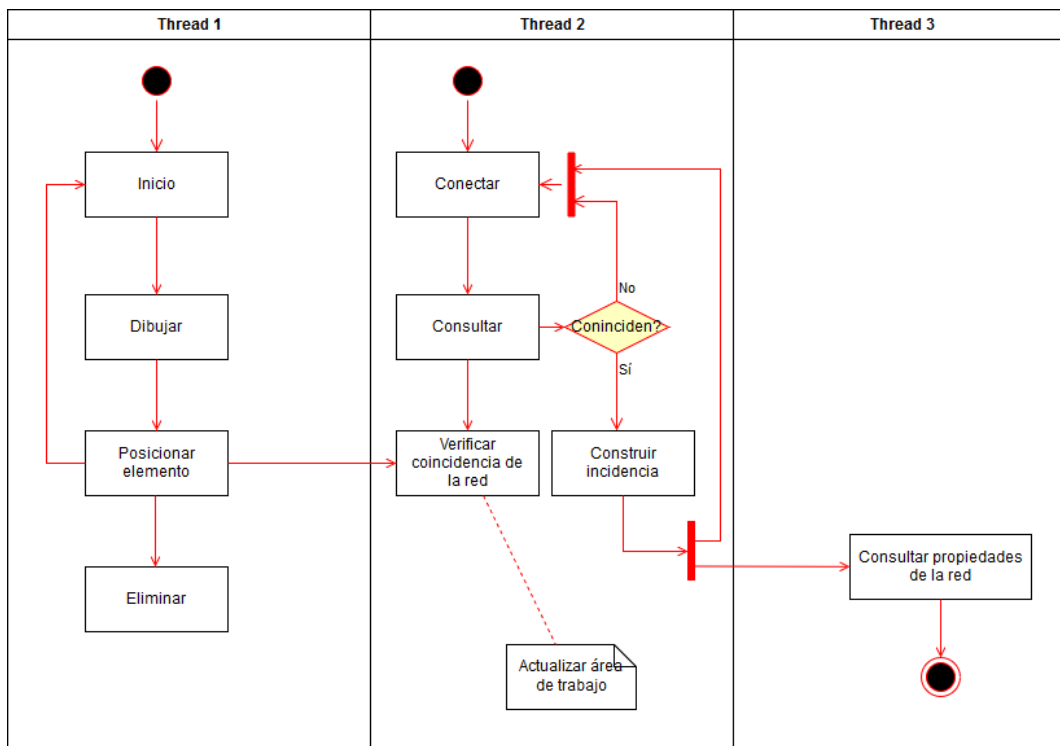


Figura 73: Diagrama de actividades.

Gestión del modelo: Describe la organización de los propios modelos en unidades jerárquicas. La implementación presente es a través del diagrama de secuencias.

- Diagrama de secuencias: Permite determinar de manera gráfica el orden en que se ejecuta cada acción dentro del programa. En este se visualiza la jerarquía de las acciones, teniendo cada una de estas que respetar un orden, para cumplir con la correcta ejecución del programa, tal como se aprecia en la Figura 74.

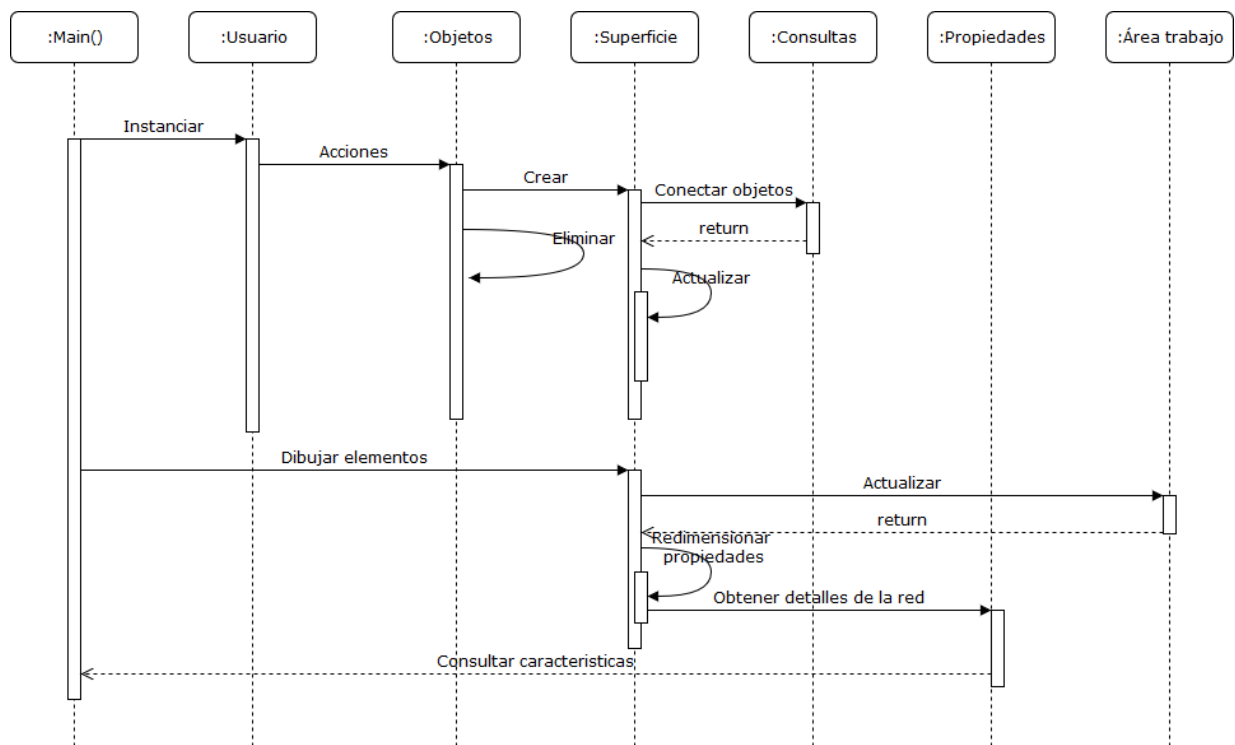


Figura 74: Diagrama de secuencias.

6. EVALUACIÓN DE LA CALIDAD DEL SOFTWARE

6.1. INTRODUCCIÓN

Como parte del proceso de evaluación por pares productos de Software al interior de la Universidad Tecnológica de Pereira, este documento presenta los aspectos preponderantes durante el proceso de planeación, diseño, implementación, verificación y validación del presente producto de Software, denominado PiNet.

A nivel mundial, existe un conjunto de estándares como la normas ISO/IEC 9126, NTP/ISO 17799, ISO 27001, BS 25999 y la ISO/IEC 19505, las cuales permiten dar

guías y buenas prácticas para la evaluación de la calidad de software, su seguridad y su integridad, siendo todas ellas ampliamente difundidas, aceptadas y consideradas como referentes internacionales.

El estándar ISO/IEC 9126 presenta un modelo para la evaluación de la calidad del software mediante el siguiente conjunto de características: Funcionalidad, Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad, Portabilidad y Calidad en Uso. A su vez, cada una de estas características se define y divide en un conjunto de sub-características, de la siguiente manera:

- Funcionalidad - atributos que se relacionan con la existencia de un conjunto de funciones y sus propiedades específicas.
 - Idoneidad
 - Exactitud
 - Interoperabilidad
 - Seguridad
 - Cumplimiento de normas
- Fiabilidad - atributos relacionados con la capacidad del software de mantener su nivel de prestación bajo condiciones establecidas durante un período establecido.
 - Madurez
 - Recuperabilidad
 - Tolerancia a fallos

- Usabilidad - atributos relacionados con el esfuerzo necesario para su uso, por un conjunto de usuarios.
 - Aprendizaje
 - Comprensión
 - Operatividad
 - Atractividad
- Eficiencia - atributos relacionados con la relación entre el nivel de desempeño y la cantidad de recursos
 - Comportamiento en el tiempo
 - Comportamiento de recursos
- Mantenibilidad - atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.
 - Estabilidad
 - Facilidad de análisis
 - Facilidad de cambio
 - Facilidad de pruebas
- Portabilidad - atributos relacionados con la capacidad de un sistema software para ser transferido desde una plataforma a otra.
 - Capacidad de instalación

- Capacidad de reemplazo
- Adaptabilidad
- Co-Existencia

El estándar de seguridad de la información ISO 17799:2000, es la forma generalizada de contar con un estándar que permite reconocer o validar el marco de referencia de seguridad aplicado por las organizaciones, el cual se basa principalmente en la primera parte del estándar BS 7799 conocida como Código de Prácticas (BS 7799 Part 1: Code of Practice), emitida por el BSI (British Standard Institute). El ISO 17799, es una guía de implementación del sistema de administración de seguridad de la información, y se enfoca en tutelar los siguientes principios de la seguridad informática:

- Confidencialidad - asegurar que únicamente personal autorizado tenga acceso a la información.
- Integridad - garantizar que la información no será alterada, eliminada o destruida por entidades no autorizadas.
- Disponibilidad - asegurar que los usuarios autorizados tendrán acceso a la información cuando la requieran.

Estos principios de protección de información son el núcleo básico, sin embargo se debe tener presente siempre que, dependiendo de la naturaleza y objetivos particulares, estos elementos de protección se pueden modificar o tener énfasis diferentes.

La ISO/IEC 19505 Information technology - Open Distributed Processing - Unified Modeling Language (UML), es un estándar que describe un lenguaje gráfico para visualizar,

especificar, construir y documentar un sistema. UML permite describir un esquemático del sistema, donde se incluyen aspectos conceptuales tales como los procesos, las funciones del sistema y los aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos, etc. Es importante remarcar que UML es un lenguaje para describir el modelo, para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. Los tipos principales de diagramas UML son:

- UML estructural: muestra la estructura estática de los objetos en un sistema.
- UML de comportamiento: muestra el comportamiento dinámico de los objetos en el sistema.
- UML de interacción: muestra una vista sobre los aspectos dinámicos de los sistemas modelados.

En el modelo de evaluación de Software que sigue la Universidad Tecnológica de Pereira se vislumbra un alineamiento con estos modelos internacionales al tener presente los siguientes criterios de evaluación, los cuales se pueden encontrar como características o sub-características de los modelos ISO:

- Robustez - sólido aún en situaciones difíciles.
- Extendibilidad - fácil de que le sean añadidas nuevas características.
- Desempeño - hace lo que tiene que hacer en el tiempo requerido, no desperdicia espacio en RAM ni en disco.

- Usabilidad o amigable al usuario - fácil de usar desde el punto de vista del usuario final.
- Integridad - que la información no se pierda ni se la puedan modificar o cambiar o capturar personas no autorizadas; o que la información almacenada permanezca consistente.
- Portabilidad - que pueda portarse fácilmente de una plataforma a otra.
- Compatibilidad - que sea compatible con anteriores versiones, si las hay.
- Mantenimiento - que sea de fácil mantenimiento.
- Documentación - que esté suficientemente documentado.
- Grado de invención o de innovación. Este aspecto, además se desglosa en las siguientes dos observaciones:
 - ¿Cómo es el desarrollo de la temática con relación a la productividad nacional en el área tratada (estado del arte)?
 - ¿Considera usted que el software tiene impacto y contribuye a las necesidades de la academia u organizaciones públicas o privadas en el ámbito regional, nacional o internacional?

Con el fin de comparar lo descrito por normas y por el modelo de evaluación de Software que sigue la Universidad Tecnológica de Pereira, se presenta el siguiente comparativo en el Cuadro 2:

Criterio	Modelo UTP	Modelos ISO
Robustez	Sólido aún en situaciones difíciles	Capacidad de reaccionar apropiadamente ante situaciones excepcionales
Extendibilidad	Fácil de que le sean añadidas nuevas características	Facilidad de adaptación del software a nuevos requisitos o cambios en la especificación.
Desempeño	Hace lo que tiene que hacer en el tiempo requerido, no desperdicia espacio en RAM ni en disco	Relacionado como eficiencia, o atributos relacionados con la relación entre el nivel de desempeño y la cantidad de recursos. Se tiene presente su comportamiento en el tiempo y su comportamiento de recursos.
Usabilidad o amigable al usuario	Fácil de usar desde el punto de vista del usuario final	Atributos relacionados con el esfuerzo necesario para su uso, por un conjunto de usuarios
Integridad	Que la información no se pierda ni se la puedan modificar o cambiar o capturar personas no autorizadas; o que la información almacenada permanezca consistente	Capacidad de proteger sus diferentes componentes contra los procesos o elementos que no tengan derecho de acceso a los mismos.
Portabilidad	Que pueda portarse fácilmente de una plataforma a otra	Capacidad de ser transferido y adaptado desde una plataforma a otra
Compatibilidad	Que sea compatible con anteriores versiones, si las hay	En la característica Portabilidad, las subcaracterísticas de: <ul style="list-style-type: none"> ■ Co-Existencia ■ Capacidad de reemplazo
Mantenimiento	Que sea de fácil mantenimiento	Definido como mantenibilidad, o atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software
Documentación	Que esté suficientemente documentado	Descripción mediante modelos UML y documentación general por manuales.

Cuadro 2: Cuadro comparativo de estándares.

Grado de invención o de innovación	¿Cómo es el desarrollo de la temática con relación a la productividad nacional en el área tratada (estado del arte)? ¿Considera usted que el software tiene impacto y contribuye a las necesidades de la academia u organizaciones públicas o privadas en el ámbito regional, nacional o internacional?	
------------------------------------	---	--

Cuadro 2: Cuadro comparativo de estándares, continuación.

A continuación, se muestra el análisis respectivo de calidad del software con el marco de referencia empleado por la Universidad Tecnológica de Pereira, con el fin de mostrar cómo estos aspectos son tenidos en cuenta dentro del desarrollo del software PiNet.

6.2. ANÁLISIS CALIDAD DE SOFTWARE PiNet

6.2.1. Robustez

PiNet es sólido ante diferentes tipos de acciones de entrada, ya sean eventos de ratón y teclado, o parámetros ingresados por el usuario. Para el ingreso de parámetros, se restringe desde un inicio al uso únicamente de valores numéricos, si el usuario tratar de acceder un carácter de naturaleza distinta, el sistema omitirá dicha acción. Para el caso de los eventos de ratón y teclado, se permite el accionamiento de ambos con la finalidad de interactuar con los diferentes objetos y propiedades de la interfaz, como lo es la adición de elementos dentro del área de trabajo por medio del ratón, o el uso de diferentes teclas para el accionamiento de eventos, tal como la eliminación de elementos

activos o adición de parámetros en las ventanas emergentes.

6.2.2. Extendibilidad

El modelo que rige el diseño del software se basa en el paradigma de la programación orientado a objetos, respetando cada una de sus propiedades intrínsecas como lo son la herencia, encapsulamiento, polimorfismo, entre otras. Debido a ello se facilita el realizar la adición de nuevas características al presente software en caso de ser necesario. A través del lenguaje unificado de modelado (UML), y en la sección 5.3 del manual de usuario, es posible observar el despliegue de cada uno de los elementos que lo componen, así como de la manera en que se interrelacionan.

6.2.3. Desempeño

PiNet posee pocas exigencias en cuanto a la relación desempeño-costos computacional. PiNet puede ejecutar redes hasta un aproximado de 150 elementos dentro del área de trabajo y sin demora apreciable en el tiempo de ejecución por parte del usuario. Para cumplir con dichas condiciones, el equipo anfitrión debe poseer las siguientes características (o superiores):

- Procesador Intel Core 2 Duo @ 2.0Ghz.
- 4 GB de memoria RAM.
- 500 MB de espacio libre en el disco duro.

6.2.4. Usabilidad o amigable al usuario

La interfaz gráfica de PiNet es de naturaleza intuitiva, cuenta con elementos visuales que facilitan la rápida comprensión de su funcionamiento, tanto para usuarios ya relacionados con la temática del campo, como para personas que no han tenido experiencia con programas similares. En caso de un usuario requerir de una guía exhaustiva, se dispone de un manual de usuario que clarifica cada aspecto necesario para el correcto aprovechamiento del software.

6.2.5. Integridad

El producto final a entregar consta de un archivo binario protegido contra escritura, esto es, un ejecutable que permite al usuario observar e interactuar con cada uno de los elementos y propiedades de la interfaz, todo ello bajo restricciones mínimas (necesarias para garantizar la robustez del sistema), como lo es el ingreso de parámetros y eventos de ratón y teclado. El usuario se encuentra en la posibilidad de almacenar cada uno de los diseños deseados, así como cargar archivos de diseños previos. Si el usuario corrompe archivos creados por el mismo, estos no serán ejecutados nuevamente dentro de la interfaz, y siempre garantizando que el usuario no pueda acceder al archivo binario ejecutable. Por cuestiones de evaluación, la presente distribución anexa dentro de sus elementos la carpeta de código fuente, esto con la finalidad de permitir al evaluador corroborar de manera propia cada uno de los aspectos mencionados dentro de la metodología implementada para el desarrollo del software.

6.2.6. Portabilidad

PiNet es un programa portable, no requiere de instalación dentro de un equipo anfitrión. La distribución del ejecutable cuenta con cada uno de los módulos y paquetes necesarios para su correcta operación, por tanto puede ser migrado entre diversas plataformas compatibles con sus requisitos, sin implicar deterioro o fallos en el sistema.

6.2.7. Compatibilidad

PiNet al ser un programa portable no requiere del uso de recursos compartidos propios del sistema operativo, implicando esto la no interferencia entre los diferentes hilos presentes al momento de ejecución del software, garantizando así la coexistencia con los demás procesos activos y su reinstalación (capacidad de reemplazo).

6.2.8. Mantenibilidad

El programa es construido en su totalidad por módulos, los cuales a su vez contienen objetos. Por tanto, en caso de ser requerida una intervención, es posible recurrir a los modelos UML que describen dichas estructuras y corregir o adicionar de manera satisfactoria la necesidad presentada. Los modelos UML se muestran y describen en la sección 5.3 del manual de usuario.

6.2.9. Documentación

Dentro de la sección 5.3 del manual de usuario de PiNet se ofrece a un mayor nivel de detalle cada uno de los aspectos que componen el software, desde la construcción hasta

el funcionamiento. Se cuenta con distintos diagramas UML, como lo son los diagramas de clasificación estructural, el de comportamiento dinámico, y la gestión del modelo. El presente manual de usuario y ayudas disponibles en la interfaz de usuario, también enriquecen la documentación disponible.

6.2.10. Grado de invención o de innovación

Para proveer información acerca del nivel de innovación con el que cuenta el software PiNet, se proporciona el documento anexo en la Figura 75 el cual certifica el impacto a nivel académico, y en su sociedad relacionada, presentado por PiNet, esto por medio de su uso como herramienta que permite mejorar los métodos de enseñanza - aprendizaje al poner en práctica conceptos y desarrollo de metodologías para analizar, implementar y sintetizar autómatas de estados finitos y sistema de eventos discretos con simulación determinística sin temporización o con temporización desde el punto de vista de los eventos o estados. Los impactos a nivel de invención o de innovación también se ven reflejados en un proyecto de grado, publicaciones en las que PiNet tuvo participación como elemento de modelamiento y en un Proyecto de Investigación activo con la Vicerrectoría de Investigación Innovación y Extensión de la Universidad Tecnológica de Pereira; estos impactos se listan a continuación.

Pereira, 17 de octubre de 2017

Ingeniero
Jhoniers Gilberto Guerrero Erazo
Vicerrector Académico
Presidente CIARP
Universidad Tecnológica de Pereira

Referencia: certificación de impacto del producto **PiNet** empleado por la comunidad de estudiantes.

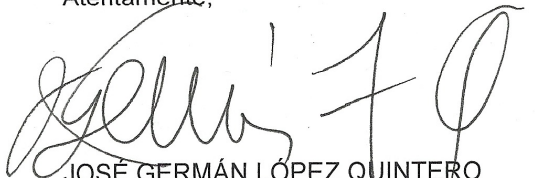
Cordial Saludo

Por medio del presente documento me permito certificar que el producto de software denominado **PiNet** es empleado por la comunidad de estudiantes y profesores del programa de Ingeniería Eléctrica de la Universidad Tecnológica de Pereira.

El impacto del producto PiNet empleado por la comunidad descrita, se evidencia principalmente en su uso como herramienta que permite mejorar los métodos de enseñanza – aprendizaje al permitir poner en práctica conceptos y desarrollo de metodologías para analizar, implementar y sintetizar autómatas de estados finitos y sistemas de eventos discretos con simulación determinística sin temporización y con temporización desde el punto de vista de los eventos o estados.

La plataforma de software desarrollada permite, además, realizar el análisis matricial y de propiedades de un sistema de eventos discretos, fuera de poder ser aplicada en la interpretación de descripciones dadas por lenguajes naturales y lógica temporal para generar estrategias de control efectivo, en aplicaciones de índole automático y en sistemas de navegación autónoma. Algunas áreas específicas del conocimiento donde se emplea la herramienta son las relacionadas con: Automatización de procesos, sistemas de eventos discretos, descripción de sistemas por lenguaje natural y navegación autónoma.

Atentamente,



JOSÉ GERMÁN LÓPEZ QUINTERO
Director Programa de Ingeniería Eléctrica
Universidad Tecnológica de Pereira

Impacto en proyecto de grado, a nivel de pregrado:

Título: Metodología para optimización, a través de algoritmos genéticos, de sistemas de manufactura flexible libres de bloqueos modelados mediante redes de Petri.

Universidad Tecnológica de Pereira - Ingeniería Eléctrica, 2017

Personas orientadas: Cristian Camilo Grisales Londoño, David Daniel Villegas Trujillo

Impacto a nivel de artículos y ponencias en eventos:

Título: Planificación de movimientos para sistemas de navegación autónoma terrestre utilizando lógica temporal lineal

Autores: Jorge Luis Martínez Valencia, Mauricio Holguín Londoño, Andrés Escobar Mejía

Evento: II Congreso Internacional Electromecánica y Eléctrica

ISBN: 978-9942-759-09-2

Lugar: Latacunga, Ecuador, julio 12-14 de 2017.

Título: Metodología para tratar el problema de explosión combinatorial de estados utilizando LTL en aplicaciones de navegación autónoma terrestre

Autores: Jorge Luis Martínez Valencia, Mauricio Holguín Londoño, Andrés Escobar Mejía

Evento: 3er Colombian Conference on Automatic Control (CCAC 2017)

Lugar: Cartagena de Indias, Colombia, octubre 17-20 de 2017.

Impacto a nivel de investigación:

Proyecto de investigación: “Metodología para sintetizar continuamente autómatas de estados finitos a través del uso de lenguajes formales y lógicas temporales, aplicada a la navegación autónoma terrestre en ambientes parcialmente controlados”

Grupo de Investigación: Gestión de Sistemas Eléctricos, Electrónicos y Automáticos

Investigador principal: Mauricio Holguín Londoño

Entidad patrocinadora: Universidad Tecnológica de Pereira

Código del proyecto en la universidad: 6-17-2

Producto esperado específico: Productos Tecnológicos certificados o validados: Diseño industrial, esquema de circuito integrado, software, planta piloto, prototipo industrial y signos distintivos. Para este caso se debe presentar un software.

Referencias

- [1] SILVA MANUEL, RECALDE L. *Redes de Petri continuas: Expresividad, análisis y control de una clase de sistemas lineales conmutados*.
- [2] SILVA MANUEL *Las Redes de Petri: en la Automática y la informática* Madrid, España, 1985. ISBN: 84 7288 0451
- [3] OROZCO ÁLVARO, GUARNIZO CRISTIAN, HOLGUÍN MAURICIO, *Automatismos Industriales* Universidad Tecnológica de Pereira, 2008
- [4] GIRAULT CLAUDE, VALK RUDIGER, *Petri Nets for Systems Engineering, A Guide to Modeling, Verification, and Applications*. Springer-Verlag, 2003. ISBN: 3-540-41217-4.

- [5] LINO OLIVEIRA, SZTAJNBERG, ALEXANDRE *Analizador e Simulador de Redes de Petri*, 2007
- [6] J. RUMBAUGH, I. JACOBSON, G. BOOCH *El Lenguaje Unificado de Modelado. Manual de referencia* Pearson Educación S.A. Madrid, 2000.