Universidad Técnica Estatal de Quevedo

Estudiante: Cedeño Orlando, Vilcacundo Jordy, Robalino Bryan y Orrala William

Curso: Ingeniería en Software 7mo “A”.

Tema: Modelamiento de sistemas distribuidos con Redes de Petri.

Asignatura: Aplicaciones Distribuidas.

Docente: Guerrero Ulloa Gleiston Cicerón.

Índice

Ilustraciones

1. Introducción

El avance tecnológico y la complejidad creciente de los sistemas informáticos han impulsado la necesidad de modelos y herramientas capaces de diseñar, analizar y validar estos sistemas de manera eficiente. En este contexto, las redes de Petri emergen como una herramienta poderosa y versátil para la modelación de sistemas, especialmente aquellos que son concurrentes, distribuidos o que presentan características de paralelismo. La importancia de las redes de Petri radica en su capacidad para representar formalmente el comportamiento dinámico de los sistemas, permitiendo una comprensión detallada de sus procesos y la interacción entre sus componentes.

Las redes de Petri son un formalismo que se utiliza para modelar los cambios entre estados en un tiempo discreto. Se han aplicado a varios sistemas, incluidos los sistemas concurrentes, los programas informáticos y los modelos epidemiológicos. Las redes de Petri se pueden usar para modelar sistemas simultáneos e identificar y resolver errores antes de la ejecución [1]. También se pueden usar para optimizar los programas de computadora al determinar la concurrencia máxima permitida por la estructura de la red de Petri [2]. Además, las redes de Petri se han utilizado para modelar y simular modelos epidemiológicos, centrándose en el cálculo del número de reproducción básico, que mide el número promedio de individuos secundarios durante su vida infecciosa [3]. PNet es una biblioteca de Python que simplifica el modelado y la simulación de redes de Petri mediante el uso de un lenguaje basado en texto y funciones habituales de Python para las reglas de transición [4].

1.1. Importancia de la modelación con redes de Petri

La modelación con redes de Petri ofrece un marco teórico sólido que facilita la representación gráfica y matemática de los sistemas. Esta dualidad permite a los investigadores y desarrolladores analizar propiedades críticas como la vivacidad, la seguridad, y la ausencia de condiciones de carrera, entre otras [5]. La capacidad de las redes de Petri para describir y analizar el flujo de información y el control en sistemas distribuidos y paralelos las convierte en una herramienta indispensable en la ingeniería de software y sistemas.

1.2. Objetivo

Explorar la teoría y aplicaciones de las redes de Petri para modelar sistemas distribuidos. Buscando entender cómo se pueden usar para diseñar, implementar y analizar sistemas complejos, presentando ejemplos prácticos y discutiendo casos de estudio para destacar ventajas y desafíos del uso de redes de Petri en este campo.

2. Fundamentos de redes de Petri

2.1. Conceptos de redes de Petri

Las Redes de Petri son una clase de modelos matemáticos y gráficos utilizados para representar sistemas dinámicos, concurrentes y distribuidos. Estas redes consisten en un grafo dirigido y bipartito, compuesto por dos tipos de nodos: plazas y transiciones [6]. Las plazas representan estados del sistema donde se almacenan recursos o tokens, mientras que las transiciones representan eventos o acciones que pueden ocurrir en el sistema. Los nodos están conectados por arcos que indican las relaciones de dependencia entre ellos [7].

El estado de una Red de Petri se define por el marcaje, que es la distribución de tokens en las plazas en un momento dado. El marcaje refleja el estado actual del sistema y cómo se distribuyen los recursos o la información en ese instante. La evolución de la red se produce mediante el disparo de transiciones, donde una transición puede activarse si tiene los tokens necesarios en sus plazas de entrada, consumiendo esos tokens y produciendo nuevos tokens en las plazas de salida [8].

Las Redes de Petri son ampliamente utilizadas en diversos campos, como la modelización de sistemas de manufactura, protocolos de comunicación, sistemas de control, entre otros, debido a su capacidad para representar de manera clara y concisa la dinámica de sistemas complejos con eventos concurrentes, paralelos, asincrónicos y estocásticos [9].

2.2. Elementos de una red de Petri

Plazas y Transiciones: Las plazas en una Red de Petri representan estados del sistema donde se almacenan recursos, tokens o información relevante. Por otro

lado, las transiciones representan eventos o acciones que pueden ocurrir en el sistema, como la ejecución de una tarea o la disponibilidad de un recurso [10].

Arcos: Los arcos en una Red de Petri son las conexiones entre plazas y transiciones, y determinan las relaciones de dependencia entre ellas. Un arco dirigido de una plaza a una transición indica que la transición requiere la presencia de tokens en esa plaza para poder dispararse. Por otro lado, un arco de una transición a una plaza indica que la transición produce tokens en esa plaza al activarse [10].

Tokens: Los tokens son entidades discretas que residen en las plazas y representan la disponibilidad de recursos, la ocurrencia de eventos o el estado del sistema en un momento dado. La presencia o ausencia de tokens en una plaza determina el estado del sistema en ese instante [10].

Disparo de Transiciones: Una transición en una Red de Petri puede dispararse si tiene al menos un token en cada una de sus plazas de entrada. Al dispararse, la transición consume los tokens de entrada y produce tokens en las plazas de salida, reflejando así un cambio de estado en el sistema y la ocurrencia del evento asociado a la transición [10].

Marcaje: El marcaje de una Red de Petri es la distribución de tokens en las plazas en un momento específico. Representa el estado actual del sistema y cómo se distribuyen los recursos o la información en ese instante. A medida que se disparan las transiciones, el marcaje evoluciona y el sistema experimenta cambios en su estado [10].

Regla de Disparo: La regla de disparo de una transición establece las condiciones bajo las cuales una transición puede activarse. Por lo general, una transición está habilitada para dispararse si tiene todos los tokens necesarios en sus plazas de entrada, cumpliendo así con los requisitos previos para su ejecución [10].

2.3. Propiedades y características de las redes de Petri

Las redes de Petri son un modelo matemático y gráfico utilizado para describir sistemas distribuidos, concurrentes y paralelos. Tienen propiedades y características distintivas que las hacen útiles para el análisis y diseño de sistemas [11].

2.3.1. Propiedades

Bipartitas: Una red de Petri es un grafo bipartito, lo que significa que consta de dos conjuntos de nodos: lugares (o estados) y transiciones. Los lugares contienen tokens que representan el estado del sistema, mientras que las transiciones representan eventos que pueden ocurrir para cambiar de un estado a otro [11].

No determinismo: Las redes de Petri pueden modelar sistemas no deterministas, donde múltiples transiciones pueden estar habilitadas simultáneamente y la elección de cuál disparar puede depender de reglas específicas [11].

Disparo de transiciones: Las transiciones en una red de Petri se disparan cuando tienen suficientes tokens en sus lugares de entrada. Esto permite modelar la evolución del sistema a medida que ocurren eventos [11].

Estado del sistema: El estado global del sistema se representa mediante la distribución de tokens en los lugares de la red de Petri. Esto facilita la visualización y comprensión del comportamiento del sistema [11].

2.3.2. Características

Modelado de sistemas concurrentes: Las redes de Petri son especialmente útiles para modelar sistemas concurrentes, donde múltiples eventos pueden ocurrir simultáneamente [12].

Análisis de propiedades: Permiten analizar propiedades importantes del sistema, como la alcanzabilidad de ciertos estados, la ausencia de deadlocks y la conservación de recursos [12].

Expresividad: Las redes de Petri son lo suficientemente expresivas como para modelar una amplia gama de sistemas, desde sistemas simples hasta sistemas complejos de tiempo real [12].

Verificación y validación: Se pueden utilizar para verificar y validar el comportamiento de sistemas antes de su implementación, lo que ayuda a identificar posibles errores y problemas de diseño [12].

3. Modelamiento de sistemas distribuidos

El modelado de sistemas distribuidos implica el desarrollo y verificación de sistemas complejos a gran escala [13]. Estos sistemas se caracterizan por su naturaleza distribuida, con componentes que son autónomos y se comunican asíncronamente [14]. Los formalismos de modelado tradicionales derivados de sistemas centralizados son poco realistas para sistemas distribuidos, por lo que se ha propuesto un modelo integrado cronometrado de sistemas distribuidos [15]. Este formalismo soporta la asincronía, autonomía, y localidad del comportamiento en sistemas distribuidos y puede traducirse en autómatas temporizados para su verificación [16]. El uso de este formalismo permite la verificación automatizada de propiedades como la libertad de bloqueo y la eficiencia de terminación [5]. Además, los sistemas de aprendizaje profundo distribuidos pueden modelarse y optimizarse para predecir los beneficios de rendimiento de la capacitación en sistemas distribuidos en comparación con máquinas individuales [16][17]. En general, el modelado de sistemas distribuidos es esencial para comprender y optimizar el comportamiento de los sistemas distribuidos a gran escala.

3.1. Ventajas del modelamiento con redes de Petri en sistemas distribuidos

El modelado de redes de Petri ofrece varias ventajas en sistemas distribuidos. En primer lugar, las redes de Petri proporcionan una notación gráfica que permite el modelado de sistemas complejos con muchos dispositivos operando secuencialmente o en paralelo [18]. Esto facilita la comprensión y la construcción de modelos del sistema [19]. Adicionalmente, las redes Petri tienen similitudes con otros lenguajes de modelado como SFC (GRAFCET), lo que facilita la síntesis de algoritmos de control discreto implementados con PLCs [20]. Además, las redes Petri se pueden utilizar para evaluar el desempeño del sistema modelado, buscar formas de aumentar el rendimiento y verificar el correcto funcionamiento de los sistemas de control [21]. La capacidad de simular y verificar el comportamiento del sistema utilizando redes Petri es particularmente útil para garantizar que no haya cuellos de botella y que la transmisión y el procesamiento de datos sean eficientes [22].

3.2. Pasos para el modelamiento de sistemas distribuidos

Requisitos del sistema: Comienza por comprender los requisitos del sistema distribuido. Esto implica identificar las necesidades funcionales y no funcionales del sistema, así como los requisitos de rendimiento, seguridad, escalabilidad, disponibilidad, etc [23].

Análisis de dominio: Realiza un análisis detallado del dominio del problema para comprender el contexto en el que el sistema distribuido operará. Esto puede incluir la identificación de los actores, las interacciones entre ellos, los datos involucrados y los flujos de información [23].

Modelado conceptual: Crea un modelo conceptual del sistema distribuido utilizando técnicas como diagramas de casos de uso, diagramas de clases, diagramas de secuencia, etc. Este modelo captura la estructura y el comportamiento general del sistema, sin entrar en detalles de implementación [23], [24].

Identificación de componentes: Identifica los componentes principales del sistema distribuido, como servicios, módulos, nodos, etc. Esto implica dividirlo en partes más pequeñas y manejables desarrolladas e implementadas de forma independiente [23], [24].

Diseño de la arquitectura: Define la arquitectura del sistema distribuido, incluyendo patrones de comunicación, distribución de componentes, gestión de la concurrencia, tolerancia a fallos, etc. Esto implica la elección de una arquitectura de referencia, cliente-servidor, basada en microservicios, arquitectura orientada a eventos, entre otras [24].

Modelado de datos: Diseña el modelo de datos que utilizará el sistema distribuido. Esto implica definir la estructura de los datos, las relaciones entre ellos, las restricciones de integridad, etc. Puedes utilizar herramientas como diagramas entidad-relación o diagramas de clases para este fin [24].

Diseño de interfaces: Define las interfaces de comunicación entre los diferentes componentes del sistema distribuido. Esto incluye los protocolos de comunicación, los formatos de mensaje, los contratos de servicio, etc [24].

Validación y verificación: Realiza pruebas de validación y verificación para asegurarte de que el modelo del sistema distribuido cumple con los requisitos

especificados. Esto puede implicar pruebas de unidad, pruebas de integración, pruebas de rendimiento, etc [24].

Implementación: Implementa el sistema distribuido utilizando las tecnologías y herramientas adecuadas. Esto implica traducir el modelo diseñado en código ejecutable y desplegarlo en un entorno de producción [23], [24].

Mantenimiento y evolución: Una vez que el sistema distribuido está en funcionamiento, es importante realizar un mantenimiento continuo y realizar mejoras periódicas para adaptarse a los cambios en los requisitos del negocio o en el entorno operativo [24].

3.3. Ejemplos de aplicaciones del modelamiento con redes de Petri

Manufactura y Producción: Las redes de Petri pueden utilizarse para modelar sistemas de producción en líneas de ensamblaje, procesos de fabricación y control de inventario. Por ejemplo, se pueden modelar el flujo de materiales y productos en una fábrica, identificando cuellos de botella y optimizando la eficiencia del proceso [25].

Sistemas de Comunicación y Protocolos: En el ámbito de las redes de computadoras y las telecomunicaciones, las redes de Petri pueden emplearse para modelar protocolos de comunicación, como el protocolo TCP/IP, y para analizar el rendimiento de redes informáticas, como la congestión en redes de área local (LAN) o en redes de área extensa (WAN) [25]

Sistemas de Transporte y Logística: Las redes de Petri son útiles para modelar sistemas de transporte, como sistemas de tráfico urbano, redes ferroviarias y sistemas de gestión de flotas. Pueden utilizarse para simular y optimizar el flujo de vehículos y pasajeros, así como para diseñar sistemas de control de tráfico más eficientes [25], [26].

Sistemas de Control y Automatización: En ingeniería de control y automatización, las redes de Petri se utilizan para modelar y analizar sistemas de control de procesos industriales, sistemas de control de tráfico aéreo, sistemas de control de tráfico marítimo, entre otros. Pueden ayudar a diseñar estrategias de control óptimas y a prevenir situaciones de conflicto [26].

Sistemas de Software y Hardware: En el desarrollo de software y hardware, las redes de Petri son útiles para modelar sistemas concurrentes, como sistemas

operativos multitarea, protocolos de comunicación entre componentes de hardware y software, y sistemas distribuidos. Ayudan a detectar y evitar problemas de concurrencia y sincronización [26], [27].

Biología y Bioinformática: En biología y bioinformática, las redes de Petri pueden utilizarse para modelar y simular sistemas biológicos, como vías metabólicas, redes de regulación génica y sistemas biológicos complejos. Ayudan a comprender mejor los procesos biológicos y a diseñar terapias más efectivas [27].

Procesos de Negocio y Gestión de Proyectos: Las redes de Petri se pueden utilizar para modelar y analizar procesos de negocio, como flujos de trabajo, procesos de producción y sistemas de gestión de proyectos. Ayudan a identificar cuellos de botella, mejorar la eficiencia y reducir los tiempos de entrega [27].

4. Implementación de sistemas distribuidos con redes de Petri

4.1. Herramientas y lenguajes de programación utilizados

CPN tools: CPN Tools es una herramienta destacada para editar, simular y analizar CP-nets, con una interfaz gráfica diseñada junto a expertos en HCI. Ofrece retroalimentación contextual, verificación de sintaxis incremental, y generación de código durante la construcción del modelo. Permite simular partes correctas del modelo, ignorando errores, modificarlo en simulación y continuar después de revisar las modificaciones. Además, facilita la especificación y verificación de propiedades del sistema mediante un lenguaje de consulta simple [28].

TINA toolset: ofrece un editor para redes de Petri descritas gráfica o textualmente, construcción de grafos de alcanzabilidad, análisis estructural, verificación de modelos LTL de estado/evento y análisis de trayectorias para redes de Petri temporizadas [29].

Petrify: Las redes de Petri, interpretadas como Gráficos de Transición de Señales, desempeñan un papel fundamental en la especificación y diseño eficiente de circuitos de control asíncronos. Estas redes, junto con herramientas como Petrify, permiten una representación visual y automatizada que simplifica el proceso de síntesis y reduce la propensión a errores asociada con la construcción manual de las redes de Petri [30].

PNML (Petri Net Markup Language): PNML es un formato de archivo de intercambio para redes de Petri basado en XML. Este artículo presenta un marco, llamado EZPetri, basado en PNML. EZPetri es una perspectiva y un conjunto de complementos de la plataforma Eclipse [31].

4.2. Consideraciones para la implementación de sistemas distribuidos

Concurrencia: Las Redes de Petri proporcionan una representación unificada para modelar y visualizar tanto el comportamiento secuencial como el paralelo de un controlador. Su lenguaje gráfico, fácil de comprender, combinado con un formalismo matemático sólido, representa una verdadera ventaja al diseñar sistemas concurrentes complejos [32].

Comunicación entre nodos: Se utilizan Redes de Petri en un entorno distribuido para gestionar adaptaciones de comportamiento entre nodos remotos. Cada nodo incluye una Red de Petri de contexto extendida con la capacidad de interactuar remotamente con otras Redes de Petri de contexto en diferentes nodos. Para validar el modelo en entornos distribuidos, se demuestra la semántica de diversas relaciones de dependencia de contexto en situaciones que podrían generar inconsistencias [33].

Distribución de recursos: Las redes de Petri ofrecen una opción para representar sistemas de comunicación en lugar de los sistemas de colas tradicionales, empleando nodos de bifurcación y unión. Esto es especialmente útil para modelar redes de comunicación que incorporan esquemas de sincronización, como redes con control de ventana, sistemas Kanban o colas finitas con bloqueo general [34].

Escalabilidad: La incorporación de TPN también posibilita la representación de limitaciones temporales, tales como plazos y liberaciones, así como el tiempo de procesamiento, proporcionando así un fundamento matemático sólido para la elaboración precisa de métodos de planificación. Asimismo, TPN ofrece un conjunto de técnicas consolidadas para analizar y verificar las propiedades estructurales y de comportamiento [35].

4.3. Casos de estudio de sistemas distribuidos implementados con redes de Petri

Sistemas de Control de Tráfico Aéreo Distribuido: El control de tráfico aéreo, una parte esencial del sistema de aviación constituye el departamento funcional más crucial para asegurar la seguridad y el orden de las aeronaves. La red de Petri se destaca como herramienta fundamental en la modelación y análisis de sistemas de eventos discretos, ya que puede representar con precisión la relación entre el estado y los cambios de estado del sistema, además de describir eventos que pueden ser inciertos o imprecisos [36].

Procesos de Fabricación Distribuida: Las Redes de Petri de Objetos (OPNs, por sus siglas en inglés) ofrecen un enfoque natural y modular para modelar una variedad de sistemas del mundo real. La traducción hacia Prolog, que conserva la estructura de las OPNs al codificar su semántica, elimina la necesidad de desplegarlas en una red de Petri plana. Este proceso de traducción brinda soporte tanto para las semánticas de referencia como de valor [37].

Sistemas de procesamiento y análisis: Las aplicaciones de Big Data han demostrado ser eficaces para analizar grandes volúmenes de datos, incluso aquellos que no siguen una estructura específica. Sin embargo, esta eficacia viene acompañada de nuevos desafíos. Un ejemplo de ello es la dificultad en prever el rendimiento de frameworks como Hadoop y Spark, una tarea que puede resultar costosa. Por lo tanto, se hace necesario proporcionar modelos que sirvan como un valioso respaldo para diseñadores y desarrolladores. En este contexto, se emplean redes de Petri fluidas para anticipar el tiempo de ejecución de aplicaciones MapReduce y Spark, una estrategia pertinente para la predicción del rendimiento en tiempo de ejecución [38].

Automatización Industrial Distribuida: Los sistemas de fabricación reconfigurables, apoyados por la Internet Industrial de las Cosas (IoT), son modulares y se integran fácilmente, facilitando reconfiguraciones eficientes de sistemas y componentes con un tiempo de inactividad mínimo. Comúnmente, los sistemas industriales emplean controladores secuenciales representados mediante Redes de Petri Interpretadas para el Control (CIPNs, por sus siglas en inglés) [39].

5. Análisis y validación de sistemas distribuidos modelados con redes de Petri

Es un enfoque integral para resolver problemas de seguridad de redes en sistemas descentralizados que utilizan redes de Petri. Este enfoque se centra en el análisis y verificación de modelos de ataque a la red, desde la generación automática de modelos de componentes hasta la mejora continua de simulaciones de ataques y estrategias de defensa. Una combinación de métodos como el procesamiento automatizado de bases de datos, la selección y construcción de modelos, así como una validación y verificación rigurosas, garantiza la precisión y confiabilidad de los modelos desarrollado [40].

Este es un ejemplo de un método para desarrollar sistemas celulares robóticos cooperativos con enfoque en la seguridad y la corrección, enfatiza el desarrollo de software basado en componentes y el uso de middleware ROS, y el uso de redes de Petri como enfoque basado en modelos, la adopción de modelos formales puede reducir el esfuerzo de prueba durante el desarrollo, ofreciendo un enfoque eficiente para abordar problemas de complejidad y garantizar la seguridad de los sistemas descentralizados con redes de Petri [41].

5.1. Métodos de análisis y validación

La importancia del análisis y la validación de datos en el contexto del aprendizaje automático y enfatiza que estos aspectos son esenciales para garantizar la precisión y confiabilidad del sistema. Un enfoque centrado en datos y la inspección de datos de TensorFlow como un medio para lograrlo. Es aplicable a sistemas descentralizados modelados con Petri, donde el análisis y la validación de datos son esenciales para garantizar el correcto funcionamiento y la precisión de todo el sistema [42].

El cambio en el desarrollo de sistemas hacia un desarrollo basado en modelos con énfasis en MBSE. Se enfatiza la importancia de tener modelos disponibles en una etapa temprana y la necesidad de realizar análisis formales para su verificación y validación, se realiza una revisión de la literatura sobre V&V MBSE, destacando objetivos y desafíos. El MBSE contribuye a las primeras etapas de V&V y señala áreas

para futuras investigaciones, particularmente en sistemas descentralizados modelados utilizando redes de Petri [43].

Este es un ejemplo de el desarrollo de un sistema de pronóstico y monitoreo de inundaciones crítico para la seguridad, se enfatiza la precisión y confiabilidad, durante el desarrollo se utilizan técnicas multiagente de Gaia y redes de Petri de colores jerárquicos (redes CP) para diseñar, analizar y probar el rendimiento del sistema en un entorno distribuido [44].

Este es un ejemplo con enfoque integral para modelar y probar la seguridad de la red utilizando redes de Petri extendidas. Primero, el modelo de ataque a la red se genera automáticamente a partir de una base de datos de modelos conocidos. Estos modelos se han mejorado para incluir operaciones de protección y acciones normales del usuario, luego se seleccionan y combinan subconjuntos de modelos para formar un modelo completo del sistema objetivo, el modelo fue probado y validado utilizando métodos apropiados. Finalmente, se simula un ataque de red al sistema objetivo y mejor la estrategia de defensa mediante el aprendizaje por refuerzo [45].

5.2. Técnicas de simulación y verificación

El uso de técnicas de minería de procesos para evaluar modelos de simulación y analizar sus resultados, se centra en la validación y validación de modelos de sistemas de eventos discretos, enfatizando la importancia de evaluar la adecuación de los modelos y evaluar la confiabilidad de los resultados, el uso de la minería de procesos es para analizar el comportamiento de la simulación y sugiere posibles direcciones para desarrollar programas para evaluar objetivamente la idoneidad de los modelos de simulación [46].

Las diferencias entre los métodos de verificación formal y basado en simulación, centrándose en la complejidad computacional involucrada en la verificación formal, la posibilidad de lograr complejidad polinómica para ciertos diseños (como los circuitos aritméticos), pero también se reconoce la dificultad de aplicarla directamente a circuitos más complejos (como los procesadores). Un nuevo método de verificación formal basado en diagramas de decisión binaria (BDD),

puede resolver operaciones de bucle único y de bucle múltiple y reducir el tiempo de ejecución [47].

En este contexto la importancia de la verificación formal en la verificación de circuitos digitales según su complejidad crece, los métodos de simulación no son suficientes y se utilizan métodos formales como BDD y SAT, Aunque pueden ser costosos, los esquemas derivados de BDD pueden probarse eficientemente en tiempo y espacio polinómicos, además, los esquemas KFDD pueden ser más pequeños y se analiza en detalle su complejidad de verificación, mostrando mejoras con respecto a BDD [48].

Un nuevo método para la verificación formal de modelos de simulación de eventos paralelos utilizando Entity Interaction Graph (EIG) y Computational Time Logic (CTL). Este método ha demostrado ser eficaz para verificar la exactitud del modelo EIG transformando el modelo en una estructura Kripke y utilizando herramientas de verificación como NuSMV. Esto sugiere que este enfoque puede mejorar la confiabilidad de simulaciones de eventos discretos paralelos [49].

5.3. Casos de estudio de análisis y validación de sistemas distribuidos

Este ejemplo es sobre una nueva tecnología que utiliza almacenamiento de energía en baterías (BESS) para controlar la potencia de los sistemas de generación distribuida (DG) para mitigar el impacto negativo de la generación distribuida basada en energías renovables (DG RE) en la calidad de la energía, carga de vehículos eléctricos (EV). El método se valida mediante un estudio de caso real utilizando un entorno de prueba MATLAB/Simulink y un alimentador de distribución IEEE de 13 nodos. La gestión inteligente de la batería mejora la calidad de la energía solar [50].

Este ejemplo sobre el desarrollo de un modelo base de red para respaldar la investigación relacionada con la descarbonización del sistema energético y la integración renovable descentralizada. El modelo propuesto se valida con datos reales de caudal y costo marginal de ubicación (LMP), demostrando su viabilidad y funcionalidad, su capacidad para adaptarse a diferentes configuraciones y

escenarios futuros es excelente, lo que la convierte en una herramienta útil para evaluar sistemas descentralizados [51].

Un nuevo método para dimensionar los sistemas de almacenamiento de energía en baterías (BESS) para transformar sistemas de energía aislados en microrredes flexibles y resistentes. Utiliza un modelo MILP de dos etapas para maximizar las cargas críticas y minimizar los costos durante las interrupciones, Pruebe con datos reales, analice el impacto de la energía solar fotovoltaica (PV) en la escala BESS y mida la mejora en la resiliencia con métricas específicas, contribuye significativamente a los desafíos energéticos asociados con el cambio climático y los eventos extremos [52].

Un estudio exhaustivo sobre la optimización y validación de sistemas híbridos de energía renovable para satisfacer las necesidades de vivienda en regiones específicas se utiliza herramientas de modelado como HOMER y MATLAB para desarrollar modelos y analizar su desempeño en diversas condiciones, además, se introduce el control predictivo del modelo para mejorar la eficiencia y la estabilidad del sistema, se aborda los desafíos de integrar múltiples fuentes de energía renovables en un entorno descentralizado [53].

6. Conclusiones

En conclusión, este documento ha demostrado la relevancia de las redes de Petri para el modelado, análisis, y validación de sistemas distribuidos. A través de la teoría y ejemplos prácticos, se ha evidenciado cómo estas redes facilitan la comprensión y gestión de la complejidad inherente a los sistemas que presentan comportamientos concurrentes y distribuidos. La utilidad práctica de las redes de Petri, apoyada por herramientas de simulación y técnicas de verificación, subraya su papel indispensable en la ingeniería de sistemas complejos. Así, las redes de Petri se consolidan como una herramienta esencial en el diseño y operación eficiente de sistemas distribuidos, impulsando la innovación y mejorando la fiabilidad en diversas áreas tecnológicas.

7. Practicas

7.1. Vilcacundo Jordy

7.1.1. Problema

Sistema de estacionamiento automatizado.

7.1.2. Modelo

Imaginen una ciudad donde encontrar un lugar de estacionamiento se convierte en una tarea diaria estresante para los conductores, contribuyendo al tráfico congestionado, la contaminación ambiental y la insatisfacción general. La solución propuesta es la implementación de un Sistema de Estacionamiento Automatizado (SEA), diseñado para optimizar la gestión y asignación de espacios de estacionamiento, reduciendo así el tiempo que los conductores pasan buscando un lugar donde aparcar.

7.1.3. Solución

Para el proceso de elaboración de la práctica se hizo uso del simulador HPetri Sim, en el cual se realizó el modelo y simulación del ejercicio.

1. Primero debemos establecer las plazas o lugares que vamos a tener en cuenta

- Vehículo en la entrada
- Tarjeta aprobada
- Tarjeta rechazada
- Barrera abierta
- Vehículo ingresando
- Vehículo fotografiado
- Vehículo estacionado
- Barrera cerrada
- Estacionamiento ocupado
- Estacionamiento disponible

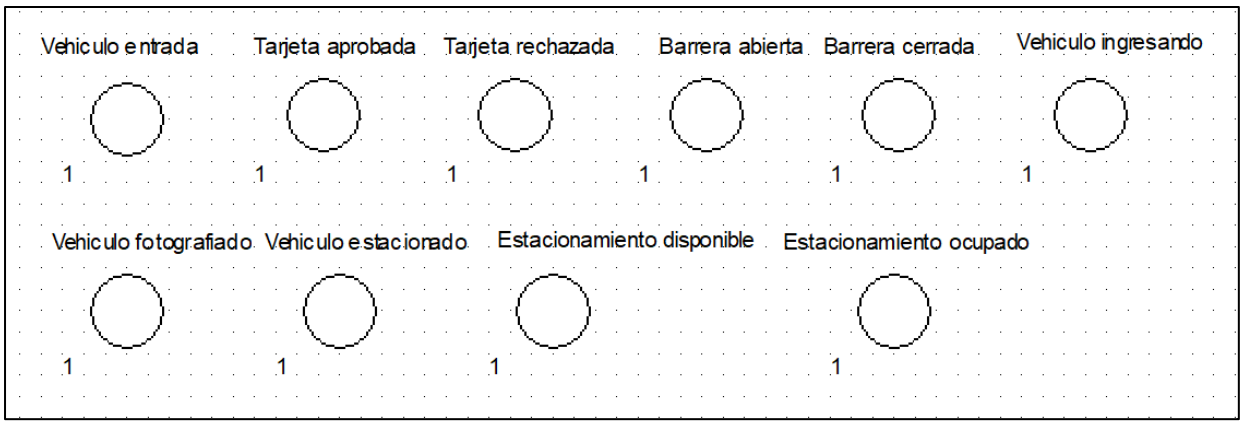

Ilustración 1: Plazas o Lugares practica 1, Vilcacundo Jordy.

2. Segundo establecemos las transiciones que se van a realizar.

- Lectura de tarjeta valida
- Lectura de tarjeta invalida
- Pagar ticket
- Señal de ok
- Sensor de ingreso
- Sensor de cámara
- Sensor detector de ocupación
- Valida estacionamiento ocupado
- Valida estacionamiento ok

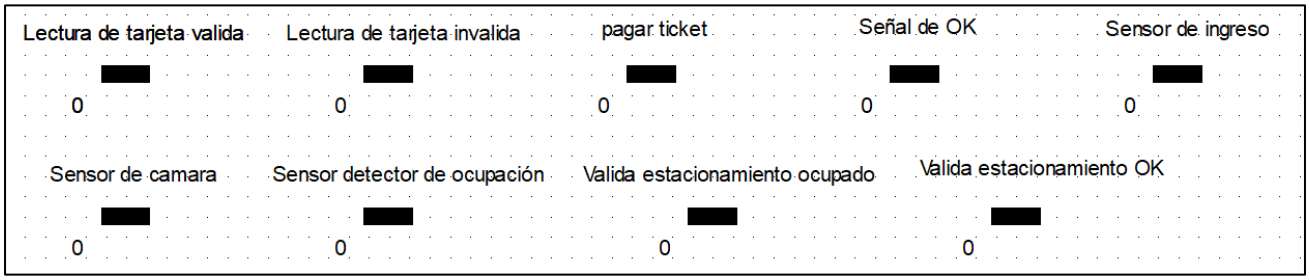

Ilustración 2: Transiciones para realizar practica 1, Vilcacundo Jordy.

3. Realizamos las conexiones mediante el uso de arcos.

La plaza "Vehículo entrada" se conecta mediante arcos a las transiciones "Lectura de tarjeta valida, Lectura de tarjeta invalida, Pagar ticket".

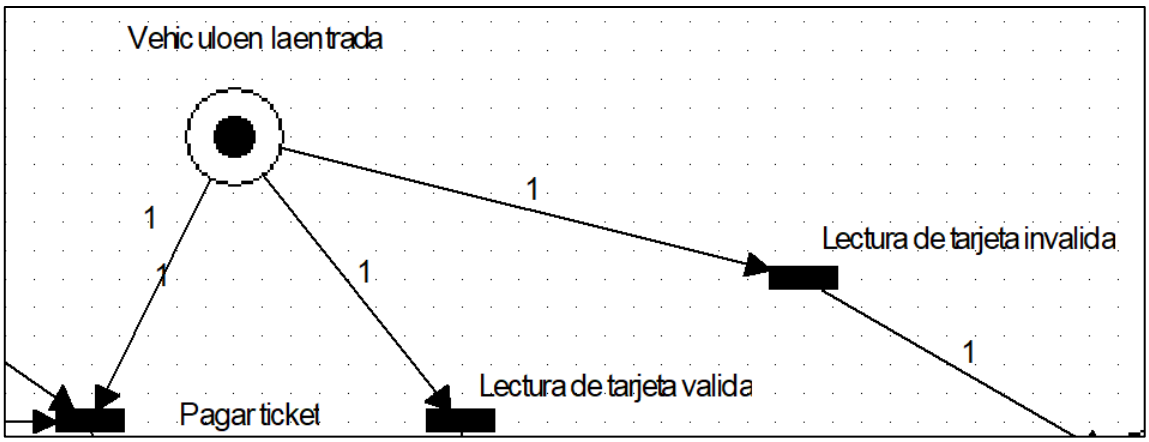

Ilustración 3: Conexión entre plaza y transiciones iniciales, Vilcacundo Jordy.

Al conectar la plaza con las transiciones, tenemos tres escenarios que pueden suceder al momento de realizar el proceso.

Como primer escenario tenemos, la conexión de la transición “Lectura de tarjeta invalida” con la plaza “Tarjeta rechazada”, en caso de que el vehículo que está en la entrada, al momento de dar lectura a su tarjeta es invalida, el proceso finaliza completamente ya que no tiene permitido el ingreso al estacionamiento.

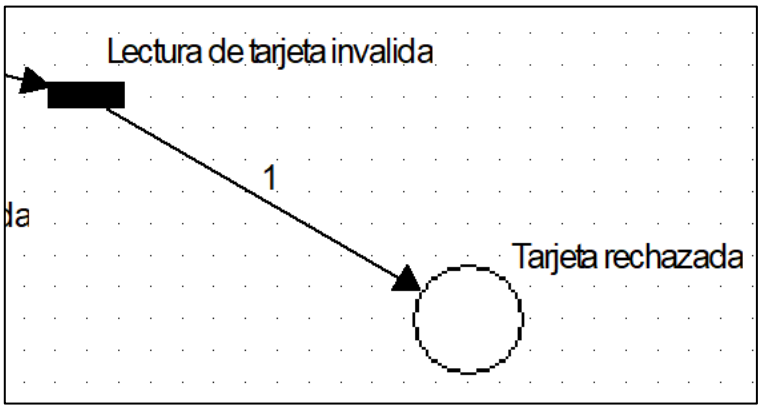

Ilustración 4: Conexión entre la transición "Lectura de tarjeta invalida" con la plaza "Tarjeta rechazada", Vilcacundo Jordy.

Ahora como segundo escenario tenemos, la conexión de la transición “Lectura de tarjeta valida” la cual envía el token a la plaza “Tarjeta aprobada” la cual dispara una señal a la transición “Señal de OK”.

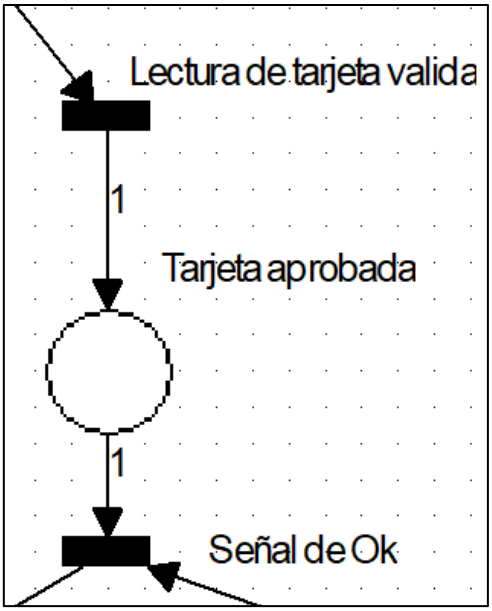

Ilustración 5: Conexión secuencial entre la transición "Lectura de tarjeta valida", la plaza "Tarjeta aprobada" y la transición "Señal de OK", Vilcacundo Jordy.

Una vez que se dispara la señal de OK, los tokens se desplazan hacia la plaza “Barrera abierta”, debido a que hay dos tokens, el primero es el vehículo y el segundo es el token de la plaza “Barrera cerrada”.

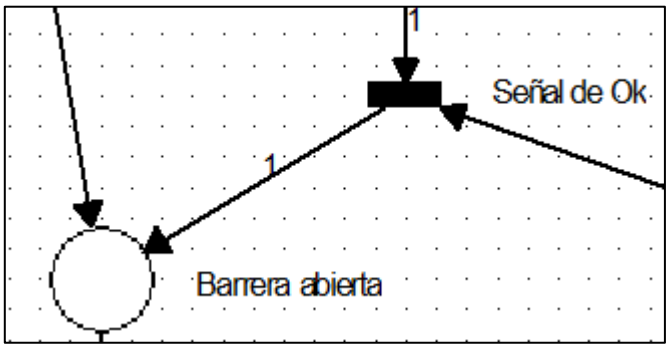

Ilustración 6: Unión en la transición "Señal de OK" hacia la plaza "Barrera abierta", Vilcacundo Jordy.

Luego de que los dos tokens se encuentren en la plaza “Barrera abierta”, los tokens se bifurcan en la transición “Sensor detector de ingreso”, donde una vez que se detecte el ingreso del vehículo la barrera se cerrara, luego de eso el vehículo continuara ingresando hacia el “Sensor de camara” donde el vehículo será fotografiado ingresando al estacionamiento, luego de eso el “Sensor detector de ocupación” detectara el vehículo estacionado.

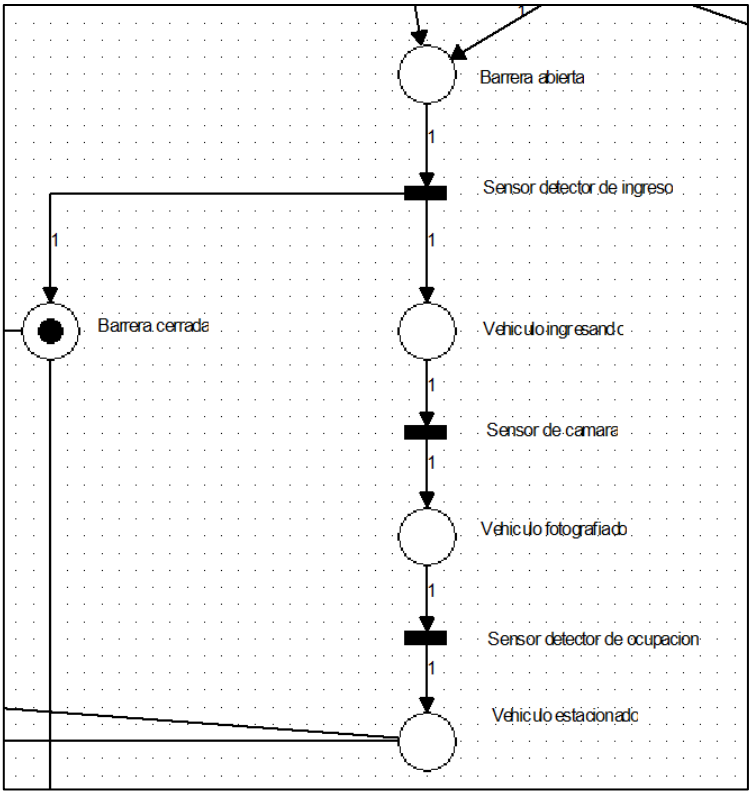

Ilustración 7: Proceso de ingreso al parqueadero, Vilcacundo Jordy.

Una vez que el vehículo se encuentre estacionado, el token viajara a la transición "Validar estacionamiento ocupado", en caso de ser así el token se ubicara en la plaza "Estacionamiento ocupado", una vez que el vehículo se retire, el token viajara a la transición "Validar estacionamiento OK" donde se verificara si el estacionamiento se encuentra vació, seguido a eso el token viaja a la plaza "Estacionamiento disponible".

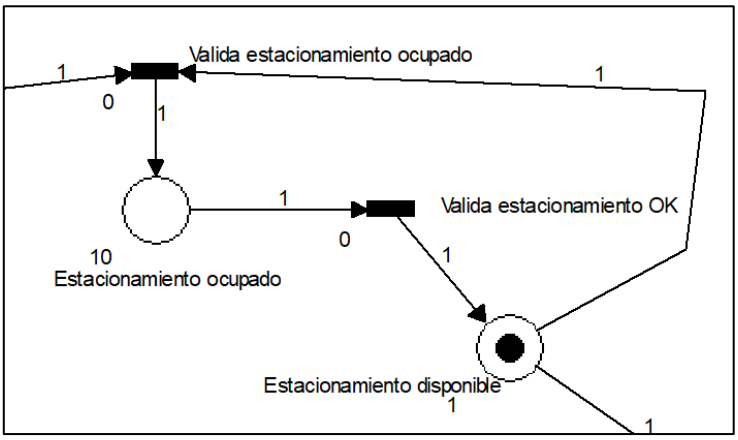

Ilustración 8: Validación del espacio de estacionamiento, Vilcacundo Jordy.

Ahora para el tercer escenario seria la transición “Pagar ticket” donde el vehículo deberá pagar un ticket para tener acceso al estacionamiento, una vez se haga esa transición se repite todo el proceso ya especificado.

7.1.4. Diagrama Completo

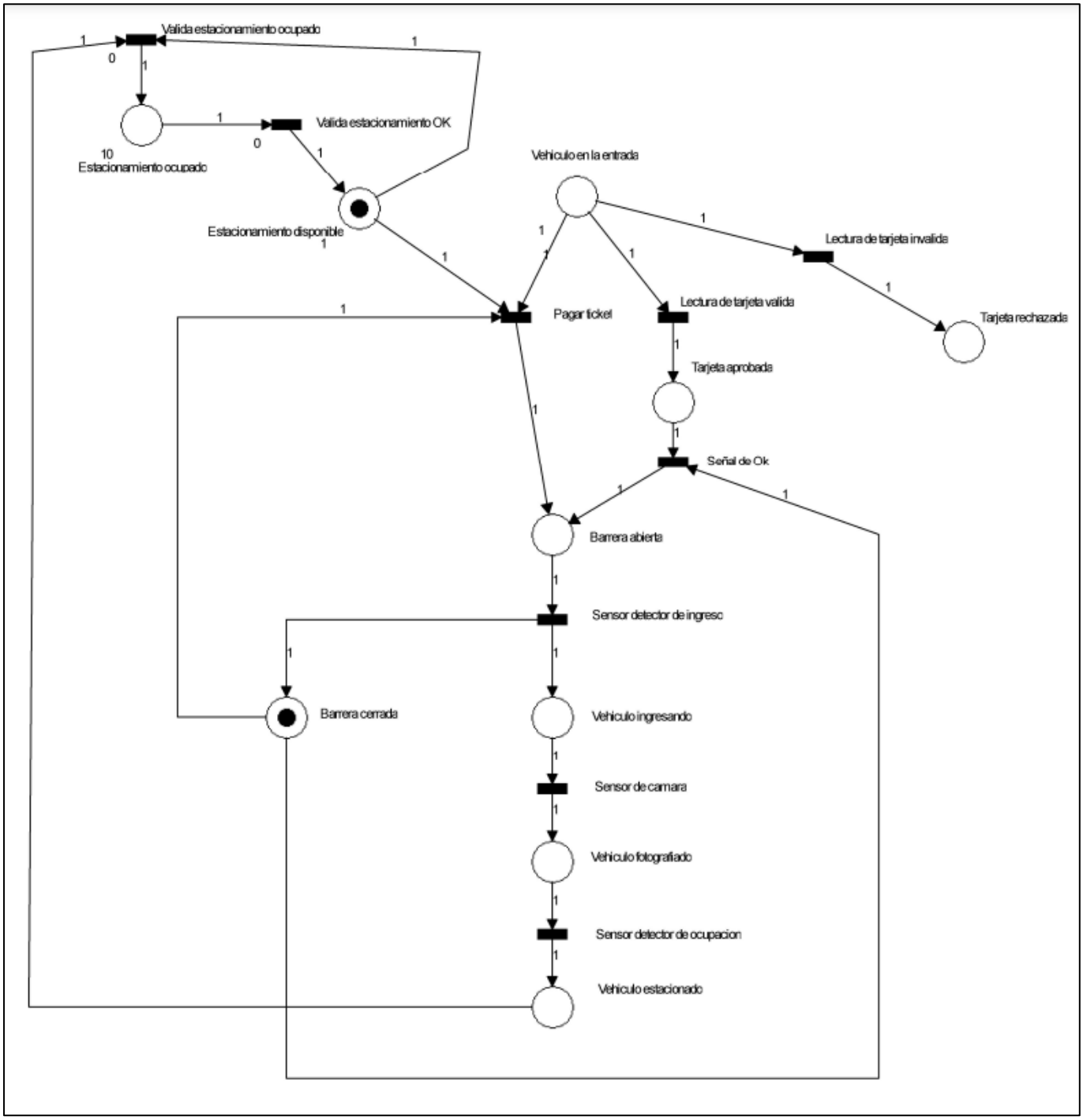

Ilustración 9: Diagrama de Petri desarrollado en base a la practica 1, Vilcacundo Jordy.

7.2. Cedeño Orlando

Implementación de Aplicaciones Distribuidas: Simulación del Proceso de Fabricación mediante Diagramas de Petri

7.2.1. Objetivo

Utilizar un Diagrama de Petri para modelar y simular el proceso de fabricación de un producto genérico (PT), basándose en el Bill of Materials (BOM) proporcionado y las existencias y capacidades de almacenamiento actuales.

7.2.2. Problema

La empresa Ensamblac se especializa en la fabricación de un producto genérico (PT). El proceso de fabricación implica el ensamblaje de varios subcomponentes y materias primas en un producto terminado. La cadena de producción se ha detallado en un Diagrama de Petri, que representa las materias primas (MP1, MP2, MP3, MP4), los sub-ensambles (SE1, SE2), y el producto terminado (PT). El objetivo es utilizar un Diagrama de Petri para modelar y simular el proceso de fabricación del producto genérico (PT), basándose en el Bill of Materials (BOM) proporcionado y las existencias y capacidades de almacenamiento actuales.

El diagrama de materiales (BOM) especifica las cantidades requeridas de cada materia prima para producir los sub-ensambles y el producto terminado.

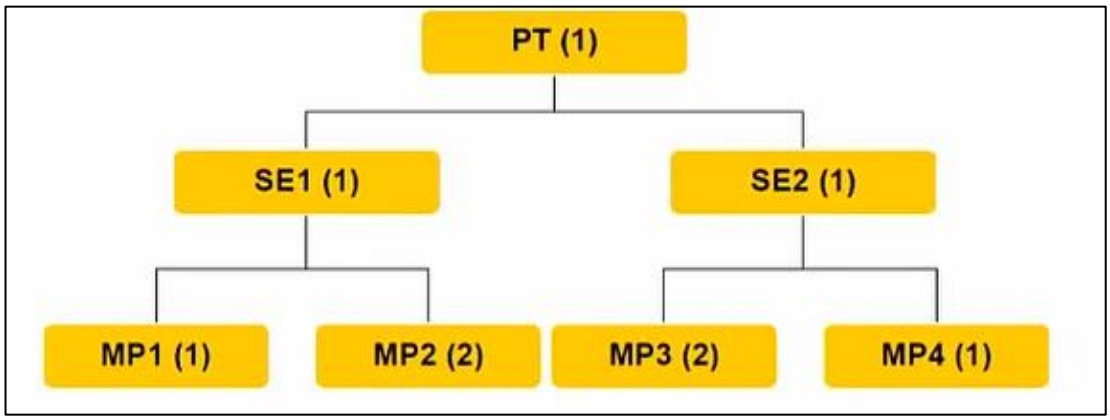

Ilustración 10: Diagrama de Materiales, Cedeño Orlando.

El inventario actual y la capacidad de almacenamiento para cada componente del proceso se deben analizar detenidamente.

Materiales	Existencias	Capacidades de las zona de Almacenamiento
PT	4	50
SE1 (Buffer 1)	3	10
SE2 (Buffer 2)	5	10
MP1	5	100
MP2	2	100
MP3	0	100
MP4	3	100

Ilustración 11: Datos de materiales, existencias y capacidades, Cedeño Orlando.

7.2.3. Modelo

- Cálculo de pedidos materia prima 1.

RESULTADOS MATERIA PRIMA1	
Total de unidades de PT solicitadas	50
Menos unidades de PT disponibles	4
Total de unidades a fabricar	46
Menos unidades de ensamble 1 (ES1) disponible	3
Total de ensambles 1 (ES1) requeridos	43
Multiplicar Requerimientos de MP1 por ES1	1
Total de MP1 requerida	43
Menos MP1 en existencia o disponible	5
Total MP1 a solicitar a PMP1	38

Ilustración 12: Pedidos Materia Prima 1, Cedeño Orlando.

- Cálculo de pedidos materia prima 2.

RESULTADOS MATERIA PRIMA2	
Total de unidades de PT solicitadas	50
Menos unidades de PT disponibles	4
Total de unidades a fabricar	46
Menos unidades de ensamble 1 (ES1) disponible	3
Total de ensambles 1 (ES1) requeridos	43
Multiplicar Requerimientos de MP2 por ES1	2
Total de MP2 requerida	86
Menos MP1 en existencia o disponible	2
Total MP2 a solicitar a PMP1	84

Ilustración 13: Pedidos Materia Prima 2, Cedeño Orlando.

- Cálculo de pedidos materia prima 3.

RESULTADOS MATERIA PRIMA3	
Total de unidades de PT solicitadas	50
Menos unidades de PT disponibles	4
Total de unidades a fabricar	46
Menos unidades de ensamble 2 (ES2) disponible	5
Total de ensambles 1 (ES2) requeridos	41
Multiplicar Requerimientos de MP3 por ES2	2
Total de MP3 requerida	82
Menos MP3 en existencia o disponible	0
Total MP3 a solicitar a PMP1	82

Ilustración 14: Pedidos Materia Prima 3, Cedeño Orlando.

- Cálculo de pedidos materia prima 4.

RESULTADOS MATERIA PRIMA4	
Total de unidades de PT solicitadas	50
Menos unidades de PT disponibles	4
Total de unidades a fabricar	46
Menos unidades de ensamble 2 (ES2) disponible	5
Total de ensambles 1 (ES2) requeridos	41
Multiplicar Requerimientos de MP4 por ES2	1
Total de MP4 requerida	41
Menos MP4 en existencia o disponible	3
Total MP4 a solicitar a PMP1	38

Ilustración 15: Pedidos Materia Prima 4, Cedeño Orlando.

Se modela el Diagrama de Petri en base a los siguientes elementos:

- **Lugares:** Los lugares representarán los inventarios de materias primas (MP1, MP2, MP3, MP4), sub-ensambles (SE1, SE2) y el producto terminado (PT).
- **Transiciones:** Las transiciones representarán las operaciones de ensamblaje.
- **Arcos dirigidos:** Conectarán los lugares con las transiciones, mostrando el flujo del proceso de fabricación.
- **Fichas:** Se indicarán las cantidades de cada componente en sus respectivos lugares.
- **Crear los lugares:** Se crearán lugares para cada uno de los materiales y sub-ensambles, así como para el producto terminado.
- **Asignar fichas a los lugares:** Se asignarán fichas a los lugares según las existencias actuales indicadas en la tabla.

- **Establecer las transiciones:** Se establecerán transiciones que muevan las fichas de las materias primas a los sub-ensambles y de los sub-ensambles al producto terminado, de acuerdo con las cantidades requeridas indicadas en el BOM.
- **Conectar los lugares con las transiciones mediante arcos dirigidos**: Los arcos de entrada a una transición se conectarán desde los lugares que representan los componentes necesarios para el ensamblaje, y los arcos de salida irán hacia los lugares que representan el componente ensamblado.
- **Agregar las capacidades de las zonas de almacenamiento**: Se agregarán como una restricción el número máximo de fichas que puede tener cada lugar.

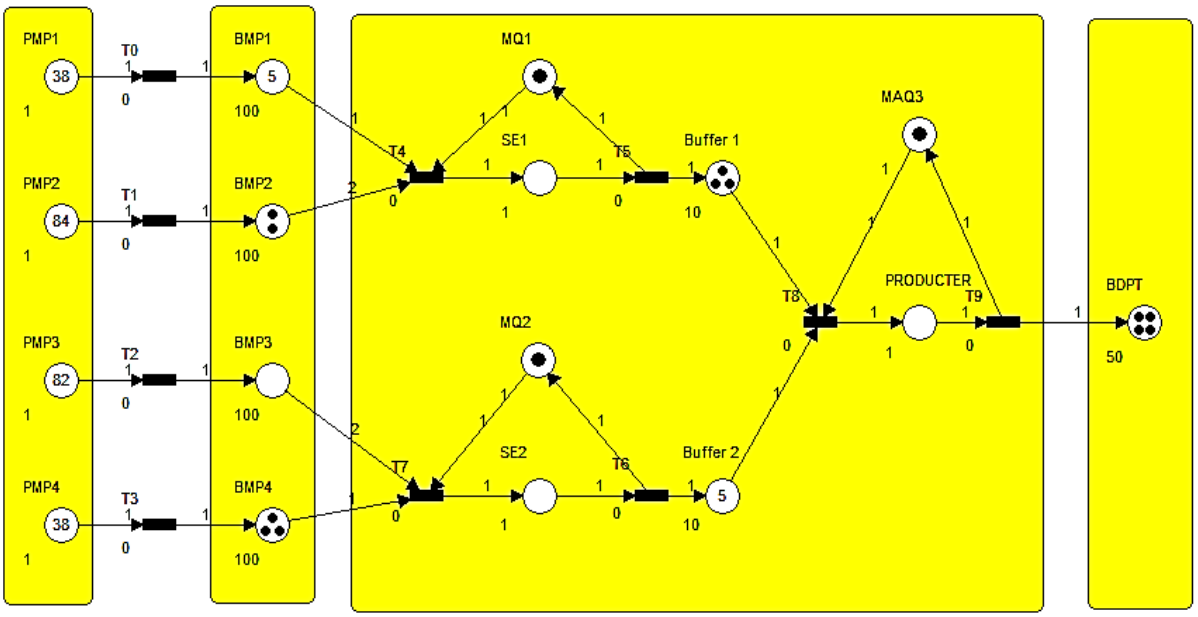

Ilustración 16: Diagrama de Petri Pre-Simulación, Cedeño Orlando.

Una vez realizada la simulación, se obtienen las siguientes soluciones:

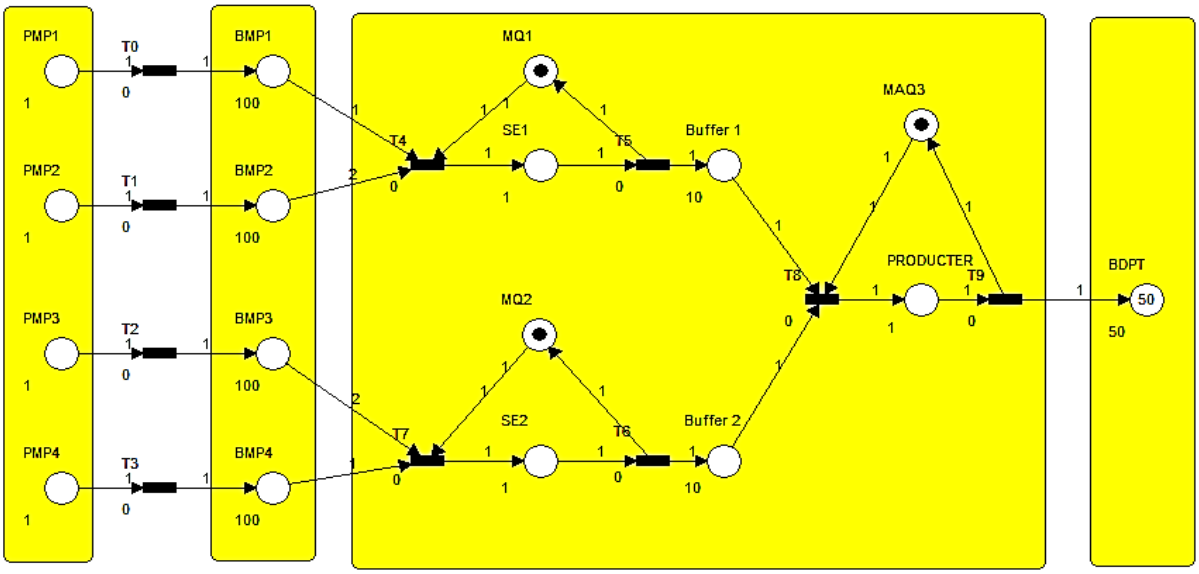

Ilustración 17: Diagrama de Petri Post-Simulación, Cedeño Orlando.

7.2.4. Solución

- El Inventario Final reveló que se han producido y están disponibles al final de la simulación una cantidad específica de componentes en cada lugar designado. Por ejemplo, la presencia de fichas en los lugares correspondientes a subensambles (SE1 o SE2) o al producto terminado (PT) indica cuántos de esos componentes se han fabricado.
- El Uso de Materias Primas se evaluó mediante la presencia o ausencia de fichas en los lugares designados para las materias primas (MP1, MP2, MP3, MP4). La ausencia de fichas en estos lugares señala que se han utilizado todas las materias primas disponibles para la fabricación, mientras que las fichas restantes indican un excedente.
- Se observó la Capacidad de Almacenamiento para determinar si algún lugar alcanzó su capacidad máxima. La presencia de una cantidad de fichas igual a la capacidad máxima en un lugar indica que está lleno y no puede aceptar más componentes sin exceder su capacidad de almacenamiento.
- Se identificaron posibles Cuellos de Botella y se evaluó la Eficiencia del proceso. La falta de fichas en los lugares de entrada de una transición indica que no se pueden producir más componentes debido a la escasez de

materiales. Además, el exceso de fichas en algún lugar puede indicar un cuello de botella en el proceso posterior, lo que señala áreas que requieren atención para mejorar la eficiencia.

- El Cumplimiento de Metas de Producción se determinó comparando la cantidad de fichas en el lugar correspondiente al producto terminado (PT) con la meta de producción deseada. Si el número de fichas es menor que la meta establecida, esto indica que las metas de producción no se están cumpliendo.

7.2.5. Archivo de Ejercicio en HPSim

https://drive.google.com/drive/folders/1di7S4FIjarpy4CGHFSsM4OANP5Cy4JtY?usp=sharing

7.3. Robalino Bryan

7.3.1. Problema

Simular un proceso industrial utilizando redes de Petri, mostrando el procedimiento paso a paso con robots y máquinas. La red de Petri permite simular el flujo de piezas desde un almacén de entrada, pasando por máquinas de procesamiento, hasta llegar a un almacén de salida.

7.3.2. Modelo

Se detalla el procedimiento de traslado de piezas entre almacenes, robots y máquinas en la red de Petri.

El primer nodo representa los espacios libres en el almacén.

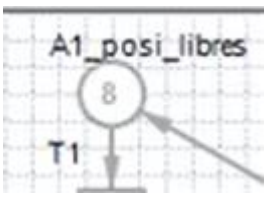

Ilustración 18: Espacios libres en almacén, René García 2021.

El segundo nodo indica las posiciones ocupadas que se van ocupando poco a poco en el almacén.

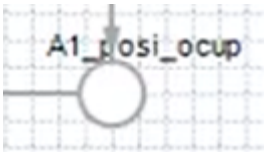

Ilustración 19: posiciones ocupadas, René García 2021.

Se definen los robots y máquinas disponibles para el proceso de almacenamiento y procesamiento de piezas.

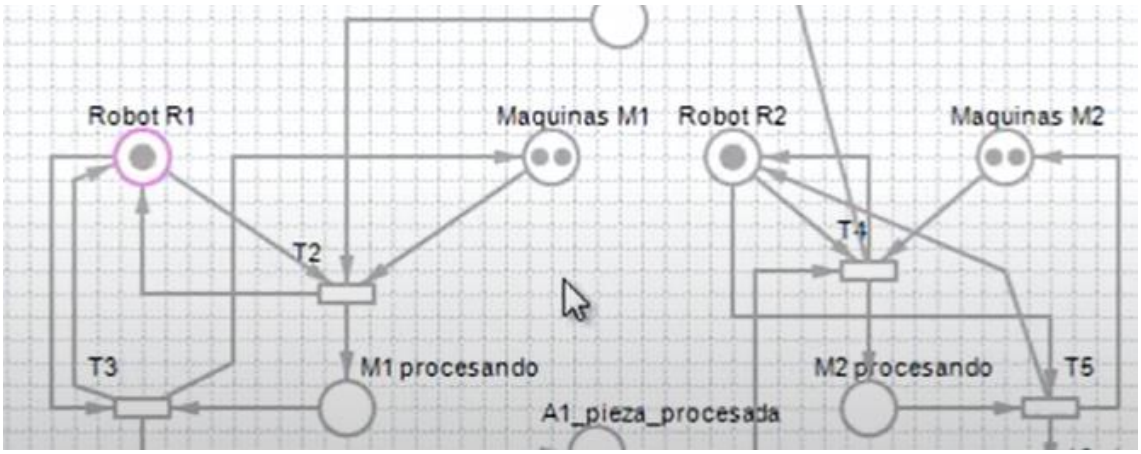

Ilustración 20: Definición de Robots, René García 2021.

Proceso de automatización de una línea de producción, Una vez procesada la pieza, el robot 2 la traslada al almacén de salida.

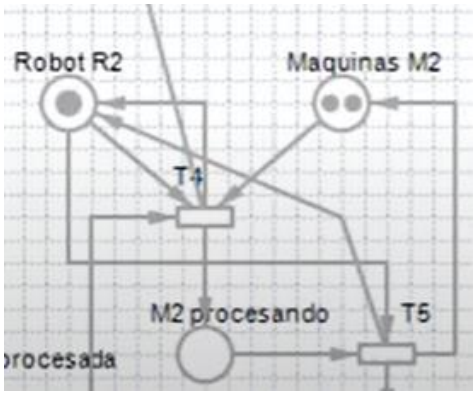

Ilustración 21: Automatización de líneas de producción, René García 2021.

Se simula el proceso de producción paso a paso, considerando cada transición y su activación. El robot 1 traslada la pieza a las máquinas M1 para su procesamiento.

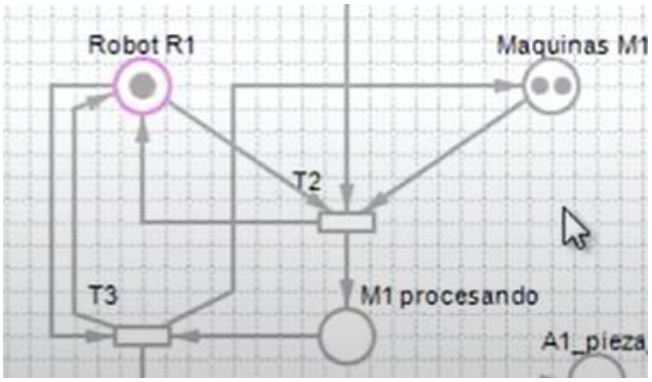

Ilustración 22: Procesamiento Robot 1 y Máquina M1, René García 2021.

El robot R1 transporta piezas procesadas al almacén 1.

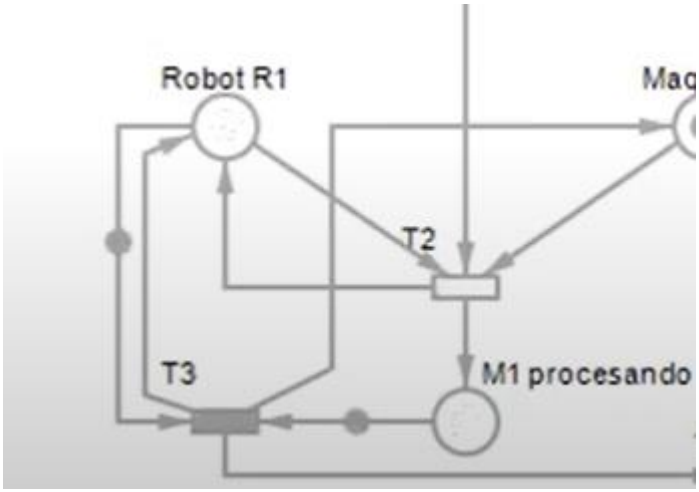

Ilustración 23: Transporte de Piezas Robot 1, René García 2021.

El robot R2 se encarga de trasladar piezas procesadas a las máquinas M2.

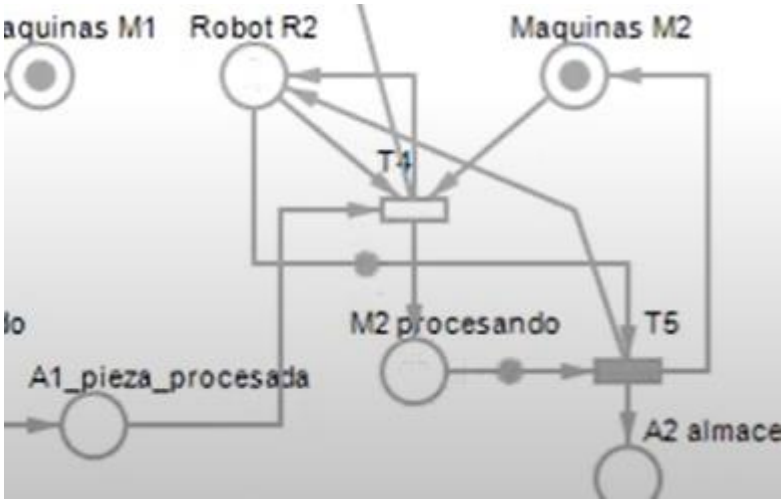

Ilustración 24: Transporte de piezas Robot 2, René García 2021.

La primera pieza procesada se dirige al almacén de salida después de completar el proceso.

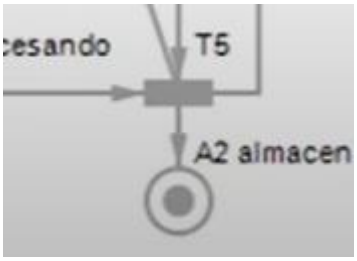

Ilustración 25: Transporte al almacén, René García 2021.

7.3.3. Solución

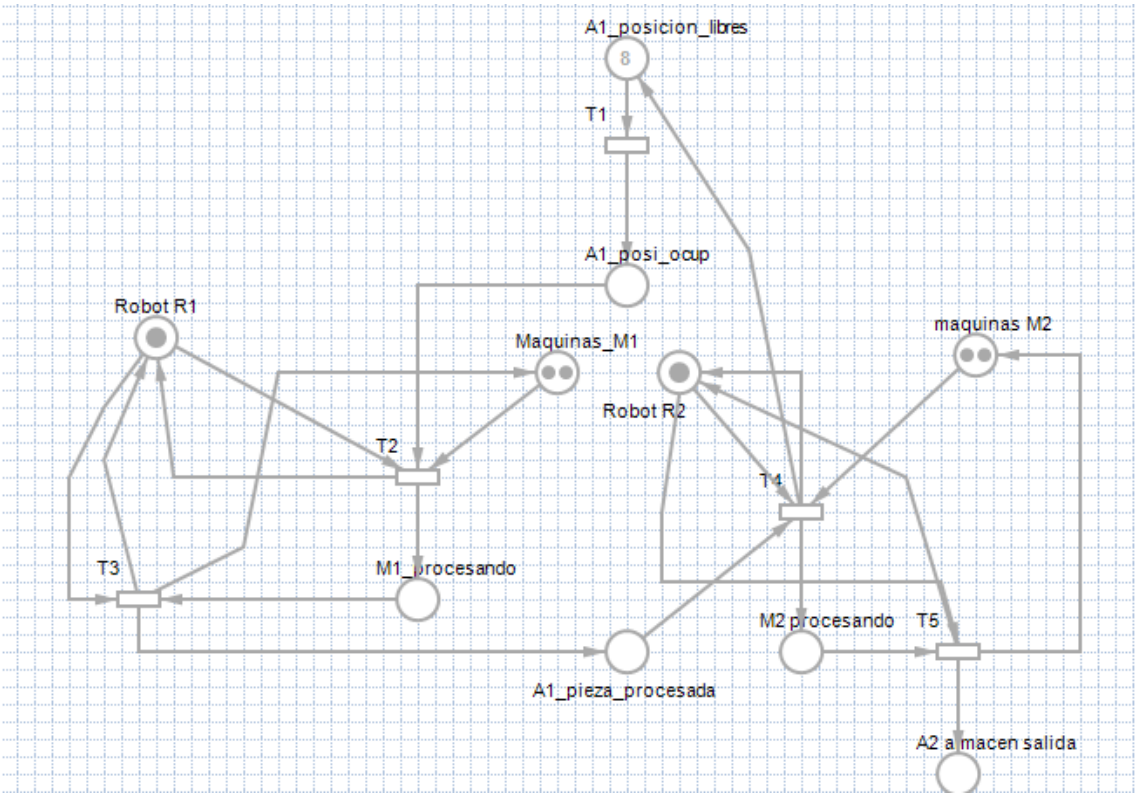

Ilustración 26: Diagrama de Petri Final, Robalino Bryan.

7.4. Orrala William

7.4.1. Problema

Este problema puede dar una idea clara del potencial de una PN, el problema consiste de cinco Filósofos que en forma alternada piensan y comen. Están sentados en una mesa circular donde ha sido depositada comida china, cada filósofo tiene frente a él un plato donde servirse y entre cada uno de ellos un palillo

chino. Como sabemos, para comer comida china necesitamos dos palillos, entonces cada filosofo para comer debe tener dos palillos en su poder.

El problema está en que si cada filósofo tiene un sólo palillo en su poder todos los filósofos entrarán en una espera eterna para comer, otro problema puede surgir y es el que un filósofo obtenga siempre los dos palillos y se mantenga comiendo, lo que producirá que los otros filósofos no puedan comer (cayendo en un estado de inanición). Analicemos los eventos y condiciones de este problema, así como las precondiciones y postcondiciones para un sólo filósofo y luego para más.

7.4.2. Modelo

Condiciones	
1	El palillo 1 está ocupado
2	El palillo 2 está desocupado
3	El palillo1 está ocupado
4	El palillo 2 está ocupado
5	El filósofo está meditando
6	El filósofo está comiendo
Eventos	
1	El filósofo empieza a meditar
2	El filósofo empieza a comer
Condiciones iniciales: 1,2, 5	

Eventos	Precondiciones	Postcondiciones

1	6	1,2,5
2	1,2,5	3,4,6

Sí ahora se trata de dos filósofos se obtendrá lo siguiente:

Condiciones	
1	El palillo 1 está ocupado
2	El palillo 2 está desocupado
3	El palillo1 está ocupado
4	El palillo 2 está ocupado
5	El filósofo 1 está meditando
6	El filósofo 1 está comiendo
7	El filósofo 2 está meditando
8	El filósofo 2 está comiendo
Eventos	
1	El filósofo 1 empieza a meditar
2	El filósofo 1 empieza a comer
3	El filósofo 2 empieza a meditar
4	El filósofo 2 empieza a comer

Condiciones iniciales: 1,2, 5		
Eventos	Precondiciones	Postcondiciones
1	6	3, 4, 5
2	3, 4, 5	1, 2, 6
3	8	3, 4, 7
4	3, 4, 7	1, 2, 8

Como sabemos las condiciones corresponden a nodos y los eventos a transiciones, las condiciones 1 y 3, 2 y 4 pueden modelarse con un sólo nodo. El problema para cinco filósofos es modelado como se muestra en la siguiente figura. Los nodos P1 y P5 representan los palillos, al inicio cuando todos los filósofos piensan, cada nodo tiene un token. Cada filósofo es representado por dos nodos M y C representan los estados meditando y comiendo respectivamente. Para que un filósofo coma es necesario que tenga los dos palillos en sus manos, entonces sus dos nodos deben tener un token y de esta manera poder comer.

7.4.3. Solución

Podemos concluir diciendo de que las Redes de Petri son una alternativa de modelado de sistemas, aplicados principalmente hacia el control y proceso, por su facilidad de manejo en el problema de la sincronización de procesos.

También se dijo que constan de cuatro partes:

- Nodos
- Transiciones
- Funciones de entrada
- Funciones de salida
- Las entradas y/o salidas de una transición son conjuntos que pueden tener elementos repetidos o múltiples ocurrencias.
- Cuentan con una asignación de tokens que es la parte dinámica de las Redes de Petri.

- Las Redes de Petri se pueden representar gráficamente, un círculo O representa un nodo y una barra representa una transición, y los tokens son representados por pequeños puntos · .
- Las Redes de Petri tienen reglas de disparo, siendo la principal, la que dice: "todos los nodos de entrada de la transición, deben tener al menos el mismo número de tokens, que número de arcos van hacia la transición para que ésta sea disparada". Cuando la transición cumple dicha condición se dice que es ENABLED.
- Existen extensiones a las Redes de Petri: por ejemplo, las Redes de Petri Coloreadas (PNC), las Redes de Petri Temporales, Redes de Petri Estocásticas.

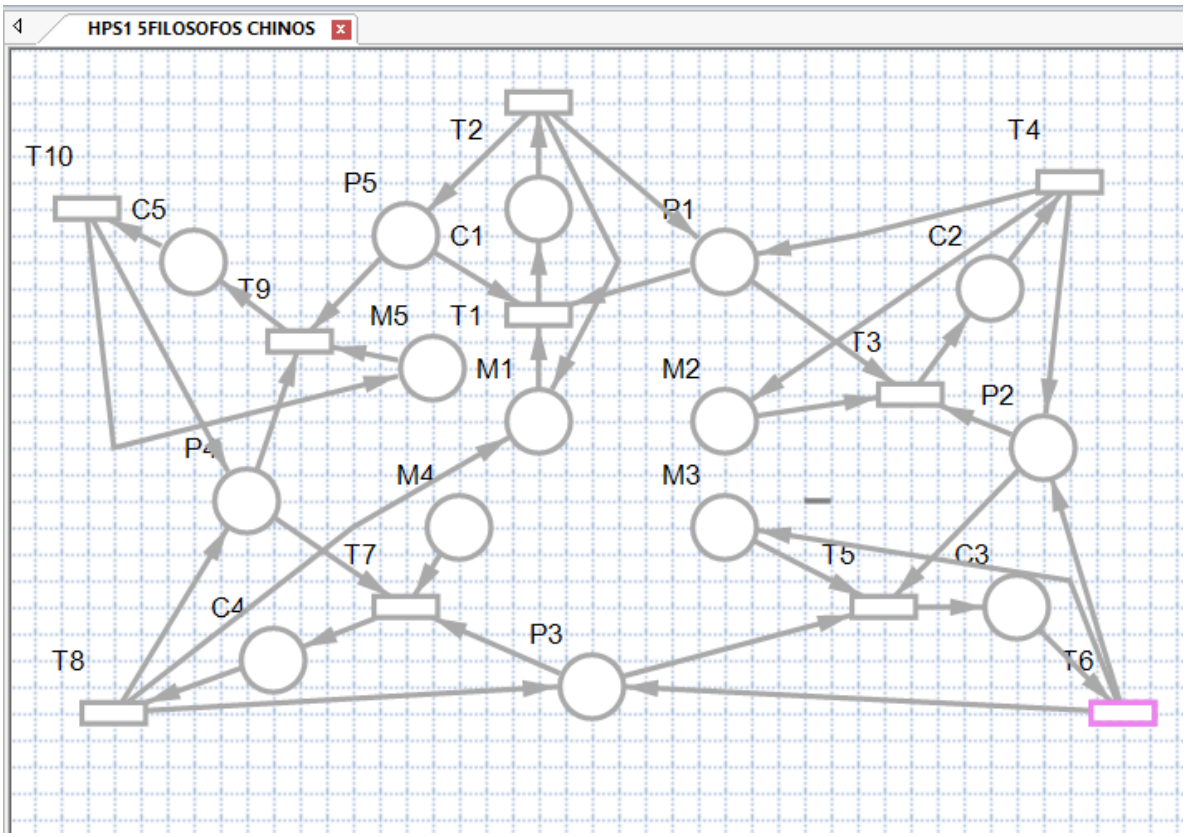

Ilustración 27: Diagrama de Petri, Orrala William

8. Link del Github

https://github.com/OrlandCede20/RedesPetriExpo2.git

9. Referencias

[1] F. Wittbold, R. Bernemann, R. Heckel, T. Heindel, and B. König, "Stochastic Decision Petri Nets," Mar. 2023.

[2] Z. E. Chay, B. F. Goh, and M. H. Ling, “PNet: A Python Library for Petri Net Modeling and Simulation,” Feb. 2023.

[3] M. Rawson and M. Rawson, “Petri Nets for Concurrent Programming,” Aug. 2022.

[4] M. Blondin and J. Esparza, “Separators in Continuous Petri Nets,” Sep. 2022.

[5] M. R. Besharati and A. S. Khameneh, “Modeling of Resource Allocation Mechanisms in Distributed Computing Systems using Petri Nets and Stochastic Activity Networks (SAN): a Review and Reo-based Suggestion,” Mar. 2021.

[6] J. Pablo, C. Villalobos, and G. Mejía, “REDES DE PETRI Y ALGORITMOS GENÉTICOS, UNA PROPUESTA PARA LA PROGRAMACIÓN DE SISTEMAS DE MANUFACTURA FLEXIBLE *.”

[7] G. Zapata, “ESTRUCTURAS GENERALIZADAS PARA CONTROLADORES LÓGICOS MODELADAS MEDIANTE REDES DE PETRI,” 2016. [Online]. Available: https://www.researchgate.net/publication/266276625

[8] M. de Oliveira Oliveira, “Synthesis and Analysis of Petri Nets from Causal Specifications,” 2022, pp. 447–467. doi: 10.1007/978-3-031-13188-2_22.

[9] M. Digital Scholar, A. G. Baratè Haus L A Ludovico Davide Andrea Mauro, A. Baratè, G. Haus, L. Ludovico, and D. Mauro, “Formalizing Schoenberg’s Fundamentals of Musical Composition through Petri Nets,” 2018. [Online]. Available: https://mds.marshall.edu/wdcs_faculty

[10] J. Kock, “Whole-grain Petri nets and processes,” May 2020, doi: 10.1145/3559103.

[11] M. Rawson and M. Rawson, “Petri Nets for Concurrent Programming,” Aug. 2022.

[12] W. Reisig, *Understanding Petri nets: Modeling techniques, analysis methods, case studies*, vol. 9783642332784. Springer-Verlag Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-33278-4.

[13] D. Palko *et al.*, "Cyber Security Risk Modeling in Distributed Information Systems," *Applied Sciences*, vol. 13, no. 4, p. 2393, Feb. 2023, doi: 10.3390/app13042393.

[14] J. Hao, Y. Yang, C. Xu, and X. Du, "A comprehensive review of planning, modeling, optimization, and control of distributed energy systems," *Carbon Neutrality*, vol. 1, no. 1, p. 28, Dec. 2022, doi: 10.1007/s43979-022-00029-1.

[15] Y. Zeng, B. Chen, P. Pan, K. Li, and G. Chen, "Modeling the Training Iteration Time for Heterogeneous Distributed Deep Learning Systems," *International Journal of Intelligent Systems*, vol. 2023, pp. 1–15, Feb. 2023, doi: 10.1155/2023/2663115.

[16] W. B. Daszczuk, "Modeling and Verification of Asynchronous Systems Using Timed Integrated Model of Distributed Systems," *Sensors*, vol. 22, no. 3, p. 1157, Feb. 2022, doi: 10.3390/s22031157.

[17] D. Xin and L. Shi, "Trajectory Modeling by Distributed Gaussian Processes in Multiagent Systems," *Sensors*, vol. 22, no. 20, p. 7887, Oct. 2022, doi: 10.3390/s22207887.

[18] D. Clavel, C. Mahulea, and M. Silva, "On Liveness Enforcement of Distributed Petri Net Systems," *IEEE Trans Automat Contr*, vol. 68, no. 6, pp. 3776–3782, Jun. 2023, doi: 10.1109/TAC.2022.3198328.

[19] O.-G. DUCA, E. MINCA, M.-A. PAUN, I. V. GURGU, O. E. DRAGOMIR, and C. BIDICA, "PETRI NET MODELING OF A PRODUCTION SYSTEM WITH PARALLEL MANUFACTURING PROCESSES," *Journal of Science and Arts*, vol. 23, no. 1, pp. 305–318, Mar. 2023, doi: 10.46939/J.Sci.Arts-23.1-c05.

[20] A. Heyduk, "Petri Networks for Mechanized Longwall System Simulation," 2020, pp. 307–318. doi: 10.1007/978-3-030-50936-1_27.

[21] J.-P. Signoret and A. Leroy, "Petri Net Modelling," 2021, pp. 587–660. doi: 10.1007/978-3-030-64708-7_33.

[22] M. Djahafi and N. Salmi, "Using Stochastic Petri net modeling for self-adapting Publish/Subscribe IoT systems," in *NOMS 2023-2023 IEEE/IFIP Network*

Operations and Management Symposium, IEEE, May 2023, pp. 1–4. doi: 10.1109/NOMS56928.2023.10154382.

[23] F. de Asís López Fuentes, *Sistemas distribuidos*. 2015. [Online]. Available: www.cua.uam.mx

[24] "Sistemas Distribuidos Módulo 1 Introducción a los Sistemas Distribuidos," 2020.

[25] "Aplicación de redes de petri en la modelación de sistemas de eventos discretos." Accessed: Feb. 17, 2024. [Online]. Available: https://uaeh.edu.mx/scige/boletin/icbi/n1/e4.html

[26] L. D. Ana María Huayna, M. Augusto Cortez Vásquez, and M. Hugo Vega Huerta, "Aplicación de las redes de Petri a la simulación discreta de sistemas."

[27] T. Piré, F. Bergero, and E. Kofman, "AADECA 2010-Semana del Control Automático-XXII • Congreso Argentino de Control Automático 31 de Agosto al 2 de Septiembre de," 2010.

[28] L. Wells, "Performance analysis using CPN tools," in *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools - valuetools '06*, New York, New York, USA: ACM Press, 2006, p. 59. doi: 10.1145/1190095.1190171.

[29] N. Kokash, C. Krause, and E. de Vink, "Reo + mCRL2 : A framework for model-checking dataflow in service compositions," *Formal Aspects of Computing*, vol. 24, no. 2, pp. 187–216, Mar. 2012, doi: 10.1007/s00165-011-0191-6.

[30] M. A. Riwanto and W. N. Budiarti, "Development of Digital Science Comics for Elementary School as a Support for Digital Literacy in Online Learning," in *Proceedings of the 4th International Conference on Learning Innovation and Quality Education*, New York, NY, USA: ACM, Sep. 2020, pp. 1–4. doi: 10.1145/3452144.3452221.

[31] A. Arcoverde, G. Alves, and R. Lima, "Petri nets tools integration through Eclipse," in *Proceedings of the 2005 OOPSLA workshop on Eclipse technology*

eXchange - eclipse '05, New York, New York, USA: ACM Press, 2005, pp. 90–94. doi: 10.1145/1117696.1117715.

[32] M. Jurikovič, P. Čičák, and K. Jelemenská, "Parallel controller design and synthesis," in *Proceedings of the 7th FPGAworld Conference*, New York, NY, USA: ACM, Sep. 2010, pp. 35–40. doi: 10.1145/1975482.1975486.

[33] J. D. F. de la Hoz, J. S. Sosa, and N. Cardozo, "Distributed Context Petri Nets," in *Proceedings of the Workshop on Context-oriented Programming - COP '19*, New York, New York, USA: ACM Press, 2019, pp. 24–31. doi: 10.1145/3340671.3343359.

[34] A. Bouillard and B. Gaujal, "Backward coupling in petri nets," in *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools - valuetools '06*, New York, New York, USA: ACM Press, 2006, p. 33. doi: 10.1145/1190095.1190137.

[35] E. Tavares, P. Maciel, B. Silva, and M. Oliveira, "A time petri net-based approach for hard real-time systems scheduling considering dynamic voltage scaling, overheads, precedence and exclusion relations," in *Proceedings of the 20th annual conference on Integrated circuits and systems design*, New York, NY, USA: ACM, Sep. 2007, pp. 312–317. doi: 10.1145/1284480.1284563.

[36] W. Qinghao and Y. Dengkai, "A New Method for Evaluating Air Traffic Control Safety," in *Proceedings of the 2017 VI International Conference on Network, Communication and Computing*, New York, NY, USA: ACM, Dec. 2017, pp. 217–221. doi: 10.1145/3171592.3171640.

[37] B. Farwer and M. Leuschel, "Model checking object petri nets in prolog," in *Proceedings of the 6th ACM SIGPLAN international conference on Principles and practice of declarative programming*, New York, NY, USA: ACM, Aug. 2004, pp. 20–31. doi: 10.1145/1013963.1013970.

[38] E. Gianniti, A. M. Rizzi, E. Barbierato, M. Gribaudo, and D. Ardagna, "Fluid Petri Nets for the Performance Evaluation of MapReduce and Spark Applications,"

ACM SIGMETRICS Performance Evaluation Review, vol. 44, no. 4, pp. 23–36, May 2017, doi: 10.1145/3092819.3092824.

[39] V. Lesi, Z. Jakovljevic, and M. Pajic, "Reliable industrial IoT-based distributed automation," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, New York, NY, USA: ACM, Apr. 2019, pp. 94–105. doi: 10.1145/3302505.3310072.

[40] M. D. Petty *et al.*, "Modeling cyberattacks with extended Petri nets," in *Proceedings of the ACM Southeast Conference*, New York, NY, USA: ACM, Apr. 2022, pp. 67–73. doi: 10.1145/3476883.3520209.

[41] S. Ebert, "Safe adaptation of cobotic cells based on petri nets," in *Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems*, New York, NY, USA: ACM, May 2022, pp. 43–47. doi: 10.1145/3524844.3528075.

[42] E. Caveness, P. S. G. C., Z. Peng, N. Polyzotis, S. Roy, and M. Zinkevich, "TensorFlow Data Validation: Data Analysis and Validation in Continuous ML Pipelines," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA: ACM, Jun. 2020, pp. 2793–2796. doi: 10.1145/3318464.3384707.

[43] J. Cederbladh, A. Cicchetti, and J. Suryadevara, "Early Validation and Verification of System Behaviour in Model-Based Systems Engineering: A Systematic Literature Review," *ACM Transactions on Software Engineering and Methodology*, Nov. 2023, doi: 10.1145/3631976.

[44] N. Akhtar *et al.*, "Hierarchical Coloured Petri-Net Based Multi-Agent System for Flood Monitoring, Prediction, and Rescue (FMPR)," *IEEE Access*, vol. 7, pp. 180544–180557, 2019, doi: 10.1109/ACCESS.2019.2958258.

[45] M. D. Petty *et al.*, "Modeling cyberattacks with extended Petri nets," in *Proceedings of the ACM Southeast Conference*, New York, NY, USA: ACM, Apr. 2022, pp. 67–73. doi: 10.1145/3476883.3520209.

[46] I. Sitova and J. Pecerska, "Process Mining Techniques in Simulation Model Adequacy Assessment," in *2019 60th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*, IEEE, Oct. 2019, pp. 1–4. doi: 10.1109/ITMS47855.2019.8940672.

[47] L. Weingarten, K. Datta, and R. Drechsler, "PolyMiR: Polynomial Formal Verification of the MicroRV32 Processor," in *Proceedings of the 18th ACM International Symposium on Nanoscale Architectures*, New York, NY, USA: ACM, Dec. 2023, pp. 1–6. doi: 10.1145/3611315.3633262.

[48] M. Schnieber and R. Drechsler, "Polynomial Formal Verification of KFDD Circuits," in *Proceedings of the 21st ACM-IEEE International Conference on Formal Methods and Models for System Design*, New York, NY, USA: ACM, Sep. 2023, pp. 82–89. doi: 10.1145/3610579.3611080.

[49] T. Hu, Y. Yao, and F. Zhu, "A CTL-based Parallel Discrete Event Simulation Model Verification Method," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, IEEE, Mar. 2019, pp. 2250–2255. doi: 10.1109/ITNEC.2019.8729515.

[50] M. M. Biswas, S. Razia Akhter, and K. C. Paul, "Power Quality Analysis for Distributed Generation and Electric Vehicle Integrated Distribution System," in *2020 55th International Universities Power Engineering Conference (UPEC)*, IEEE, Sep. 2020, pp. 1–6. doi: 10.1109/UPEC49904.2020.9209796.

[51] M. V. Liu, B. Yuan, Z. Wang, J. A. Sward, K. M. Zhang, and C. L. Anderson, "An Open Source Representation for the NYS Electric Grid to Support Power Grid and Market Transition Studies," *IEEE Transactions on Power Systems*, pp. 1–11, 2022, doi: 10.1109/TPWRS.2022.3200887.

[52] L. Jia, S. Pannala, and A. Srivastava, "Enhancing Resilience in Islanded Microgrids with PV-Battery using Bi-level MILP Optimization," in *2023 IEEE International Conference on Energy Technologies for Future Grids (ETFG)*, IEEE, Dec. 2023, pp. 1–6. doi: 10.1109/ETFG55873.2023.10407440.

[53] S. Rehman, H. U. R. Habib, S. Wang, M. S. Buker, L. M. Alhems, and H. Z. Al Garni, "Optimal Design and Model Predictive Control of Standalone HRES: A Real Case Study for Residential Demand Side Management," *IEEE Access*, vol. 8, pp. 29767–29814, 2020, doi: 10.1109/ACCESS.2020.2972302.