Universidad Técnica Estatal de Quevedo

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

Curso: Ingeniería en Software 7mo “A”.

Tema: Desarrollo de Aplicaciones en Ambientes Distribuidos.

Asignatura: Aplicaciones Distribuidas.

Docente: Guerrero Ulloa Gleiston Cicerón.

Índice

Ilustraciones

1. Modelamiento de aplicaciones distribuidas

1.1. Diagramas UML

1.1.1. Utilización de diagramas UML para modelar la estructura y el comportamiento de sistemas distribuidos

UML se fundamenta en un metamodelo de cuatro capas y presenta 14 tipos de diagramas, los cuales son útiles para representar las diversas perspectivas de un sistema (como estructura o comportamiento) y diferentes niveles de abstracción (como análisis o diseño). Este enfoque facilita la gestión de la complejidad en la especificación del sistema y la asignación de responsabilidades entre diversos interesados, entre otros beneficios. Aunque los diversos diagramas de UML abordan distintos aspectos de un sistema en desarrollo, no son independientes, sino que están estrechamente interconectados de varias maneras [1].

El UML cuenta con una variedad de diagramas que facilitan el análisis y diseño de software; por ejemplo, se pueden utilizar diagramas de clases, que representan la estructura estática de un sistema en términos de clases y las relaciones entre clases [2].

Los diagramas de casos de uso, que representan cómo interactúan los actores con el sistema, son ahora una parte integral de la metodología UML y ofrecen ventajas sobre la mera descripción textual de los requisitos funcionales [2]**.** Facilitan la

identificación de requisitos y son comprensibles tanto para clientes como para usuarios. Estos casos de uso no solo tienen un valor descriptivo, sino que también pueden ser utilizados como base para las pruebas del sistema y la documentación dirigida a los usuarios. La estructura de los casos de uso se enfoca en la descripción de secuencias de interacción entre el sistema y diversos actores, ya sean personas u otros sistemas. Se trata al sistema como una entidad cerrada, donde los actores obtienen resultados observables [3].

El uso generalizado del Lenguaje Unificado de Modelado (UML) como notación estándar ha sido adoptado en la construcción de sistemas distribuidos críticos en tiempo y recursos. En consecuencia, es necesario desarrollar técnicas de análisis predictivo basadas en modelos (MBPA), tales como el análisis de uso de recursos y la predicción de carga, para aprovechar los modelos UML desde las etapas iniciales del proceso de diseño. Algunas de las cuales hacen uso de modelos UML, especialmente en diagramas de actividad y de secuencia [4].

Dado que los sistemas distribuidos modernos se basan en objetos, ha habido un interés considerable en el uso del Lenguaje Unificado de Modelado (UML). De hecho, UML ha surgido rápidamente como la notación estándar para el análisis y diseño orientado a objetos. Sin embargo, debido a que intenta ofrecer notación para la mayoría de los aspectos del diseño orientado a objetos, los usuarios tienen a su disposición una amplia variedad de diagramas de diseño en UML y hay pocas pautas sobre cómo se define precisamente el diseño [5].

1.1.2. Diagramas de clases en contextos distribuidos

Los diagramas de clases mantienen su posición como la técnica de modelado más crucial y extensamente empleada en UML. En su origen histórico, estos diagramas surgieron de los conceptos de modelado entidad-relación y de la representación gráfica de módulos, influenciados a su vez por los diagramas de flujo de datos. En esencia, los diagramas de clases delinean la estructura de un sistema de software, estableciéndose, así como la notación central inicialmente abordada para el modelado orientado a objetos [6].

La modularidad desempeña un papel crucial en el desarrollo, implementación y cuidado de sistemas distribuidos. Su relevancia abarca tanto las etapas de diseño y mantenimiento del ciclo de vida del software como la funcionalidad final de un programa específico. En el caso del diseño del paquete RPC, también se hace uso del sistema de módulos parametrizados. La parametrización facilita la personalización del paquete durante el proceso de vinculación, permitiendo la elección de la representación externa, el mecanismo de transporte o la estrategia de evaluación mediante la conexión de los parámetros de módulo correspondientes [7].

En el desarrollo de sistemas operativos distribuidos, se busca implementar abstracciones comprensibles y eficientes para los programadores, garantizando la versatilidad en la incorporación de funcionalidades y propiedades no funcionales. Esto incluye la manipulación de archivos, gestión de calendarios, acceso transparente a través de redes y almacenamiento en caché. Se equilibra la transparencia, al aliviar al programador de la complejidad de lidiar con redes, con el control, permitiendo la selección del protocolo más eficiente y escalable según las necesidades específicas [8].

1.1.3. Diagramas de secuencia en contextos distribuidos

Los diagramas de secuencia son empleados para representar las interacciones entre objetos, detallando la secuencia en que se ejecutan las llamadas a métodos y cuándo finalizan. En el ámbito de la ingeniería de requisitos, los desarrolladores enfrentan el desafío de derivar una implementación completa a partir de un conjunto finito de diagramas de secuencia. Diversas propuestas han surgido con el propósito de ampliar los diagramas de secuencia y estructurar esta derivación, incorporando frecuentemente estructuras de control que posibilitan alternativas, paralelismo, iteraciones y llamadas recursivas en dichos diagramas. Estas propuestas se aplican de distintas maneras en los dos estándares fundamentales para los diagramas de secuencia [9].

Los Diagramas de Secuencia (SD) se utilizan para enfrentar la complejidad en la modelización de sistemas distribuidos. Se detallan los distintos pasos que llevan a

la verificación del refinamiento de los SD, tanto teórica como prácticamente. Se establece de manera formal una relación de refinamiento que cumple con las propiedades necesarias [10].

En la configuración de sistemas de control, los diagramas de secuencia, que pertenecen al subgrupo de los diagramas de comportamiento, tienen un papel fundamental al proporcionar una representación clara del funcionamiento de una aplicación de control en distintos escenarios [11].

En el desarrollo distribuido de sistemas orientados a objetos, los casos de uso específicos de una región interactúan con casos de uso globales o comunes para atender a requisitos comerciales particulares de esa región en específico. De este modo, un escenario involucra tanto componentes regionales como globales, generando una significativa intercalación entre los diagramas de secuencia que representan el flujo de eventos de los casos de uso. Comprender cómo se intercalan estos diagramas de secuencia es crucial, ya que proporcionará una perspectiva general de los escenarios del sistema completo. Esta comprensión resulta fundamental para diseñar casos de prueba de manera efectiva y garantizar una cobertura adecuada de los escenarios [12].

1.1.4. Diagramas de actividad en contextos distribuidos

Un diagrama de actividad representa la secuencia de actividades en un sistema, donde una actividad es una ejecución no atómica en progreso que, en última instancia, resulta en alguna acción. Esta acción puede incluir la llamada a otra operación, el envío de una señal, o la creación y destrucción de objetos. Gráficamente, el diagrama de actividad se compone de vértices y arcos que ilustran la relación entre estas actividades [13].

El diagrama de actividad UML es especialmente adecuado para representar procesos de software, derivando conceptos clave de diagramas de flujo, grafos de transición de estado y redes de Petri [14], [15]. Estos diagramas están compuestos por nodos y arcos, donde los nodos representan procesos o control de procesos, incluyendo estados de acción, estados de actividad, decisiones, carriles, bifurcaciones, uniones y objetos. La visualización del flujo de control entre diversas

actividades, junto con la representación del flujo de valores de objetos, es un aspecto destacado de estos diagramas. Los estados de flujo de objetos ilustran objetos que pueden fungir como entrada o salida de una actividad. En resumen, estos diagramas son herramientas valiosas para modelar tanto el flujo de trabajo en sistemas comerciales como el comportamiento complejo de operaciones [15].

El uso del diagrama de actividad UML en la generación de casos de prueba ha sido abordado por investigadores. No obstante, la tarea de encontrar un conjunto de casos de prueba a partir de un diagrama de actividad es ardua, principalmente debido a la presencia de bucles y actividades concurrentes, lo que conduce a una proliferación de caminos y, en la práctica, resulta poco factible considerar la totalidad de los caminos de ejecución para las pruebas [16].

1.2. Diagramas BPMN

1.2.1. Elementos principales de BPMN

En BPMN, los elementos se agrupan en cuatro categorías principales que son fundamentales para la creación de cualquier diagrama de proceso de negocio [17]. Los Objetos de Flujo abarcan tareas, eventos y puertas de enlace. Las tareas representan acciones individuales dentro de un proceso, mientras que los eventos indican sucesos como el inicio o fin de un proceso, pudiendo clasificarse según su posición en inicio, intermedio o fin. Las puertas de enlace guían la divergencia y convergencia del flujo, utilizando condiciones lógicas y permitiendo múltiples trayectos dentro del proceso [18], [19].

Por otro lado, los Objetos de Conexión son los elementos que enlazan los objetos de flujo. La secuencia de flujos determina el orden de las actividades, mientras que los mensajes y las asociaciones representan la interacción entre diferentes partes del proceso o con elementos externos. Estos componentes son esenciales para establecer la secuencia y la comunicación dentro del proceso [18].

Por último, las Piscinas y Carriles son estructuras que organizan las tareas y eventos. Una piscina representa a un participante completo en el proceso, mientras que los carriles dividen esta piscina en roles o responsabilidades específicas. Estas

estructuras facilitan la visualización de la interacción entre diferentes entidades o departamentos involucrados en el proceso [19].

1.2.2. Ventajas del modelamiento con BPMN en sistemas distribuidos.

La aplicación de BPMN en sistemas distribuidos sobresale por su capacidad para proporcionar una representación visual clara de los procesos complejos. Esta notación facilita una comunicación efectiva entre todos los involucrados en el proyecto, desde analistas de negocio hasta desarrolladores y gerentes. La representación visual intuitiva de las interacciones entre los componentes distribuidos facilita la comprensión compartida de los procesos y fomenta la colaboración [20].

Al ser un estándar de la industria bien establecido, BPMN promueve la interoperabilidad entre diversas herramientas y plataformas. Esto previene la dependencia de un proveedor específico y permite que las organizaciones adopten las mejores prácticas de modelado de procesos. Por lo tanto, la flexibilidad y la adaptabilidad son ventajas significativas al utilizar BPMN en el contexto de sistemas distribuidos [21].

Además, BPMN desempeña un papel crucial en la automatización de procesos al permitir la implementación de flujos de trabajo automatizados que operan eficientemente a través de diferentes sistemas y servicios. Esta capacidad es especialmente valiosa para gestionar cambios, ya que los diagramas BPMN detallados pueden ayudar a identificar rápidamente áreas para adaptación o mejora [22].

1.2.3. Integración de BPMN con tecnologías de sistemas distribuidos

La integración de modelos BPMN con sistemas distribuidos es fundamental para la ejecución eficiente de los procesos de negocio. Esta integración se facilita mediante el uso de tecnologías como middleware de integración, servicios web, API REST, y protocolos de mensajería. Estas tecnologías permiten la comunicación entre distintos componentes y servicios, asegurando que los procesos modelados funcionen sin problemas a través de diversas plataformas y sistemas [23], [24].

Las herramientas de BPM y BPMN suelen ofrecer capacidades de integración o extensiones que permiten conectar los modelos de procesos con aplicaciones externas. Esto incluye sistemas como ERP, CRM y otras bases de datos y aplicaciones empresariales. La integración efectiva es crucial para automatizar los procesos de negocio modelados, permitiendo una operación fluida y coherente entre distintos sistemas [25].

1.3. Redes de Petri

1.3.1. Aplicación de Redes de Petri para modelar sistemas concurrentes y distribuidos.

La teoría de las redes de Petri ofrece un enfoque versátil y poderoso para el modelado de sistemas concurrentes y distribuidos, proporcionando un marco formal tanto gráfico como matemático para la representación y análisis de procesos que ocurren simultáneamente. Esta teoría se basa en la idea de que los sistemas complejos, especialmente aquellos que operan con múltiples procesos en paralelo, pueden ser modelados de manera efectiva para comprender su comportamiento, optimizar su rendimiento y verificar su corrección [26].

- Aplicaciones en sistemas concurrentes

En el contexto de sistemas concurrentes, las redes de Petri se utilizan como un formalismo para describir y analizar la interacción entre procesos que se ejecutan al mismo tiempo. Esto es particularmente útil para el diseño y análisis de sistemas informáticos y de comunicaciones, donde la concurrencia es una característica inherente [26]. La capacidad de las redes de Petri para modelar explícitamente la simultaneidad y la sincronización entre eventos hace que sean excepcionalmente adecuadas para construir, optimizar y probar software a nivel de concurrencia. Esto permite a los desarrolladores detectar y corregir posibles errores de sincronización y condiciones de carrera antes de que el sistema sea implementado, aumentando la fiabilidad y eficiencia del software [27].

- Modelado en sistemas distribuidos

Las redes de Petri también se extienden al modelado de sistemas distribuidos, como los sistemas ciberfísicos distribuidos (DCPS). En estos sistemas, las redes de Petri facilitan la representación de la interacción y la comunicación entre componentes distribuidos, modelando los procesos de envío y recepción a través de canales de comunicación [26]. Esto permite diseñar sistemas distribuidos con una comprensión clara de cómo los componentes interactúan y se sincronizan entre sí, lo cual es crucial para garantizar la coherencia y la fiabilidad del sistema en su conjunto [27].

- Procesos secuenciales sincronizados en sistemas distribuidos

En el ámbito de los sistemas distribuidos, las redes de Petri se emplean para modelar procesos secuenciales sincronizados (SSP), donde se utiliza una red de Petri de control para coordinar la ejecución de los procesos y evitar el bloqueo o la inactividad [27]. Esta aplicación es especialmente importante en sistemas donde la correcta secuencia y sincronización de operaciones son críticas para el rendimiento y la correcta funcionalidad del sistema.

- Extensiones de redes de Petri

Para abordar la complejidad y dinamismo de los sistemas modernos, se han desarrollado extensiones de las redes de Petri, como las redes de Petri marcadas y aumentadas dinámicamente [26]. Estas extensiones permiten modelar el comportamiento dinámico y las interacciones simultáneas de manera más efectiva, ofreciendo una mayor flexibilidad y potencia en el modelado de sistemas que cambian con el tiempo o que requieren una gestión compleja de estados y transiciones [27].

1.3.2. Representación de transiciones, estados y sincronización en sistemas distribuidos.

La representación de transiciones, estados, y sincronización en sistemas distribuidos puede ser compleja debido a la naturaleza concurrente y distribuida de estos sistemas. Sin embargo, las redes de Petri ofrecen un marco poderoso y flexible para abordar estos desafíos. A continuación, se describe

cómo las redes de Petri pueden ser utilizadas para representar estos aspectos clave de sistemas distribuidos:

- Estados

En las redes de Petri, los estados de un sistema se representan mediante una estructura compuesta por "plazas" (también conocidas como "lugares") y "marcas". Cada plaza en la red simboliza una condición, situación o recurso específico dentro del sistema modelado, y las marcas dentro de estas plazas representan la existencia o cantidad de esa condición, situación o recurso [26], [27].

Por ejemplo, en el modelado de un sistema distribuido, una plaza podría representar un recurso como la memoria disponible en un sistema de computación, mientras que las marcas en esa plaza indicarían cuánta memoria está actualmente libre para ser asignada. Del mismo modo, en un sistema de gestión de procesos, las plazas podrían representar distintos estados de un proceso (por ejemplo, "en espera", "en ejecución", "completado"), y las marcas en esas plazas indicarían cuántos procesos se encuentran en cada uno de esos estados [26], [27].

La "marca" o "marcado" de una red de Petri, que es la distribución específica de marcas a través de sus plazas, captura el estado global del sistema modelado en un instante de tiempo. Esta configuración permite describir no solo la situación actual del sistema sino también facilita el análisis de su comportamiento dinámico a lo largo del tiempo [28]. A medida que el sistema evoluciona, las transiciones dentro de la red se disparan, lo que lleva a cambios en la distribución de marcas. Este cambio en el marcado refleja una transformación en el estado del sistema, permitiendo modelar y analizar la progresión de eventos y la interacción entre diferentes componentes del sistema [28].

Además, la representación de estados mediante plazas y marcas en las redes de Petri posibilita la modelización de sistemas con un alto grado de concurrencia y paralelismo. Dado que varias transiciones pueden dispararse independientemente si las condiciones (es decir, la disponibilidad de marcas en

las plazas de entrada) son adecuadas, las redes de Petri son excepcionalmente buenas para capturar la complejidad inherente a los sistemas distribuidos y concurrentes [27], [28].

La flexibilidad y expresividad de las redes de Petri las hacen una herramienta valiosa para el diseño, análisis y optimización de sistemas distribuidos. Al proporcionar una representación gráfica y matemática de los estados y transiciones del sistema, facilitan la identificación de posibles ineficiencias, puntos muertos o conflictos de recursos, permitiendo a los diseñadores y analistas tomar decisiones informadas para mejorar el rendimiento y la confiabilidad del sistema [27].

- Transiciones

Las transiciones en las redes de Petri desempeñan un papel crucial al modelar la dinámica de los sistemas, representando los eventos o acciones que provocan cambios de estado dentro del sistema. Visualmente, una transición se muestra como una barra o rectángulo situado entre las plazas, sirviendo como el nexo que transforma la configuración de marcas (y por tanto, el estado del sistema) de una manera específica y predefinida [26].

El mecanismo por el cual las transiciones se activan, conocido como "disparo", es fundamental para entender cómo las redes de Petri simulan la evolución de los sistemas. Una transición se considera "habilitada" y lista para dispararse cuando todas las plazas de entrada poseen un número de marcas igual o superior al requerido por la transición. Este requisito refleja las condiciones previas necesarias para que un evento ocurra en el sistema real que está siendo modelado [27], [28].

Cuando una transición se dispara, consume un determinado número de marcas de sus plazas de entrada, reflejando el uso o transformación de recursos o condiciones representadas por esas plazas. Simultáneamente, la transición produce marcas en sus plazas de salida, lo que simboliza la generación de nuevos estados, resultados o recursos como consecuencia del evento modelado [28].

- Sincronización

La sincronización en sistemas distribuidos es esencial para garantizar que múltiples procesos o tareas que se ejecutan de manera concurrente operen de forma coherente y eficiente, sin interferir negativamente unos con otros. Las redes de Petri, con su estructura formal matemática y gráfica, ofrecen un marco robusto para modelar y analizar estos mecanismos de sincronización, asegurando la correcta secuencia de operaciones y la coexistencia armoniosa de actividades paralelas [29].

En el corazón de la sincronización en las redes de Petri está el concepto del disparo de transiciones. Una transición en una red de Petri puede dispararse sólo cuando sus condiciones previas, representadas por la presencia de marcas en sus plazas de entrada, se satisfacen. Esto implica que el estado actual del sistema (la distribución de marcas en las plazas) debe ser tal que permita el disparo de la transición [29].

Las redes de Petri permiten modelar sistemas complejos donde la sincronización no sólo ocurre dentro de un componente del sistema sino también entre múltiples componentes distribuidos. Esto se logra mediante "transiciones sincronizadas", donde transiciones en diferentes partes del sistema deben dispararse juntas, reflejando una operación coordinada entre componentes [29]. Las redes de Petri coloreadas extienden esta capacidad, permitiendo una mayor granularidad y flexibilidad en la modelización de sincronizaciones complejas y la comunicación entre componentes. Por medio de diferentes colores (categorías) de marcas, estas redes pueden representar distintos tipos de mensajes o señales, facilitando la representación de protocolos de comunicación y sincronización en sistemas distribuidos [29], [30].

2. Desarrollo de aplicaciones Basados en IoT

2.1. Metodología de desarrollo

2.1.1. Selección y aplicación de metodologías de desarrollos para proyectos basados en IoT.

La importancia de las directrices de formación en el desarrollo de aplicaciones de IoT. Se centra en comprender la cadena de valor de IoT y los componentes centrales de un laboratorio de IoT. El objetivo es dotar a los participantes de habilidades para diversos roles en el Internet de las cosas. Además, oportunidades para que los proveedores de educación y las instituciones educativas ofrezcan una variedad de cursos de IoT que van desde aplicaciones prácticas hasta aplicaciones avanzadas [31]**.**

La conexión entre el aprendizaje automático (ML) y el Internet de las cosas (IoT) y cómo el ML puede impulsar el IoT al proporcionar inteligencia. Además, las aplicaciones de IoT utilizan cada vez más datos de sensores en modelos de ML para mejorar sus procesos y las complejidades de integrar canales de aprendizaje automático en el ciclo de vida de desarrollo de aplicaciones de IoT [31]**.**

Este es por ejemplo un middleware diseñado para simplificar el desarrollo de aplicaciones de IoT personalizadas. Utiliza la lógica modular de Prolog para comunicarse con el sistema IoT y tiene capacidades sofisticadas de consulta y razonamiento. Además, proporciona funcionalidad integrada para la gestión automática de dispositivos y conexiones, control de acceso, automatización avanzada y módulos de abstracción para separar aplicaciones de la infraestructura subyacente [31]**.**

Este es un ejemplo sobre selección de celdas en redes NB-IoT, una tecnología de bajo consumo energético y amplia aplicación. Las mediciones se realizaron en 30 sitios con diferentes escenarios de aplicación NB-IoT que involucran dos tipos de módulos y operadores de red. Se identifican cuatro desafíos en la selección de celdas y se propone un enfoque adaptativo para optimizarla, demostrando mejoras en cobertura y eficiencia energética a través de pruebas de simulación [31], [32]**.**

2.1.2. Consideración de aspectos como la adquisición de datos, el procesamiento en la nube y la interconexión de dispositivos.

La importancia de fortalecer la resiliencia de la infraestructura de la nube debido a su adopción generalizada y dependencia social. Se hace hincapié en un estudio detallado de las técnicas de resiliencia utilizando modelos en capas para comprender mejor las estrategias empleadas. Además, el modelo conocido como Resilinet también se utiliza para clasificar las investigaciones existentes en esta área. La importancia del análisis de big data en la sociedad actual y la necesidad de considerar los costos asociados, especialmente aquellos relacionados con las compras de hardware y energía, se analiza la relación entre el rendimiento del sistema informático, la asignación de recursos de hardware y el consumo de energía, y proporciona un sistema para equilibrar estos aspectos. Estas estrategias pueden reducir el costo general del análisis de big data en la nube y, al mismo tiempo, cumplir con los objetivos de carga de trabajo [31], [32].

2.2. Alternativas de solución

2.2.1. Exploración de alternativas de solución para la implementación de sistemas IoT.

En este ejemplo se destaca desafíos en WSNs e IoTs debido a limitaciones de recursos. Se propone un método de particionamiento para distribuir la carga de procesamiento entre nodos, generando alternativas de implementación y recomendaciones para minimizar complejidad. especialmente en la asignación de tareas a nodos para sistemas a gran escala. Se valida su eficacia en identificar soluciones arquitectónicas desfavorables, permitiendo una exploración más eficiente de nodos sensores [33].

Este es un ejemplo para implementar CNN en dispositivos IoT de borde, enfatizando sus ventajas sobre el procesamiento en la nube. Utiliza la exploración del diseño para optimizar el rendimiento y la eficiencia energética, lo que demuestra que la asignación máxima de recursos no siempre es óptima. La gestión de recursos basada en marcos reduce el tiempo de ejecución y el consumo de energía en comparación con los métodos directos [33]

Este ejemplo destaca los avances en los sistemas ciberfísicos y el Internet de las cosas están transformando la atención médica en dispositivos integrados y portátiles con un enfoque en la eficiencia energética. Análisis detallado de perfiles de potencia, rendimiento y energía para software de análisis de ECG y detección de arritmias en sistemas SoC basados en ZYNQ. La investigación de diseño abarca desde la implementación pura de software hasta el diseño de hardware/software, lo que sugiere una investigación exhaustiva de alternativas para optimizar los sistemas de atención médica en dispositivos IoT [33], [34].

2.2.2. Uso de plataformas y tecnologías como Arduino, Raspberry Pi, y servicios en la nube para IoT

Este enfoque destaca la creciente integración de sensores y actuadores en el mundo físico que son útiles para la toma de decisiones. Analiza el uso creciente de IoT en soluciones residenciales y destaca la necesidad de estandarizar la arquitectura de datos. Se destaca la importancia de desarrollar estándares antes que los requisitos regulatorios y la investigación inicial sobre arquitecturas colaborativas de IoT para ciudades inteligentes. Este enfoque es sobre el impacto en el consumo de energía de una computadora nueva, la Raspberry Pi, en comparación con otros dispositivos como computadoras de escritorio, portátiles, tabletas y teléfonos inteligentes. Desde su lanzamiento en 2012, la adopción de Raspberry Pi ha crecido rápidamente, junto con las preocupaciones sobre el consumo de energía de las tecnologías de la información y las comunicaciones (TIC) [35].

3. Conclusión

En conclusión, el uso de herramientas de modelado como diagramas UML, BPMN, y redes de Petri en el diseño y desarrollo de aplicaciones distribuidas, ejemplificado por el sistema de estacionamiento automatizado, demuestra su valor crítico. Estas herramientas no solo facilitan una comprensión profunda y detallada de la estructura y dinámica de los sistemas, sino que también permiten abordar eficientemente los desafíos inherentes a la concurrencia, la sincronización y la distribución. Su aplicación mejora significativamente la capacidad para diseñar,

analizar y optimizar sistemas complejos, asegurando soluciones robustas, eficientes y escalables. Este enfoque integral es fundamental para el éxito en el desarrollo de sistemas distribuidos y IoT, contribuyendo a soluciones innovadoras que responden a problemas reales con eficacia. La investigación de diseño abarca desde la implementación pura de software hasta el diseño de hardware/software, sugiriendo una investigación exhaustiva de alternativas para optimizar los sistemas de atención médica en dispositivos IoT.

4. Practica

4.1. Practica Redes de Petri con sistemas distribuidos

4.1.1. Problema

Sistema de estacionamiento automatizado.

4.1.2. Descripción

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.

4.1.3. Proceso

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

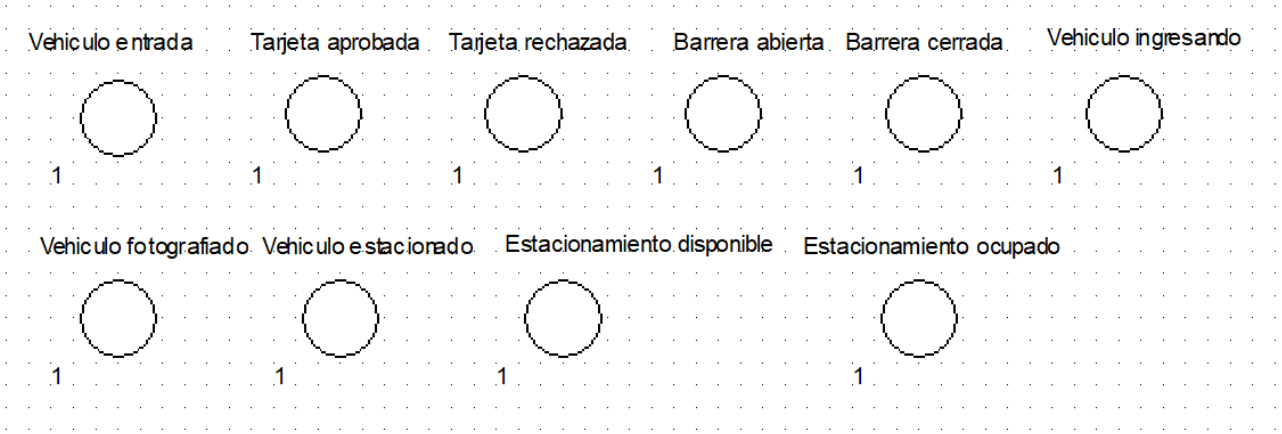

Ilustración 1.Plazas o Lugares utilizados

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

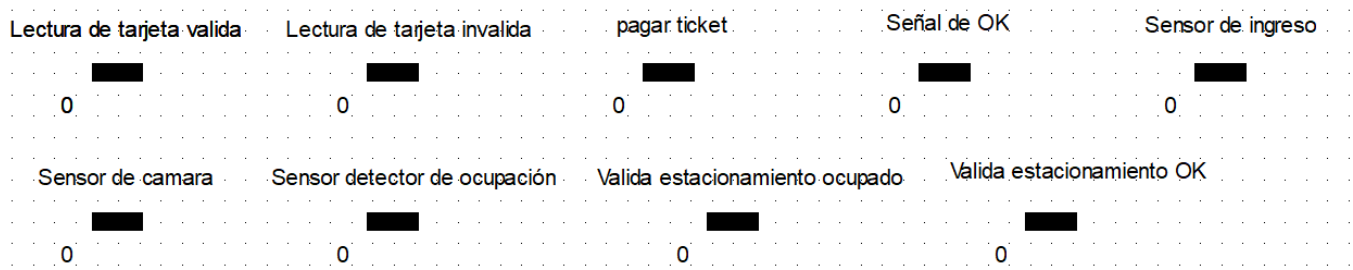

Ilustración 2. Transiciones que realiza la practica

3. Realizamos las conexiones mediante el uso de arcos

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

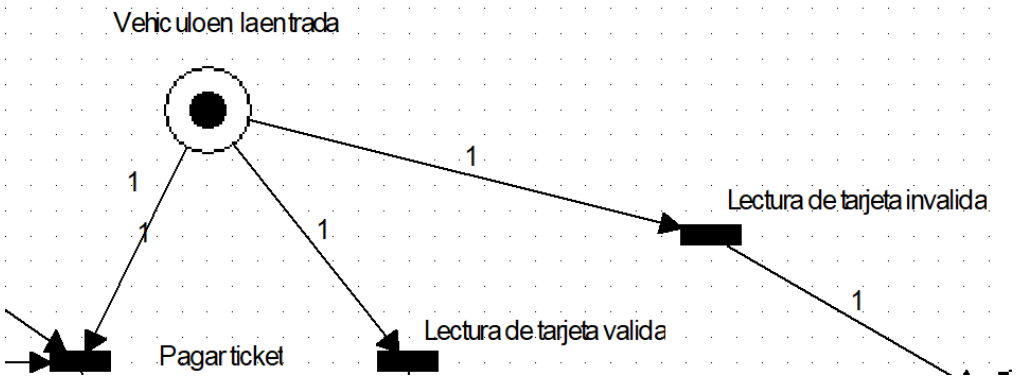

Ilustración 3. Conexión entre plaza y transiciones iniciales

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

2. 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.

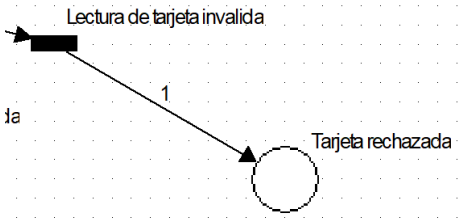

Ilustración 4. Conexión entre la transición "Lectura de tarjeta invalida" con la plaza "Tarjeta rechazada"

3. 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”

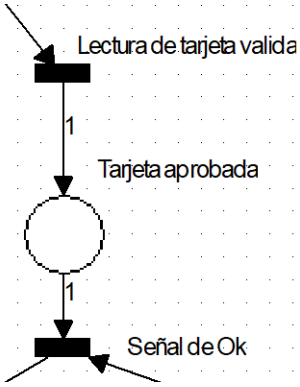

Ilustración 5. Conexión secuencial entre la transición "Lectura de tarjeta valida", la plaza "Tarjeta aprobada" y la transición "Señal de OK"

4. 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”

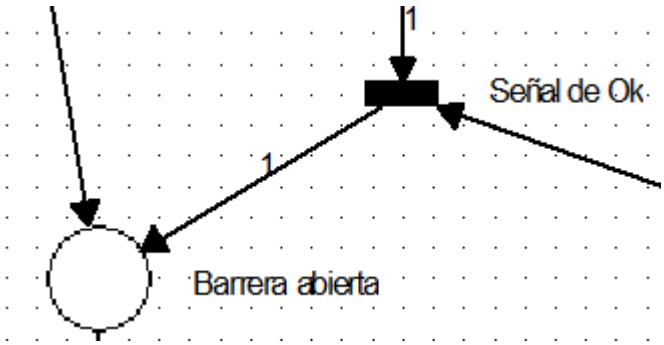

Ilustración 6. Unión en la transición "Señal de OK" hacia la plaza "Barrera abierta"

5. 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.

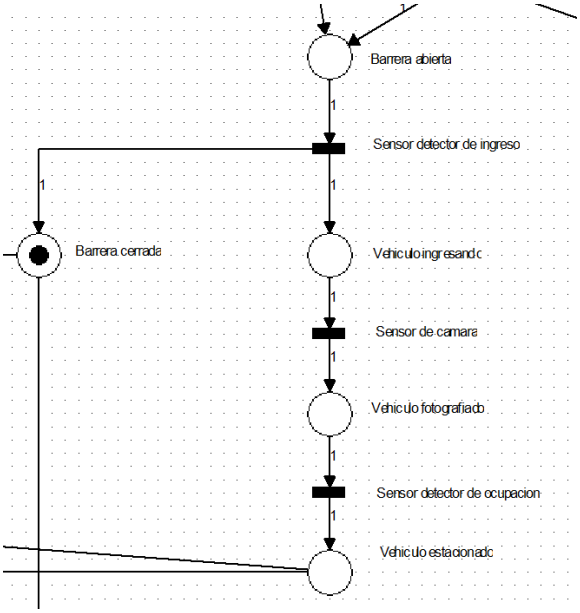

Ilustración 7. Proceso de ingreso al parqueadero.

6. 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".

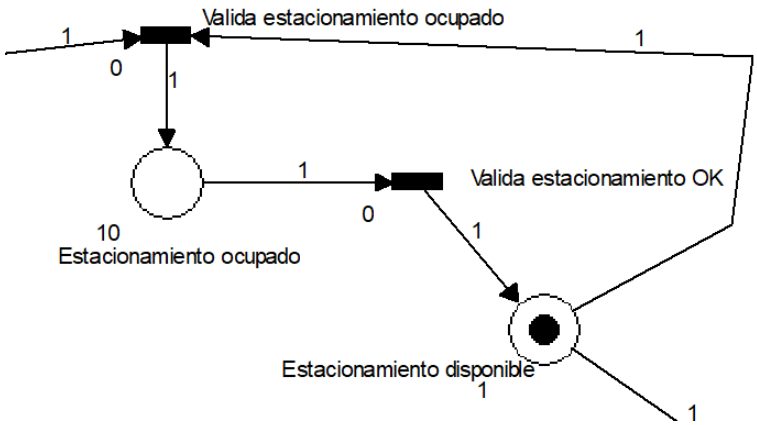

Ilustración 8. Validación del espacio de estacionamiento

7. 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.

4.1.4. Resultado

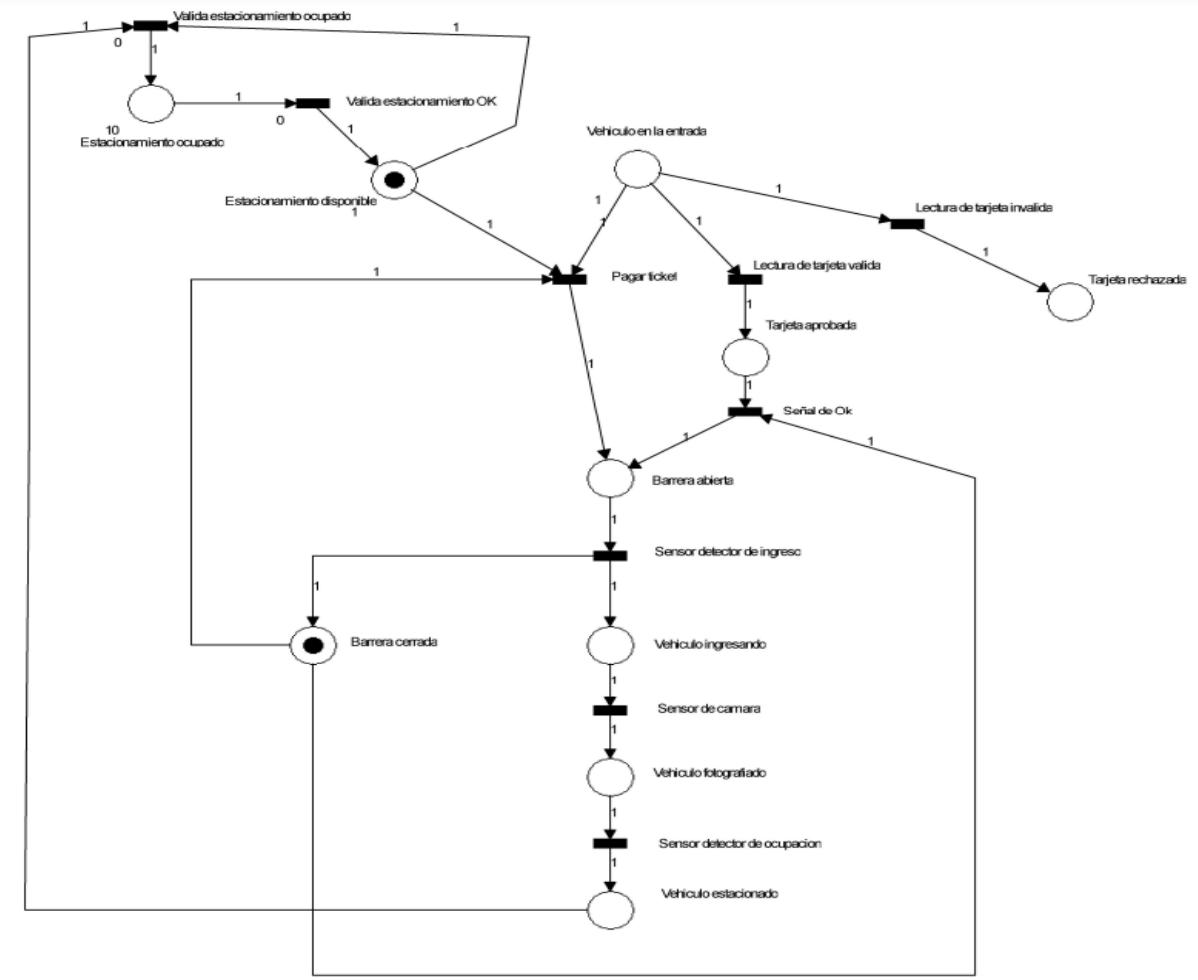

Ilustración 9. Diagrama de Petri desarrollado en base a la practica

5. Link del Github

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

6. Referencias

[1] D. Torre, Y. Labiche, M. Genero, M. T. Baldassarre, and M. Elaasar, "UML diagram synthesis techniques," in *Proceedings of the 10th International Workshop on Modelling in Software Engineering*, New York, NY, USA: ACM, May 2018, pp. 33–40. doi: 10.1145/3193954.3193957.

[2] A. Lopes, I. Steinmacher, and T. Conte, "UML Acceptance," in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, New York, NY, USA: ACM, Sep. 2019, pp. 264–272. doi: 10.1145/3350768.3352575.

[3] J. Corral-García, D. Cortés-Polo, C. Gómez-Martín, and J.-L. González-Sánchez, "Methodology and framework for the development of scientific applications with high-performance computing through web services," in *Proceedings of the 6th Euro American Conference on Telematics and Information Systems*, New York, NY, USA: ACM, May 2012, pp. 173–180. doi: 10.1145/2261605.2261631.

[4] V. Garousi, L. C. Briand, and Y. Labiche, "A UML-based quantitative framework for early prediction of resource usage and load in distributed real-time systems," *Softw Syst Model*, vol. 8, no. 2, pp. 275–302, Apr. 2009, doi: 10.1007/s10270-008-0099-7.

[5] B. Bordbar, J. Derrick, and G. Waters, "A UML Approach to the Design of Open Distributed Systems," 2002, pp. 561–572. doi: 10.1007/3-540-36103-0_56.

[6] B. Rumpe, "Class Diagrams," in *Modeling with UML*, Cham: Springer International Publishing, 2016, pp. 13–35. doi: 10.1007/978-3-319-33933-7_2.

[7] E. H. Siegel, "Applying high-level language paradigms to distributed systems," in *Proceedings of the 5th workshop on ACM SIGOPS European workshop Models and paradigms for distributed systems structuring - EW 5*, New York, New York, USA: ACM Press, 1992, p. 1. doi: 10.1145/506378.506425.

[8] D. Croce, M. Mellia, and E. Leonardi, "The quest for bandwidth estimation techniques for large-scale distributed systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 3, pp. 20–25, Jan. 2010, doi: 10.1145/1710115.1710120.

[9] B. Rumpe, "Sequence Diagrams," in *Modeling with UML*, Cham: Springer International Publishing, 2016, pp. 191–208. doi: 10.1007/978-3-319-33933-7_6.

[10] F. Dhaou, I. Mouakher, C. Attiogbé, and K. Bsaies, "Refinement of UML2.0 Sequence Diagrams for Distributed Systems," in *Proceedings of the 11th International Joint Conference on Software Technologies*, SCITEPRESS - Science and Technology Publications, 2016, pp. 310–318. doi: 10.5220/0006005403100318.

[11] D. Choinski, P. Skupin, and P. Krauze, "Application of the Sequence Diagrams in the Design of Distributed Control System," 2015, pp. 193–196. doi: 10.1007/978-3-319-24132-6_23.

[12] A. Kanjilal, G. Kanjilal, and S. Bhattacharya, "Integration of Design in Distributed Development Using D-Scenario Graph," in *2008 IEEE International Conference on Global Software Engineering*, IEEE, Aug. 2008, pp. 141–150. doi: 10.1109/ICGSE.2008.40.

[13] A. Tripathy and A. Mitra, "Test Case Generation Using Activity Diagram and Sequence Diagram," 2013, pp. 121–129. doi: 10.1007/978-81-322-0740-5_16.

[14] Y. D. C. Liang, Y. Wang, and Y. Liu, "The formal semantics of an UML activity diagram," *Journal of Shanghai University (English Edition)*, vol. 8, no. 3, pp. 322–327, Sep. 2004, doi: 10.1007/s11741-004-0072-9.

[15] L. Yu, X. Tang, L. Wang, and X. Li, "Simulating software behavior based on UML activity diagram," in *Proceedings of the 5th Asia-Pacific Symposium on Internetware*, New York, NY, USA: ACM, Oct. 2013, pp. 1–4. doi: 10.1145/2532443.2532465.

[16] W. Thanakorncharuwit, S. Kamonsantiroj, and L. Pipanmaekaporn, "Generating Test Cases from UML Activity Diagram Based on Business Flow Constraints," in *Proceedings of the Fifth International Conference on Network, Communication and Computing*, New York, NY, USA: ACM, Dec. 2016, pp. 155–160. doi: 10.1145/3033288.3033311.

[17] G. Salaun, "Quantifying the Similarity of BPMN Processes," in *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, Dec. 2022, pp. 377–386. doi: 10.1109/APSEC57359.2022.00050.

[18] H. Cheng, G. Kang, J. Liu, Y. Wen, B. Cao, and Z. Wang, "BPMN++: Comprehensive Business Process Modeling for Industrial Internet Application," in *2022 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, IEEE, Dec. 2022, pp. 548–555. doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom57177.2022.00076.

[19] I. N. Fatimah *et al.*, "USESPEC to BPMN: Web generator program for use case specification to BPMN," 2023, p. 040008. doi: 10.1063/5.0103694.

[20] M. Zhang, H. Li, Z. Huang, Y. Huang, and F. Bao, "Analyzing Realizability of BPMN Choreographies," in *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, IEEE, Sep. 2022, pp. 1–6. doi: 10.1109/DASC/PiCom/CBDCom/Cy55231.2022.9927990.

[21] I. N. Fatimah *et al.*, "USESPEC to BPMN: Web generator program for use case specification to BPMN," 2023, p. 040008. doi: 10.1063/5.0103694.

[22] Y. Falcone, G. Salaün, and A. Zuo, "Probabilistic Model Checking of BPMN Processes at Runtime," 2022, pp. 191–208. doi: 10.1007/978-3-031-07727-2_11.

[23] Y. Kirikkayis, F. Gallik, and M. Reichert, "Towards a Comprehensive BPMN Extension for Modeling IoT-Aware Processes in Business Process Models," 2022, pp. 711–718. doi: 10.1007/978-3-031-05760-1_47.

[24] E. Díaz, J. I. Panach, S. Rueda, and J. Vanderdonckt, "An empirical study of rules for mapping BPMN models to graphical user interfaces," *Multimed Tools Appl*, vol. 80, no. 7, pp. 9813–9848, Mar. 2021, doi: 10.1007/s11042-020-09651-6.

[25] P. Bocciarelli, A. D'Ambrogio, A. Giglio, and E. Paglia, "BPMN-Based Business Process Modeling and Simulation," in *2019 Winter Simulation Conference (WSC)*, IEEE, Dec. 2019, pp. 1439–1453. doi: 10.1109/WSC40007.2019.9004960.

[26] V. Sood, M. K. Nema, R. Kumar, and M. J. Nene, "Construction and Analysis of Petri Net Model for Distributed Cyber Physical Systems," *Def Sci J*, vol. 72, no. 5, pp. 721–731, Nov. 2022, doi: 10.14429/dsj.72.17987.

[27] A. S. Staines, "Concurrency and Petri Net Models," *International Journal of Circuits, Systems and Signal Processing*, vol. 16, pp. 852–858, Mar. 2022, doi: 10.46300/9106.2022.16.104.

[28] L. He and G. Liu, "Petri Net Based Symbolic Model Checking for Computation Tree Logic of Knowledge," Dec. 2020, doi: 10.1109/TCSS.2022.3164052.

[29] M. Chen, S. Hariharaputran, R. Hofestädt, B. Kormeier, and S. Spangardt, "Petri net models for the semi-automatic construction of large scale biological networks," *Nat Comput*, vol. 10, no. 3, pp. 1077–1097, Sep. 2011, doi: 10.1007/s11047-009-9151-y.

[30] Sheng-Uei Guan, Hsiao-Yeh Yu, and Jen-Shun Yang, "A prioritized Petri net model and its application in distributed multimedia systems," *IEEE Transactions on Computers*, vol. 47, no. 4, pp. 477–481, Apr. 1998, doi: 10.1109/12.675716.

[31] B. B. Ghebrial, "Professional IoT Applications Development training utilizing IoT Educational and Innovation Labs," in *Federated Africa and Middle East Conference on Software Engineering*, New York, NY, USA: ACM, Jun. 2022, pp. 90–90. doi: 10.1145/3531056.3542763.

[32] B. Qian *et al.*, "Orchestrating the Development Lifecycle of Machine Learning-based IoT Applications," *ACM Comput Surv*, vol. 53, no. 4, pp. 1–47, Jul. 2021, doi: 10.1145/3398020.

[33] Q. Anwar, M. Imran, and M. O'Nils, "Intelligence Partitioning as a Method for Architectural Exploration of Wireless SensorNode," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, Dec. 2016, pp. 935–940. doi: 10.1109/CSCI.2016.0180.

[34] F. Tsimpourlas, L. Papadopoulos, A. Bartsokas, and D. Soudris, "A Design Space Exploration Framework for Convolutional Neural Networks Implemented on Edge Devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2212–2221, Nov. 2018, doi: 10.1109/TCAD.2018.2857280.

[35] A. Bashir, M. Alhammadi, M. Awawdeh, and T. Faisal, "Effectiveness of using Arduino platform for the hybrid engineering education learning model," in *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, IEEE, Mar. 2019, pp. 1–6. doi: 10.1109/ICASET.2019.8714438.