SADIO Electronic Journal of Informatics and Operations Research (EJS) Vol. 25 No. 2 (2026) e-ISSN 1514-6774
https://revistas.unlp.edu.ar/ejs | https://doi.org/10.24215/15146774e112

# Enhanced Modeling of Reactive Systems using Non-Autonomous Petri Nets and Microservices

Mauricio Ludemann [1,2,3], Luis Ventre[1,2], Gabriel Valenzuela[1,2], Orlando Micolini [1,2].

[1] Universidad Nacional de Córdoba, Córdoba, Argentina
[2] Laboratorio de Arquitectura de Computadoras, FCEFyN
[3]mauri.ludemann@unc.edu.ar

**Abstract:** This paper presents an innovative framework for modeling and executing reactive systems by integrating Non-Autonomous Petri Nets (NAPN) with microservices architecture. Reactive systems in critical domains such as industrial control and IoT demand both formal verifiability and operational flexibility, yet existing approaches often struggle to balance these requirements. Our work addresses this gap by combining the mathematical rigor of NAPNs—which enable precise modeling of event-driven concurrency and synchronization—with the scalability and resilience of microservices.
In this extended version, the proposal incorporates a structured methodology that includes a complete event-processing flow and a clear separation between events, actions, data, and policies. This methodological refinement strengthens the correspondence between the formal model and its distributed implementation, ensuring that design-time properties are preserved during execution.
A key contribution is our hierarchical event taxonomy, which systematically classifies stimuli (temporal, asynchronous, or fault-based) to optimize processing in distributed environments. This taxonomy enables adaptive handling of both recognized and unknown events, enhancing system robustness in dynamic scenarios. The proposed architecture features an intelligent orchestrator that coordinates microservices based on event classification while maintaining consistency with the formal NAPN model.
Expected outcomes include portable, scalable, and formally verifiable reactive systems that preserve design-time properties during distributed execution. The framework particularly benefits industrial applications where reliability and real-time responsiveness are paramount. Future work will validate the approach through payment system case studies and explore machine learning extensions for dynamic performance optimization. This research bridges formal methods with modern distributed architectures, offering a principled yet practical solution for mission-critical reactive systems.

**Keywords:** Reactive systems, Non-Autonomous Petri Nets, microservices, event taxonomy, formal verification, distributed architectures.

# Modelado Mejorado de Sistemas Reactivos mediante Redes de Petri No Autónomas y Microservicios

**Resumen.** Este artículo presenta un marco innovador para el modelado y ejecución de sistemas reactivos mediante la integración de Redes de Petri No Autónomas (RPNA) con arquitecturas de microservicios. Los sistemas reactivos

Received May 2026; Accepted June 2026; Published July 2026

en dominios críticos como control industrial e IoT requieren tanto verificabilidad formal como flexibilidad operativa, sin embargo, los enfoques existentes suelen tener dificultades para equilibrar estos requisitos. Nuestro trabajo aborda esta brecha combinando el rigor matemático de las RPNA -que permiten modelado preciso de concurrencia y sincronización orientada a eventos- con la escalabilidad y resiliencia de los microservicios.
En esta versión extendida, la propuesta incorpora una metodología estructurada que incluye un flujo completo de procesamiento de eventos y una separación explícita entre eventos, acciones, datos y políticas. Esta integración metodológica fortalece la correspondencia entre el modelo formal y su implementación distribuida, asegurando que las propiedades verificadas en el diseño se mantengan durante la ejecución.
Una contribución clave es nuestra taxonomía jerárquica de eventos, que clasifica sistemáticamente los estímulos (temporales, asincrónicos o por fallas) para optimizar su procesamiento en entornos distribuidos. Esta taxonomía permite manejo adaptativo tanto de eventos reconocidos como desconocidos, mejorando la robustez del sistema en escenarios dinámicos. La arquitectura propuesta incluye un orquestador inteligente que coordina microservicios según la clasificación de eventos, manteniendo coherencia con el modelo formal RPNA.
Los resultados esperados incluyen sistemas reactivos portables, escalables y formalmente verificables que preservan las propiedades de diseño durante su ejecución distribuida. El marco beneficia particularmente aplicaciones industriales donde la confiabilidad y capacidad de respuesta en tiempo real son críticas. Trabajos futuros validarán el enfoque mediante casos de estudio en sistemas de pagos y explorarán extensiones con aprendizaje automático para optimización dinámica del desempeño. Esta investigación tiende un puente entre métodos formales y arquitecturas distribuidas modernas, ofreciendo una solución tanto teóricamente rigurosa como prácticamente viable para sistemas reactivos de misión crítica.

**Palabras clave**: Sistemas reactivos, Redes de Petri No Autónomas, microservicios, taxonomía de eventos, verificación formal, arquitecturas distribuidas.

# 1 Introducción

Los sistemas reactivos son esenciales en dominios donde la capacidad de procesar eventos en tiempo real determina su eficacia, como en control industrial, IoT y sistemas distribuidos. Estos entornos demandan no solo alta disponibilidad y escalabilidad, sino también mecanismos formales para garantizar consistencia en entornos dinámicos y concurrentes. Sin embargo, su diseño enfrenta desafíos persistentes, como la falta de modelos unificados que equilibren expresividad matemática y flexibilidad arquitectónica (Cerny et al., 2018).

Las arquitecturas de microservicios han surgido como un paradigma viable para sistemas distribuidos, permitiendo modularidad y escalamiento independiente (Shadija et al., 2017). No obstante, su adopción en sistemas reactivos requiere enfoques que integren modelos formales para garantizar coherencia en el manejo de eventos. Este trabajo propone un marco que combina Redes de Petri No Autónomas (RPNA) con microservicios, aprovechando las primeras para modelar comportamientos reactivos de manera formal y los segundos para implementación distribuida y resiliente. Las RPNA ofrecen fundamentos matemáticos para representar concurrencia y sincronización, mientras que los microservicios facilitan el despliegue elástico y la tolerancia a fallos (Richards, 2015).

Adicionalmente, se introduce una taxonomía de eventos que categoriza estímulos—temporales, asíncronos o de falla—para optimizar su procesamiento en escenarios críticos. Esta clasificación mejora la capacidad de diagnóstico en sistemas industriales, donde la trazabilidad es crucial (Bhamare et al., 2020). Al unir modelos formales con patrones modernos de arquitectura, este trabajo busca superar las limitaciones de enfoques ad hoc, ofreciendo un método riguroso para sistemas donde la reactividad y la confiabilidad son prioritarias.

# 2 Fundamentación Teórica

## 2.1 Redes de Petri No Autónomas

Las Redes de Petri No Autónomas (RPNA) representan una evolución significativa respecto a las redes de Petri clásicas, al incorporar la capacidad de modular el comportamiento del sistema en función de eventos externos. Esta característica las hace particularmente adecuadas para modelar sistemas donde la dinámica no solo depende del estado interno, sino también de interacciones con el entorno (David & Alla, 2010). A diferencia de los modelos tradicionales, las RPNA permiten una representación formal de sistemas concurrentes y distribuidos, facilitando el análisis de propiedades como vivacidad, seguridad y fairness. Además, su capacidad para simular comportamientos complejos las convierte en una herramienta valiosa para la verificación de sistemas reactivos antes de su implementación (Murata, 1989).

En el contexto de sistemas industriales y de IoT, donde la concurrencia y la respuesta a eventos son críticas, las RPNA ofrecen un marco teórico robusto para garantizar consistencia en el procesamiento de estímulos. Su integración con arquitecturas modernas, como los microservicios, puede cerrar la brecha entre los modelos formales y las implementaciones distribuidas, asegurando que las propiedades verificadas en el diseño se mantengan durante la ejecución (Macaulay & Singer, 2011).

## 2.2 Arquitectura de Microservicios

La arquitectura de microservicios ha surgido como un paradigma dominante en el desarrollo de sistemas distribuidos, superando muchas de las limitaciones de las arquitecturas monolíticas y orientadas a servicios (SOA). Al descomponer las aplicaciones en servicios independientes, cada uno con responsabilidades bien delimitadas, este enfoque promueve la escalabilidad horizontal, la resiliencia y la capacidad de evolución incremental (Cerny et al., 2018). Sin embargo, la adopción de microservicios en sistemas reactivos introduce desafíos adicionales, particularmente en la gestión de la consistencia y la coordinación entre servicios distribuidos.

Según Shadija (Shadija et al., 2017), la granularidad de los microservicios es un factor determinante en su desempeño: mientras que una descomposición excesiva puede aumentar la sobrecarga por comunicación, una modularización insuficiente limita la escalabilidad. Para sistemas reactivos, donde la latencia y el throughput son críticos, es esencial equilibrar estos aspectos mediante patrones como Event Sourcing y CQRS (Command Query Responsibility Segregation), que permiten manejar flujos de eventos de manera eficiente (Richards, 2015).

### 2.3 Sistemas Reactivos Orientados a Eventos

Los sistemas reactivos se caracterizan por su capacidad de responder de manera asincrónica a estímulos externos, priorizando la escalabilidad, la resiliencia y la capacidad de procesamiento en tiempo real. Estos sistemas son fundamentales en dominios como el control industrial, donde la detección y clasificación de eventos deben realizarse sin comprometer la estabilidad del sistema (Macaulay & Singer, 2011). Un diseño efectivo requiere no solo una infraestructura que permita el desacoplamiento entre productores y consumidores de eventos, sino también mecanismos para garantizar el orden, la durabilidad y el procesamiento oportuno de los mismos.

La combinación de modelos formales, como las RPNA, con arquitecturas basadas en eventos y microservicios, ofrece un enfoque integral para abordar estos desafíos. Por un lado, las RPNA proporcionan la formalidad necesaria para garantizar propiedades críticas en el diseño; por otro, los microservicios permiten una implementación flexible y escalable. Esta sinergia es particularmente relevante en entornos donde la reactividad debe coexistir con altos requisitos de confiabilidad, como en sistemas SCADA y aplicaciones de IoT industrial (Cerny et al., 2018).

## 3 Estado del Arte

El diseño de sistemas reactivos distribuidos ha sido abordado desde múltiples perspectivas teóricas y tecnológicas. Sin embargo, la integración de modelos formales basados en redes de Petri con arquitecturas de microservicios continúa siendo un área con escasa sistematización. Este apartado presenta una síntesis del estado del arte relevante en torno al procesamiento de eventos, los formalismos para modelado reactivo y los enfoques existentes para aplicar redes de Petri en sistemas distribuidos.

### 3.1 Procesamiento de Eventos y sus Limitaciones Formales

El Procesamiento de Eventos Complejos (CEP, Complex Event Processing) constituye uno de los enfoques clásicos para la detección y tratamiento de patrones en flujos de datos. En su formulación tradicional, CEP se basa en agentes que filtran, transforman o correlacionan eventos en tiempo real (Etzion & Niblett, 2011). Sin embargo, la literatura reconoce que su semántica carece de rigor matemático y que la composición de eventos suele expresarse mediante lenguajes sin base formal estricta, dificultando la verificación de propiedades en sistemas críticos (Grez et al., 2019).

Para mitigar esta limitación, se han propuesto lenguajes formales como Complex Event Logic (CEL), que definen patrones de coincidencia mediante expresiones declarativas. No obstante, tanto CEP como CEL siguen presentando un acoplamiento fuerte entre la especificación de eventos y la lógica de control del sistema, lo que complica su integración con arquitecturas distribuidas donde la lógica debe ser desacoplada, verificable y portable. Esta brecha motiva la búsqueda de modelos alternativos con semántica formal más clara.

### 3.2 Modelos Formales para Sistemas Reactivos

En el ámbito del modelado reactivo, los Statecharts (Harel & Politi, 1999) proporcionan un lenguaje orientado a eventos con soporte para jerarquías, concurrencia y acciones, ampliamente utilizado en ingeniería de sistemas. Sin

embargo, pese a su expresividad operacional, carecen de una semántica matemática que permita verificación formal exhaustiva. Por su parte, las estructuras de eventos formuladas por Winskel (Winskel, 1987) y sistematizadas por Hoffmann (Hoffmann, 2011) ofrecen un enfoque basado en orden parcial para representar causalidades, conflictos y concurrencia, constituyendo un fundamento teórico sólido para el análisis de sistemas que reaccionan a estímulos externos.

Las redes de Petri —en sus variantes autónomas y no autónomas— se posicionan como uno de los modelos más utilizados para representar comportamiento reactivo con semántica formal rigurosa (Murata, 1989)(David & Alla, 2010). Su capacidad para modelar concurrencia, sincronización, restricciones de habilitación y evolución del estado las convierte en una opción natural para formalizar la lógica de sistemas distribuidos.

### 3.3 **Arquitecturas de Microservicios y Sistemas Distribuidos**

Las arquitecturas de microservicios emergieron como evolución de los enfoques orientados a servicios, buscando mayor modularidad, autonomía y escalabilidad (Cerny et al., 2018). En contraste con los sistemas monolíticos o los ESB tradicionales utilizados en SOA, los microservicios promueven comunicación ligera, despliegue independiente y resiliencia ante fallos. En el contexto de sistemas reactivos, los principios de la arquitectura reactiva —aislamiento, comunicación asincrónica y elasticidad— han sido ampliamente difundidos por Bonér (Bonér, 2016), destacando la necesidad de mecanismos que permitan mantener coherencia en presencia de concurrencia distribuida.

A pesar de sus ventajas, la adopción de microservicios introduce desafíos importantes: consistencia entre servicios, coordinación en tiempo real y ausencia de modelos formales que garanticen propiedades globales de ejecución. Estas limitaciones fortalecen la motivación para integrar arquitecturas de microservicios con técnicas de verificación formal.

### 3.4 **Redes de Petri Aplicadas a Microservicios**

En los últimos años han surgido intentos de vincular redes de Petri con arquitecturas de microservicios. Merkouche (Merkouche et al., 2023) proponen un modelo de autoscaling basado en una extensión denominada Hierarchical Parallel Petri Nets (HPPNs), que incorpora portales y jerarquías para representar decisiones de escalado. Sin embargo, esta variante altera la semántica clásica de las redes de Petri, dificultando la aplicación de análisis formales tradicionales y volviéndola más cercana a un lenguaje de simulación.

Por otro lado, el estudio de Soylu & Demirors (Soylu & Demirors, 2023) demuestra la utilidad de las redes de Petri 1-safe para modelar microservicios mediante abstracción y composición. Su enfoque permite analizar deadlocks y sincronización entre servicios, pero no aborda la necesidad de desacoplar eventos, acciones y datos, ni propone mecanismos para preservar propiedades del modelo durante la ejecución distribuida.

De manera complementaria, en el dominio del procesamiento distribuido de eventos, trabajos como el de Scattone & Braghetto (Scattone & Braghetto, 2018) exploran arquitecturas orientadas a microservicios para CEP, pero sin incorporar formalismos matemáticos que aseguren consistencia del sistema.

### 3.5 **Brechas Identificadas**

Pese a los avances mencionados, no existe actualmente un marco que integre:

- una taxonomía formal de eventos,
- una red de Petri no autónoma como modelo rector de la lógica,
- una arquitectura desacoplada de microservicios,
- y mecanismos para preservar la coherencia formal del modelo en tiempo de ejecución.

La literatura muestra aproximaciones parciales, pero ninguna ofrece una solución completa que combine formalidad matemática, desacoplamiento arquitectónico y escalabilidad operativa. Esta brecha constituye el principal espacio de contribución de la presente investigación.

## 4 **Propuesta**

La solución propuesta en este trabajo integra modelos formales basados en Redes de Petri No Autónomas (RPNA) con una arquitectura distribuida de microservicios. El objetivo es proporcionar una metodología rigurosa para la ejecución de sistemas reactivos que deben procesar estímulos externos de manera concurrente, manteniendo la verificabilidad del sistema y garantizando independencia entre componentes. Este enfoque combina una taxonomía explícita de eventos, un modelo formal ejecutable y una infraestructura desacoplada basada en microservicios que permite escalar, mantener y validar sistemas reactivos de forma robusta.

### 4.1 **Visión General del Enfoque Propuesto**

El enfoque se basa en separar claramente cuatro entidades fundamentales del sistema:

1. Eventos, que representan estímulos del entorno.
2. Datos, que definen el estado informativo del sistema.
3. Acciones, que representan los efectos del sistema hacia su interior o hacia agentes externos.
4. Políticas, que resuelven comportamientos no deterministas derivados de conflictos en la RPNA.

Esta separación explícita evita acoplamientos indeseados y permite que la RPNA actúe como núcleo determinístico, encargado de dirigir la evolución del sistema mediante un modelo matemático verificable. El procesamiento distribuido se logra encapsulando los distintos roles funcionales del sistema en microservicios ligeros, que interactúan mediante eventos y acciones definidos formalmente.

El ciclo completo consiste en:

1. recepción y clasificación del evento entrante,
2. identificación de la transición asociada en la RPNA,
3. evaluación de guardas y condiciones externas,
4. disparo de la transición correspondiente,
5. ejecución de acciones resultantes,
6. generación de nuevos eventos internos,
7. propagación controlada hacia otros microservicios o actores del sistema.

### 4.2 Taxonomía de Eventos para Sistemas Reactivos

El núcleo de nuestra propuesta radica en una taxonomía de eventos (Fig. 1) diseñada para optimizar el procesamiento en sistemas reactivos complejos. Esta clasificación jerárquica aborda una limitación fundamental identificada en la literatura actual: la falta de esquemas formales para categorizar estímulos en entornos distribuidos donde confluyen microservicios y modelos basados en eventos (Cerny et al., 2018). La taxonomía distingue primero entre eventos reconocidos y desconocidos, estableciendo así un primer filtro para el manejo de estímulos inesperados - aspecto crítico en sistemas industriales donde la resiliencia es prioritaria (Macaulay & Singer, 2011) -.

Para los eventos reconocidos, la clasificación se profundiza según su capacidad de ejecución. Los eventos ejecutables se subdividen en simples (disparadores unitarios de transiciones) y compuestos (agregaciones de eventos simples que requieren coordinación), diferenciación que responde al desafío de granularidad identificado por Shadija (Shadija et al., 2017) en arquitecturas distribuidas. Los no ejecutables incorporan el concepto de memoria, permitiendo almacenamiento temporal cuando existen dependencias contextuales, mecanismo que mitiga problemas de sincronización en entornos asíncronos (Richards, 2015).

Los eventos desconocidos implementan políticas configurables (almacenamiento o descarte), proporcionando flexibilidad para escenarios donde el análisis posterior pueda revelar patrones significativos. Esta dualidad es particularmente relevante en sistemas de control industrial, donde eventos aparentemente anómalos pueden indicar fallas incipientes (Macaulay & Singer, 2011). La taxonomía se implementa mediante un motor de reglas acoplado a la capa de orquestación de microservicios, permitiendo que la clasificación influya tanto en el enrutamiento como en las estrategias de procesamiento.

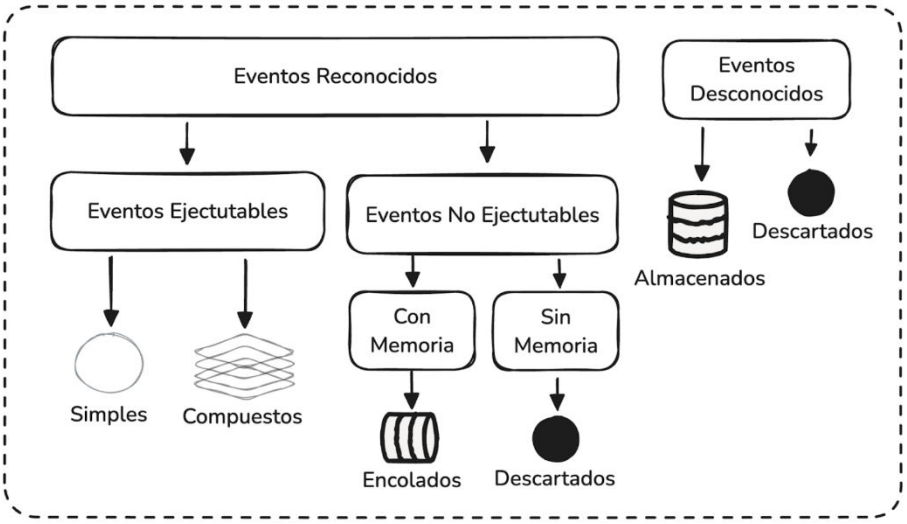


**Fig. 1**. Taxonomía de Eventos

### 4.3 Integración con Redes de Petri No Autónomas y Microservicios

La taxonomía propuesta se articula con un modelo formal basado en Redes de Petri No Autónomas (RPNA), donde los tipos de eventos determinan las condiciones de habilitación de transiciones. Esta integración proporciona beneficios significativos: las RPNA ofrecen verificación formal de propiedades del sistema, mientras que la taxonomía garantiza que los eventos se procesen según su naturaleza y prioridad. En el nivel arquitectónico, cada categoría de evento se asocia a microservicios especializados, optimizando así la asignación de recursos computacionales según la complejidad del procesamiento requerido.

Este enfoque híbrido resuelve problemas prácticos identificados en la literatura reciente (Ventre et al., 2024). Por un lado, mitiga la sobrecarga de comunicación en microservicios mediante un enrutamiento inteligente basado en la taxonomía (Shadija et al., 2017). Por otro, mantiene garantías formales mediante las RPNA, cruciales en dominios donde la corrección del sistema es no negociable (Cerny et al., 2018). La implementación utiliza colas de mensajes con prioridades diferenciadas y mecanismos de retropropagación para eventos compuestos, asegurando consistencia eventual sin comprometer la capacidad de respuesta.

### 4.4 Arquitectura General de la Solución

Nuestra solución integra los principios de las arquitecturas orientadas a microservicios con el rigor formal de las Redes de Petri No Autónomas (RPNA), creando un marco robusto para el desarrollo de sistemas reactivos. El núcleo del sistema consiste en un orquestador inteligente que actúa como intermediario entre los eventos entrantes y los microservicios especializados. Este componente central, inspirado en los patrones de mediación descritos por (Richards, 2015), no solo dirige el flujo de eventos según su clasificación taxonómica, sino que también mantiene la coherencia con el modelo formal representado en las RPNA.

La arquitectura propuesta aborda tres desafíos críticos identificados en la literatura reciente. Primero, resuelve el problema de acoplamiento en sistemas distribuidos mediante un patrón de publicación-suscripción que permite a los microservicios operar de manera independiente mientras responden a eventos de su dominio específico (Cerny et al., 2018). Segundo, incorpora mecanismos de retroalimentación que ajustan dinámicamente las prioridades de procesamiento basándose en la taxonomía de eventos, optimizando así el uso de recursos en escenarios de alta carga (Shadija et al., 2017). Tercero, implementa un módulo de reconciliación que detecta y resuelve inconsistencias entre el estado real del sistema y su representación en las RPNA, característica esencial para entornos industriales donde la tolerancia a fallos es primordial (Macaulay & Singer, 2011).

Cada microservicio en nuestra arquitectura encapsula no solo la lógica de negocio, sino también un fragmento de la RPNA global que modela su comportamiento reactivo. Esta innovadora descomposición espacial del modelo formal permite escalar horizontalmente el sistema sin perder las garantías de corrección proporcionadas por las RPNA. La comunicación entre servicios se realiza mediante un bus de eventos que soporta diferentes protocolos (HTTP/REST, gRPC, mensajería asíncrona), seleccionados automáticamente según el tipo de evento y los requisitos de calidad de servicio.

La Fig. 2 muestra la arquitectura general propuesta, la cual organiza las responsabilidades en al menos dos microservicios principales:

- **Manejador de Eventos**, encargado de recibir estímulos externos e internos y clasificarlos mediante la taxonomía establecida.
- **Ejecutor**, microservicio central que ejecuta la lógica formal mediante el marcado, la matriz de incidencia y la evaluación de guardas. De acuerdo con el sistema que se esté diseñando, puede haber uno o múltiples ejecutores. El ejecutor a su vez posee dos componentes clave:
    - **Lógica:** representada por una RPNA
    - **Política:** mecanismo utilizado para aplicar determinismo al sistema cuando hay conflictos en la RPNA

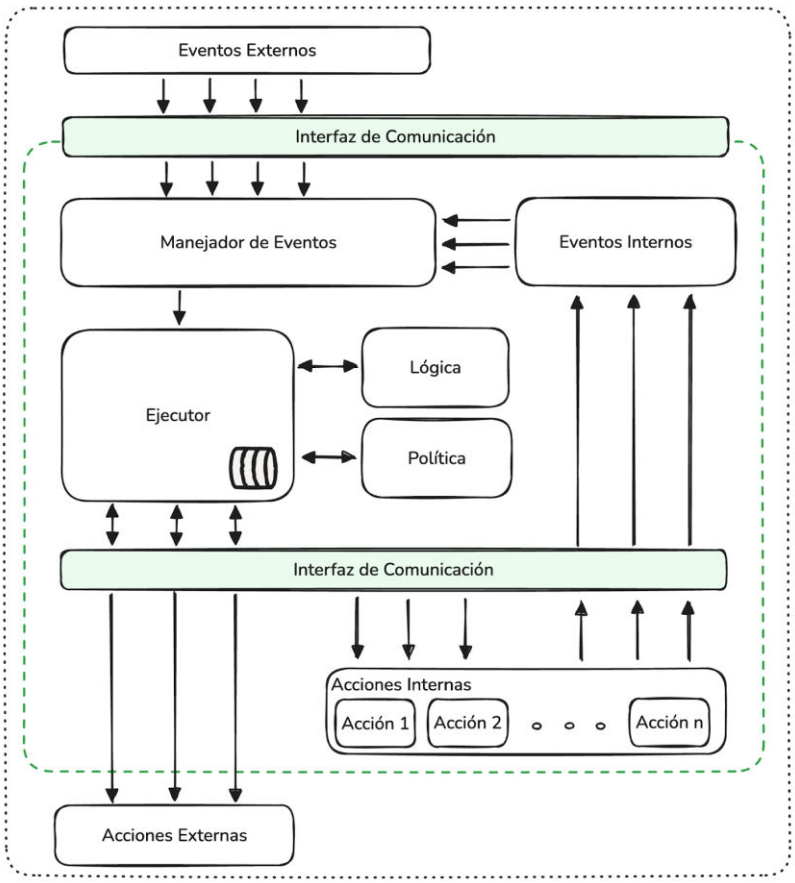


**Fig. 2**. Arquitectura de la Solución

Esta arquitectura distribuye la responsabilidad del procesamiento en componentes independientes, reduciendo el acoplamiento y permitiendo que cada microservicio escale de manera autónoma. La RPNA queda encapsulada dentro del microservicio Model Executor, lo cual habilita su portabilidad y aislamiento respecto de otros procesos. A su vez, la clasificación de eventos y la ejecución de acciones pueden implementarse en lenguajes o plataformas distintas, favoreciendo la flexibilidad tecnológica.

### 4.5 **Flujo Completo de Ejecución del Sistema**

El funcionamiento interno del sistema se compone de once pasos fundamentales:

1. **Recepción del evento externo** desde un sensor, usuario o servicio remoto.
2. **Clasificación del evento** según la taxonomía propuesta.
3. **Determinación del tipo de evento**, identificando si requiere memoria, si es descartable o si representa una excepción.
4. **Mapeo del evento a una transición** de la RPNA, aplicando las reglas definidas.
5. **Evaluación de guardas y condiciones externas**, tales como disponibilidad de recursos o cumplimiento de restricciones temporales.
6. **Verificación de sensibilización**, comprobando si el marcado actual habilita la transición.
7. **Disparo de la transición**, modificando el estado del sistema mediante la matriz de incidencia.
8. **Ejecución de acciones internas o externas**, dependiendo de la salida de la transición.
9. **Generación de eventos internos adicionales**, si la lógica del sistema lo requiere.
10. **Persistencia del estado**, manteniendo la consistencia entre ejecuciones.
11. **Emisión de resultados hacia otros microservicios**, completando el ciclo reactivo.

Este flujo modela un ciclo completo de reactividad formalizado, distribuido y verificable.

## 5 Resultados

La implementación de esta arquitectura híbrida promete avances significativos en tres dimensiones clave. En cuanto a portabilidad, el uso de contenedores para empaquetar microservicios junto con sus fragmentos de RPNA asociados permite desplegar el sistema en diversas plataformas, desde entornos cloud hasta dispositivos edge en instalaciones industriales (Macaulay & Singer, 2011). La escalabilidad se beneficia de la capacidad de distribuir selectivamente la carga de procesamiento según la categorización taxonómica de eventos, evitando así los cuellos de botella comunes en arquitecturas monolíticas (Shadija et al., 2017).

La verificabilidad del sistema constituye quizás su ventaja más distintiva. Al mantener una correspondencia biyectiva entre los microservicios operacionales y el modelo formal en RPNA, es posible realizar análisis estáticos de propiedades como ausencia de deadlocks o cumplimiento de invariantes, incluso en sistemas distribuidos complejos (Cerny et al., 2018). Esto representa un avance significativo respecto a las arquitecturas basadas puramente en eventos, donde tradicionalmente ha sido difícil garantizar propiedades globales.

La taxonomía de eventos incorporada no solo mejora la eficiencia del procesamiento, sino que también dota al sistema de una capacidad adaptativa única. Eventos inicialmente clasificados como desconocidos pueden ser reevaluados y reclasificados dinámicamente a medida que el sistema "aprende" nuevos patrones, una característica particularmente valiosa en entornos industriales donde las condiciones operativas evolucionan constantemente (Macaulay & Singer, 2011). Esta flexibilidad, combinada con el formalismo de las RPNA, crea un marco único que equilibra rigor matemático con adaptabilidad práctica.

Además, la incorporación de una descripción formal del ciclo completo de procesamiento —desde la recepción y clasificación de eventos hasta la ejecución de acciones y la generación de nuevos estímulos internos— aporta un nivel de sistematización que complementa los aspectos de portabilidad, escalabilidad y verificabilidad previamente discutidos. Este flujo detallado, junto con la integración explícita de la taxonomía de eventos y la separación entre eventos, acciones y datos, consolida un marco metodológico que facilita tanto la implementación distribuida como el razonamiento formal sobre el comportamiento del sistema. Estos resultados fortalecen la propuesta al demostrar cómo el modelo no solo es teóricamente riguroso, sino también aplicable en arquitecturas modernas basadas en microservicios.

# 6 Conclusión

Este trabajo presenta un marco innovador que integra Redes de Petri No Autónomas (RPNA) con arquitecturas de microservicios para abordar los desafíos fundamentales en el diseño de sistemas reactivos. La combinación propuesta logra un equilibrio crucial entre rigor formal y flexibilidad operativa, permitiendo el desarrollo de sistemas que son simultáneamente verificables matemáticamente y altamente adaptables a entornos dinámicos. Como destacan (Cerny et al., 2018), esta sinergia entre modelos formales y arquitecturas distribuidas modernas representa un avance significativo en el campo de los sistemas reactivos empresariales e industriales.

La taxonomía de eventos desarrollada constituye una contribución clave al proporcionar un esquema estructurado para clasificar y gestionar estímulos en sistemas complejos. Este enfoque sistemático, como señala (Richards, 2015), es particularmente valioso en escenarios donde la naturaleza heterogénea de los eventos puede comprometer la estabilidad del sistema. Nuestra categorización de eventos reconocidos y desconocidos, con sus respectivas subclases, ofrece un mecanismo robusto para manejar tanto flujos operativos normales como situaciones excepcionales, aspecto crítico en entornos como los sistemas de control industrial (Macaulay & Singer, 2011).

Los resultados de nuestra propuesta - portabilidad, escalabilidad y verificabilidad - responden directamente a las limitaciones identificadas en la literatura actual sobre sistemas distribuidos. La capacidad de descomponer un modelo formal de RPNA en microservicios especializados, como analiza (Shadija et al., 2017), permite superar el tradicional compromiso entre consistencia global y desempeño distribuido. Esto es especialmente relevante en dominios como los sistemas de pagos, donde nuestro trabajo futuro planea validar empíricamente el enfoque.

La principal contribución de esta investigación radica en demostrar cómo los paradigmas formales y las arquitecturas modernas pueden complementarse mutuamente. Las RPNA proporcionan la base teórica para garantizar propiedades críticas del sistema, mientras que los microservicios permiten implementaciones flexibles y escalables. Esta dualidad abre nuevas posibilidades para el desarrollo de sistemas reactivos en dominios donde tanto la corrección formal como la adaptabilidad operacional son requisitos fundamentales.

Además, la incorporación de una descripción formal del ciclo completo de procesamiento de eventos, junto con el modelo arquitectónico que define la interacción entre los distintos microservicios, refuerza la aplicabilidad de la propuesta en sistemas distribuidos reales. Esta integración metodológica —que articula taxonomía, flujo operacional y ejecución formal mediante RPNA— no solo facilita la implementación práctica del modelo, sino que también establece un puente claro entre la teoría y la ingeniería de software moderna. La capacidad de trasladar el

comportamiento formal del sistema hacia una arquitectura ejecutable, desacoplada y escalable, constituye un avance concreto hacia la construcción de sistemas reactivos transparentes, auditables y sostenibles en entornos de alta criticidad.

## 7 Trabajos Futuros

La implementación de un prototipo funcional en el dominio de sistemas de pagos representa solo el primer paso en una agenda de investigación más amplia. Como señalan (Cerny et al., 2018), la validación empírica de arquitecturas híbridas que combinan microservicios con modelos formales sigue siendo un área poco explorada, particularmente en escenarios del mundo real con requisitos estrictos de rendimiento y consistencia. Nuestro experimento con pagos instantáneos y programados, el cual actualmente se está llevando a cabo, servirá como banco de pruebas para evaluar no solo los aspectos técnicos de la solución, sino también su capacidad para manejar flujos de trabajo complejos con dependencias temporales y lógicas.

Una línea prometedora de investigación futura explora la integración de técnicas de aprendizaje automático con nuestro marco basado en Redes de Petri No Autónomas (RPNA). Como identifican (Shadija et al., 2017), los sistemas de microservicios modernos enfrentan el desafío de optimizar dinámicamente su desempeño ante cargas de trabajo variables. La incorporación de modelos predictivos que anticipen patrones de eventos podría mejorar significativamente la eficiencia del orquestador central, permitiendo un aprovisionamiento preventivo de recursos y una asignación más inteligente de tareas.

En el contexto de sistemas industriales, donde la ciberseguridad es primordial (Macaulay & Singer, 2011), planeamos extender nuestra taxonomía de eventos para incorporar categorías específicas de amenazas y anomalías. Esto permitiría desarrollar mecanismos de detección temprana integrados directamente en el modelo formal, combinando así las ventajas de las RPNA para modelar comportamientos del sistema con técnicas avanzadas de monitoreo de seguridad.

Otra dirección importante implica investigar patrones de descomposición óptima para mapear modelos de RPNA complejos a conjuntos de microservicios. Como discute (Richards, 2015), el diseño efectivo de arquitecturas de microservicios requiere equilibrar múltiples factores, incluyendo cohesión funcional, granularidad adecuada y minimización de acoplamiento. Nuestro trabajo futuro explorará métricas cuantitativas para evaluar diferentes estrategias de partición del modelo formal, con el objetivo de desarrollar directrices prácticas para ingenieros de software.

Además de las líneas de investigación mencionadas, una dirección especialmente prometedora consiste en avanzar hacia la automatización parcial del proceso de implementación, generando artefactos de software directamente a partir del modelo formal. Dado que la RPNA ya define explícitamente la estructura causal del sistema, las condiciones de ejecución y la relación entre eventos, acciones y datos, resulta factible explorar mecanismos de traducción automática que produzcan esqueletos de microservicios, contratos de comunicación o artefactos de despliegue a partir del modelo. Esta línea de trabajo permitiría cerrar el ciclo entre especificación formal y desarrollo práctico, reduciendo el costo de implementación y asegurando la consistencia entre modelo y código.

Asimismo, se proyecta investigar la extensión del enfoque hacia modelos jerárquicos y coloreados de Redes de Petri, con el objetivo de mejorar la expresividad y modularidad del modelo sin sacrificar su verificabilidad. Las RPNAs jerárquicas permitirían representar subsistemas completos como transiciones abstractas, facilitando el mapeo directo a microservicios individuales, mientras que las redes

coloreadas aportarían un nivel adicional de compactación y claridad, especialmente en sistemas con grandes dominios de datos. Estas extensiones abrirían la puerta a nuevas herramientas de análisis y simulación, y habilitarían la comparación sistemática entre diferentes estilos de modelado.

Finalmente, se considera de alto interés explorar la evaluación empírica del flujo de 11 pasos propuesto para el procesamiento de eventos, midiendo métricas como latencia, rendimiento, robustez ante eventos desconocidos y costo operativo en entornos reales o simulados. Este análisis, que actualmente se está llevando a cabo, permitirá no solo validar la arquitectura desde un punto de vista ingenieril, sino también comparar la propuesta frente a alternativas existentes, como arquitecturas CEP tradicionales o sistemas dirigidos por eventos no formales.

## Referencias


Bhamare, D., Zolanvari, M., Erbad, A., Jain, R., Khan, K. y Meskin, N. (2020). Cybersecurity for industrial control systems: A survey. *Computer & Security, 89*, 101677. https://doi.org/10.1016/j.cose.2019.101677

Bonér, J. (2016). *Reactive microservices architecture*. O'Reilly. https://www.oreilly.com/library/view/reactive-microservices-architecture/9781491975664/

Cerny, T., Donahoo, M. J. y Trnka, M. (2018). Contextual understanding of microservice architecture: Current and future directions. *ACM SIGAPP Applied Computing Review*, *17*(4), 29–45. https://doi.org/10.1145/3183628.3183631

David, R. y Alla, H. (2010). *Discrete, continuous, and hybrid Petri Nets.* (2ª ed.). https://doi.org/10.1007/b138130

Etzion, O. y Niblett, P. (2011). *Event processing in action.* Manning Publications. https://www.manning.com/books/event-processing-in-action

Grez, A., Riveros, C. y Ugarte, M. (2019). A formal framework for complex event processing. *Leibniz International Proceedings in Informatics, 127*, 5:1-5:18. https://doi.org/10.4230/LIPIcs.ICDT.2019.5

Harel, D. y Politi, M. (1998). *Modeling reactive systems with statecharts: The statemate approach*. McGraw-Hill. https://dl.acm.org/doi/10.5555/552084

Hoffmann, M. (2011). *Event structures* [documento PDF]. https://depend.cs.uni-saarland.de/fileadmin/user_upload/depend/neuhaeusser/concurrency_seminar_2011/event_structures.pdf

Macaulay, T. y Singer, B. (2011). *Cybersecurity for industrial control systems.* Auerbach Publications. https://doi.org/10.1201/b11352

Merkouche, S., Bouanaka, C. y Benkhelifa, E. (2023). A Petri net-based formal modeling for microservices auto-scaling. *IEEE International Conference on Computer Systems and Applications (AICCSA), 20 ACS*, 1-8. https://doi.org/10.1109/AICCSA59173.2023.10479351

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE, 77*(4), 541-580, https://doi.org/10.1109/5.24143

Richards, M. (2015). *Microservices vs. service-oriented architecture*. O'Reilly. https://www.oreilly.com/library/view/microservices-vs-service-oriented/9781491975657/

Scattone, F. F. y Braghetto, K. R. (2018). A microservices architecture for distributed complex event processing in smart cities. *International Symposium on Reliable Distributed Systems Workshops (SRDSW), 37*, 6-9 https://doi.org/10.1109/SRDSW.2018.00012

Shadija, D., Rezai, M. y Hill, R. (2017). Microservices: Granularity vs. performance. *arXiv*, 1709.09242. http://arxiv.org/abs/1709.09242

Soylu, G. K. y Demirors, O. (2023). An exploratory case study: Using Petri nets for modelling microservice-based systems. *Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 49*, 254–261. https://doi.org/10.1109/SEAA60479.2023.00047

Ventre, L. O., Micolini, O., Ludemann, M., Carranza, A., D'Andrea, D. y Candotti, E. (2024). *Caso de estudio: metodología para el diseño y desarrollo de sistemas embebidos distribuidos* [ponencia]. XXIX Congreso Argentino de Ciencias de La Computación (CACIC). Red de Universidades con Carreras en Informática. Luján, Argentina. http://sedici.unlp.edu.ar/handle/10915/164820

Winskel, G. (1987). Event structures. W. Brauer, W. Reisig y G. Rozenberg (Eds.), *Petri Nets: Applications and Relationships to Other Models of Concurrency. Lecture Notes in Computer Science* (vol. 255, pp. 325–392). Springer. https://doi.org/10.1007/3-540-17906-2_31