Revisión de herramientas para el desarrollo de Sistemas Multiagentes

1st Bryan Mendoza
Facultad de Ingeniería
Universidad de Cuenca
Cuenca, Ecuador
bsteven.mendoza7@ucuenca.edu.ec

2nd David Romero
Facultad de Ingeniería
Universidad de Cuenca
Cuenca, Ecuador
david.romeroa@ucuenca.edu.ec

***Abstract*—Los sistemas multiagente (MAS) son una arquitectura ampliamente utilizada para el desarrollo de aplicaciones distribuidas que requieren coordinación entre múltiples entidades autónomas. Sin embargo, el diseño, implementación y validación de estos sistemas presenta desafíos significativos, especialmente en aplicaciones críticas donde la comunicación entre agentes debe ser precisa y confiable. Este artículo presenta un análisis de cuatro artículos que presentan herramientas, plataformas actuales para el desarrollo de sistemas multiagente e implementaciones, con el objetivo de identificar sus enfoques, capacidades y posibles aplicaciones prácticas. Entre ellas se destacan AUTOGEN STUDIO, una herramienta sin código orientada al diseño declarativo de flujos multiagente, y una plataforma basada en JADE y OPNET que permite la validación precisa de sistemas MAS bajo simulaciones realistas de red. Herramientas como AUTOGEN también nos ayudan a trabajar con modelos extenso de lenguaje teniendo varios casos de uso donde se puede ver su efectividad.**

***Keywords*—Multi-Agent Systems, MAS, no-code development, frameworks, herramientas de desarrollo, LLM.**

I. Introducción

Los sistemas multiagente (Multi-Agent Systems, MAS) son una arquitectura de software distribuido compuesta por múltiples entidades autónomas llamadas agentes, capaces de percibir su entorno, comunicarse entre sí y tomar decisiones independientes o coordinadas para alcanzar objetivos definidos [1]. Este paradigma ha demostrado ser eficaz para resolver problemas complejos en entornos dinámicos, como la automatización industrial, la robótica, los sistemas energéticos y la gestión de emergencias [2].

En los últimos años, el creciente interés en MAS ha impulsado el desarrollo de herramientas y frameworks que facilitan su implementación. Sin embargo, el diseño de estos sistemas continúa siendo una tarea compleja que involucra la configuración de múltiples parámetros, la orquestación de agentes, y la evaluación de sus interacciones. Además, la validación de aplicaciones críticas requiere modelos precisos de la infraestructura de comunicación y ejecución realista del código de agentes [3].

Frente a esta necesidad, han surgido soluciones que van desde plataformas orientadas a desarrolladores, como *JADE* [4], hasta herramientas sin código como *AUTOGEN STUDIO*, que ofrecen entornos visuales para la creación rápida de flujos de trabajo multiagente. La elección de una u otra herramienta depende tanto de la experiencia del desarrollador como de las necesidades específicas del sistema a implementar.

El presente artículo tiene como objetivo analizar y comparar herramientas actuales para el desarrollo de sistemas multiagente. Este estudio resulta relevante para comprender el estado del arte, conocer qué soluciones están disponibles y evaluar su aplicabilidad en distintos contextos. Para ello, se analizan cuatro artículos científicos que presentan plataformas concretas: entre ellas, *AUTOGEN STUDIO*, orientada al desarrollo declarativo sin código, y una plataforma federada que integra JADE con el simulador OPNET para validar aplicaciones críticas bajo condiciones realistas de red.

II. AUTOGEN STUDIO: A No-Code Developer Tool for Building and Debugging Multi-Agent Systems

Este articulo presenta AUTOGEN STUDIO, una herramienta de desarrollo sin código que permite prototipar, depurar y evaluar rápidamente flujos de trabajo multiagente, basada en el framework AUTOGEN. La herramienta está enfocada en el desarrollo integral (IU, backend, web y API de Python) para especificar y depurar de forma declarativa flujos de trabajo multiagente, tanto con intervención humana como sin ella. Además, constituye el primer proyecto de código abierto que explora una interfaz no-code para el desarrollo autónomo de sistemas multiagentes.

Actualmente los desarrolladores deben configurar una gran cantidad de parámetros al construir sistemas multiagentes, incluyendo la definición de agentes (modelo a utilizar, indicaciones, herramientas disponibles, número de pasos, condiciones de finalización, entre otros) y los mecanismos de comunicación y orquestación. Asimismo, deben comprender y depurar las interacciones complejas entre agentes para mejorar su comportamiento. Sin embargo, los frameworks existentes priorizan representaciones basadas en código, lo que representa una gran barrera de entrada y dificulta la creación rápida de prototipos.Tampoco proporcionan herramientas ni métricas para la depuración y evaluación de agentes, ni plantillas reutilizables estructuradas para agilizar el proceso de creación de flujos de trabajo de agentes.

AUTOGEN STUDIO resuelve estas limitaciones, centrándonse en tres objetivos principales:

- **Prototipado Rápido:** Proporciona un entorno de desarrollo visual donde los desarrolladores pueden especificar rápidamente las configuraciones de los agentes y crear flujos de trabajo multiagente eficaces.
- **Herramientas para Desarrolladores:** Integra herramientas diseñadas para ayudar a los desarrolladores a comprender y depurar el comportamiento de los agentes, facilitando la mejora de los sistemas multiagente.
- **Plantillas Reutilizables:** Ofrece una galería de componentes, plantillas y flujos de trabajo reutilizables y compartibles, promoviendo la estandarización y las buenas prácticas en el desarrollo de sistemas multiagente.

AUTOGEN STUDIO se compone de dos módulos principales:

A. Interfaz de usuario (UI):

Una interfaz web intuitiva de arrastrar y soltar (Figura 1) que permite la creación y visualización de forma declarativa de flujos de trabajo complejos. Incluye herramientas para perfilar y depurar sesiones de agente, así como una galería de componentes multiagente reutilizables y compartibles para optimizar el desarrollo. Además, ofrece capacidades avanzadas de depuración con visualizaciones detalladas de las interacciones entre agentes, métricas de uso de herramientas, costos asociados, y estado de ejecución de las acciones. También introduce patrones de diseño generales aplicables a múltiples frameworks multiagente.

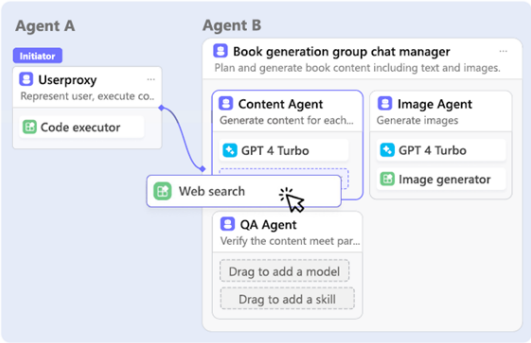

Fig. 1. AUTOGEN STUDIO: Interfaz drag and drop

B. API Backend:

API para representar agentes compatibles con LLM mediante una especificación declarativa (basada en JSON). Consta de tres componentes principales, una API web, una API de Python y una interfaz de línea de comandos. Estos componentes permiten interactuar con las clases clave del sistema, como DBManager, WorkflowManager y Profiler, facilitando la integración con aplicaciones externas y despliegues personalizados.

Adicionalmente, el artículo concluye con un caso de uso ilustrativo, en el que un ingeniero de software utiliza AUTOGEN STUDIO para desarrollar una aplicación que genera libros infantiles. Se detalla el proceso completo: desde la definición y composición del flujo de trabajo, hasta la prueba, depuración y exportación de la solución final. Este ejemplo demuestra la aplicabilidad práctica de la herramienta en escenarios reales de desarrollo multiagente.

Este ejemplo demuestra la aplicabilidad práctica de la herramienta en escenarios reales de desarrollo multiagente. En conclusión, el artículo expone como la herramienta AUTOGEN STUDIO representa una contribución significativa al ecosistema de herramientas para sistemas multiagente, al permitir la creación, depuración y evaluación de flujos de trabajo complejos sin necesidad de escribir código. Su enfoque declarativo, su interfaz visual intuitiva y su integración con herramientas modernas lo convierten en una solución accesible, versátil y potente para acelerar el desarrollo de sistemas autónomos basados en agentes.

III. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation

En el articulo se presenta el uso de AutoGen para desarrollar aplicaciones avanzadas con modelos de lenguaje de gran escala (LLM) a través de conversaciones entre múltiples agentes. AutoGen está diseñado para facilitar la creación de flujos de trabajo complejos mediante agentes personalizables y conversacionales que pueden colaborar entre sí, utilizando combinaciones de LLMs, herramientas externas e intervención humana. AutoGen introduce dos conceptos clave: agentes conversacionales y programación mediante conversaciones. Los agentes pueden mantener diálogos, ejecutar código, solicitar retroalimentación humana o utilizar herramientas externas. Esta arquitectura modular permite componer sistemas multiagente capaces de resolver tareas sofisticadas de forma autónoma o colaborativa.

Dentro de la programación conversacional AutoGen incorpora patrones de diseño para facilitar dicha programación

- **Interfaces unificadas y mecanismos de respuesta automática:**

  Todos los agentes en AutoGen comparten una interfaz estándar que incluye funciones como send, receive, generate reply lo cual permite que los agentes se comuniquen entre sí de manera coherente y modular. AutoGen introduce un mecanismo de respuesta automática por defecto donde el agente cuando recibe un mensaje, este genera y envía una respuesta automáticamente sin necesidad de un controlador centralizado.
- **Control mediante la fusión de lenguaje natural y programación:**

AutoGen permite definir el flujo de las conversaciones utilizando tanto lenguaje natural como código. Los agentes respaldados por LLMs pueden ser guiados mediante instrucciones escritas en lenguaje natural lo que permite moldear su comportamiento sin necesidad de modificar código. Tambien los desarrolladores pueden usar Python para definir condiciones de finalización, ejecutar herramientas o registrar funciones de respuesta personalizadas.

El articulo también se detallan algunas evaluaciones en escenarios prácticos usando autogen donde se incluye:

- **Resolución de problemas matemáticos:** Se implementó un sistema autónomo capaz de resolver ejercicios avanzados del dataset MATH, superando incluso a herramientas comerciales como ChatGPT con Code Interpreter.
- **Chats con recuperación aumentada:** Se desarrolló un asistente que consulta bases vectoriales y realiza recuperación interactiva cuando la información inicial es insuficiente, mejorando notablemente la precisión en respuestas.
- **Toma de decisiones simulada:** En entornos como ALFWorld, se integraron agentes que combinan planificación, ejecución y conocimiento contextual para resolver tareas en lenguaje natural.
- **Codificación colaborativa:** Se construyó un flujo de trabajo donde distintos agentes cooperan para escribir, validar y ejecutar código, reduciendo errores y mejorando la productividad.
- **Chats grupales dinámicos:** Se habilitó una conversación flexible entre múltiples agentes, con selección dinámica de turnos y roles, ideal para tareas colaborativas sin estructura fija.
- **Juegos conversacionales:** Se diseñó un entorno lúdico como el ajedrez por lenguaje natural, donde los agentes validan jugadas, interpretan reglas y permiten interacción fluida entre humanos y modelos.

IV. Platform for Multiagent Application Development Incorporating Accurate Communications Modeling

Este artículo presenta una plataforma para el desarrollo, prueba y validación de aplicaciones multiagente distribuidas, con énfasis en la simulación precisa de redes de comunicación. Esta plataforma combina el framework de desarrollo multiagente *JADE* (Java Agent Development) con el simulador de redes *OPNET*, con el objetivo de abordar de forma rigurosa los desafíos del desarrollo de aplicaciones distribuidas y críticas en tiempo real, especialmente en sistemas de protección de redes eléctricas inteligentes (smart grids).

El principal problema que aborda esta investigación es la falta de herramientas que permitan validar, antes del despliegue, el comportamiento de sistemas multiagente considerando con exactitud el rendimiento de la red de comunicaciones. En aplicaciones distribuidas y sensibles al tiempo, como la protección eléctrica, los retrasos en la comunicación pueden comprometer la seguridad y confiabilidad del sistema.

La plataforma propuesta integra tres componentes principales:

- **JADE**: entorno de ejecución de agentes compatible con el estándar FIPA (Foundation for Intelligent Physical Agents).
- **OPNET**: simulador de redes de comunicaciones que permite modelar con precisión el comportamiento de infraestructuras complejas.
- **RTI (Runtime Infrastructure)**: middleware de sincronización basado en el estándar HLA (High-Level Architecture).

A. Integración de JADE y OPNET mediante HLA

La propuesta se centra en la unificación del framework JADE, ampliamente utilizado para el desarrollo de sistemas multiagente, y el simulador de redes OPNET. Para lograr esta integración, se extendió JADE con capacidades de simulación de eventos discretos, introduciendo una clase de agente denominada `FedAgent`. Este agente, junto con un agente `RTI`, coordina la sincronización temporal y el enrutamiento de mensajes entre el entorno de ejecución de la aplicación y el simulador. Por otro lado, OPNET fue modificado para incorporar agentes genéricos que representan adecuadamente los nodos de red asociados a los agentes de JADE. La integración entre ambos entornos se realiza siguiendo el estándar HLA, lo que garantiza una ejecución sincronizada y coherente.

B. Modelo y Arquitectura del Sistema

La arquitectura se basa en una ejecución coordinada en la que JADE corre el código real de la aplicación multiagente y OPNET simula la red de comunicaciones sobre la que estos agentes operan. El agente RTI actúa como puente entre ambos, gestionando el avance del tiempo en el simulador y asegurando que los eventos y mensajes se procesen de forma coherente. Esta estructura permite que el mismo código desarrollado en JADE se utilice directamente en el entorno real, sin necesidad de modificaciones manuales.

C. Caso de Estudio: Supervisión Remota de Relés en Smart Grid

Para validar la eficacia de la plataforma, los autores implementaron un caso de estudio centrado en la supervisión remota de relés de protección de zona 3 en redes eléctricas. En este escenario, cada relé está asociado a un agente (*RA*) que, al detectar una posible falla, se comunica con otros agentes para confirmar o descartar la presencia de una anomalía antes de ejecutar una acción crítica como la desconexión de una línea. Se comparan dos arquitecturas de comunicación:

- **Cliente-servidor (C/S)**: un agente central (*DMA*) coordina la toma de decisiones. Cuando un RA detecta una falla, consulta al DMA, quien recopila información de los demás agentes y decide la acción a seguir.

- **Peer-to-peer (P2P)**: los agentes se comunican directamente entre sí sin intervención del DMA, lo que permite una respuesta potencialmente más rápida y resiliente.

El sistema fue evaluado sobre un modelo de red basado en el sistema IEEE de 39 buses, que simula subestaciones, routers, switches y enlaces T1. Se definieron tres escenarios de red:

1) Sin tráfico de fondo.
2) Con tráfico de fondo elevado.
3) Con estrategia de calidad de servicio (QoS) que prioriza el tráfico de agentes.

Las simulaciones demostraron que el rendimiento del sistema está fuertemente influenciado por la arquitectura de comunicación, el protocolo de transporte (TCP/UDP), el tamaño de los mensajes y la política de QoS. En escenarios sin QoS, el uso de TCP introdujo retrasos significativos. Por el contrario, la arquitectura P2P con UDP y QoS logró tiempos de respuesta más rápidos y estables.

El estudio también analizó el impacto de fallos en enlaces de red y la ubicación óptima del *DMA* para minimizar los tiempos de decisión. Finalmente, se destaca que el código de los agentes, una vez validado en este entorno federado, puede desplegarse directamente en el sistema real, lo cual representa un avance significativo en el ciclo de vida del desarrollo de aplicaciones distribuidas.

Esta investigación demuestra que una plataforma federada que integra simulación de redes y entornos multiagente puede facilitar el desarrollo de sistemas robustos y confiables, especialmente en dominios críticos como las redes eléctricas inteligentes.

V. Exploration of LLM Multi-Agent Application Implementation Based on LangGraph+CrewAI

Este artículo presenta una exploración de LangGraph + CrewAI, una combinación de frameworks diseñada para facilitar la implementación de sistemas multiagente basados en modelos de lenguaje. Esta integración permite construir flujos de trabajo complejos mediante el uso coordinado de agentes inteligentes logrando capacidades avanzadas de control, colaboración y ejecución autónoma de tareas. En esta combinación LangGraph aporta una estructura de control granular y persistencia en la memoria de los agentes, mientras que CrewAI potencia la cooperación mediante la asignación inteligente de roles y tareas dentro de equipos de agentes.

El desarrollo de aplicaciones con agentes LLM exige diseñar arquitecturas complejas, dividir tareas, gestionar estados conversacionales y coordinar flujos de información entre múltiples entidades. Además, los desarrolladores deben garantizar la trazabilidad, la robustez y la eficiencia en estos sistemas distribuidos. Sin embargo la mayoría de los frameworks disponibles carecen de herramientas específicas para integrar control estructurado, colaboración entre agentes y gestión de tareas en un solo entorno.

LangGraph + CrewAI efrenta las limitaciones de la mayoria de framewroks mediante un enfoque conjunto que se centra en dos conceptos importantes:

- **Control estructurado con LangGraph:** Proporciona un marco basado en grafos que facilita el control preciso del flujo de trabajo entre agentes, soportando flujos secuenciales, jerárquicos o cíclicos, con persistencia de estado y lógica reutilizable.

- **Colaboración organizada con CrewAI:** Permite definir agentes con roles específicos, delegación automática de tareas y ejecución modular, todo con compatibilidad para herramientas externas y APIs como OpenAI u Ollama.

El artículo presenta además un caso de uso donde se ilustra la aplicabilidad de esta combinación. En este caso de aplicación, se desarrolló un sistema inteligente para la gestión automática de órdenes de trabajo utilizando el marco combinado LangGraph + CrewAI. El objetivo era agilizar el procesamiento de tickets mediante la colaboración entre múltiples agentes, cada uno con responsabilidades especializadas.

El sistema se encarga de analizar el contenido textual de los tickets, comprendiendo su contexto, prioridad y posibles acciones requeridas. Para ello, se emplean modelos de embedding que permiten representar el texto de los tickets en un espacio vectorial y buscar información relacionada de manera eficiente.

Mediante CrewAI, se definieron agentes con funciones específicas donde algunos se encargan de la lectura y comprensión del texto, otros del análisis semántico y categorización, y otros de reenviar el ticket al área o herramienta correspondiente. La asignación de tareas y roles se realiza automáticamente lo que logra asegurar que cada paso del flujo esté cubierto por el agente más adecuado.

LangGraph facilita el diseño del flujo de procesamiento como un grafo controlado. Esto permite que las decisiones y transiciones entre etapas se visualicen y controlen fácilmente, integrando persistencia de datos y trazabilidad del estado de cada ticket.

VI. Conclusión

- El desarrollo de sistemas multiagente (MAS) plantea una serie de desafíos técnicos relacionados con la coordinación, comunicación, depuración y validación de agentes en entornos distribuidos. En este artículo se ha presentado un análisis de cuatro artículos de herramientas diseñadas para facilitar este proceso e implementación en casos reales.

- Entre las plataformas estudiadas, *AUTOGEN STUDIO* destaca por su enfoque sin código, que permite a los

desarrolladores construir flujos de trabajo multiagente de manera visual y declarativa. Por otro lado, la plataforma basada en la integración de *JADE* y *OPNET* ofrece un entorno para validar aplicaciones MAS bajo condiciones de red simuladas con alta fidelidad.

- AutoGen es una herramienta útil para el desarrollo de sistemas multiagente impulsados por modelos LLM porque permite diseñar flujos de trabajo conversacionales complejos mediante agentes personalizables que colaboran entre sí, ejecutan tareas, intercambian mensajes y se integran con herramientas externas o intervención humana. Gracias a sus interfaces unificadas AutoGen facilita la creación de soluciones modulares, eficientes y adaptables. Esta flexibilidad ha sido validada en múltiples escenarios prácticos, como la resolución de problemas matemáticos, recuperación aumentada de información, toma de decisiones en entornos simulados, codificación colaborativa, chats grupales dinámicos y juegos conversacionales, demostrando su potencial para acelerar el desarrollo de aplicaciones inteligentes basadas en agentes.

- Conocer el ecosistema de herramientas disponibles es fundamental para seleccionar la solución adecuada a cada contexto. Este tipo de estudios contribuye a tomar decisiones informadas y a fomentar el desarrollo de aplicaciones multiagente más accesibles, eficientes y confiables.

- La combinación de frameworks ayuda mucho en el desarrollo de sistemas multiagente complejos al permitir integrar capacidades especializadas de distintos entornos en una sola solución coherente y flexible. Este enfoque facilita el diseño de flujos de trabajo robustos donde se puede combinar el control estructurado de procesos con mecanismos de colaboración inteligente entre agentes ayudando a optimizar la ejecución de tareas distribuidas.

REFERENCES

[1] M. Wooldridge, *An introduction to multiagent systems*. John wiley & sons, 2009.

[2] S. D. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, and T. Funabashi, "Multi-agent systems for power engineering applications—part i: Concepts, approaches, and technical challenges," *IEEE Transactions on Power systems*, vol. 22, no. 4, pp. 1743–1752, 2007.

[3] F. Perkonigg, D. Brujic, and M. Ristic, "Platform for multiagent application development incorporating accurate communications modeling," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 728–736, 2015.

[4] F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with jade," in *Intelligent Agents VII Agent Theories Architectures and Languages: 7th International Workshop, ATAL 2000 Boston, MA, USA, July 7–9, 2000 Proceedings 7*, pp. 89–103, Springer, 2001.

[5] V. Dibia, J. Chen, G. Bansal, S. Syed, A. Fourney, E. Zhu, C. Wang, and S. Amershi, "Autogen studio: A no-code developer tool for building and debugging multi-agent systems," 2024.

[6] F. Perkonigg, D. Brujic, and M. Ristic, "Platform for multiagent application development incorporating accurate communications modeling," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 728–736, 2015.