Descubrimiento y diseño de microservicios asistido por IA a partir de requisitos textuales

Dr. Daniel Narváez
Doctorado en Informática – Universidad Abierta Interamericana (UAI)
CAETI – UAI
Director: Dr. Gustavo Rossi | Co-director: Dr. Alejandro Fernández
Año académico: 2026 | Fecha de exposición: 24/02/2026

Contexto. En escenarios *greenfield*, el diseño de microservicios parte de requisitos textuales ambiguos, incrementales y sin evidencia estructural explícita. Aunque los LLMs pueden generar propuestas arquitectónicas plausibles, esa plausibilidad semántica no garantiza corrección estructural ni calidad de diseño [1, 2, 3].

Tensión central. La dificultad consiste en evitar descomposiciones que conduzcan a monolitos distribuidos, alto acoplamiento, baja cohesión y deuda arquitectónica temprana [4]. En este contexto, la tesis parte de una idea clave: **lo lingüísticamente convincente no siempre coincide con lo arquitectónicamente válido**.

Pregunta de investigación. *¿Cómo generar propuestas de microservicios a partir de requisitos textuales y, al mismo tiempo, garantizar su corrección estructural mediante verificación formal?*

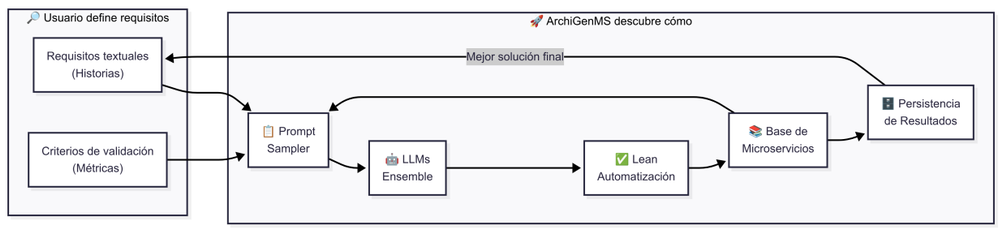
Idea central. ArchiGenMS se concibe como un **marco neuro-simbólico evolutivo** para el descubrimiento automático de microservicios. Su hipótesis central es que un enfoque puramente conexionista resulta insuficiente para diseño arquitectónico riguroso: un LLM puede capturar semántica de dominio, pero no ofrece por sí solo garantías estructurales explícitas.

Componente neural. El LLM genera candidatos arquitectónicos a partir de requisitos textuales, identificando entidades, operaciones y posibles límites funcionales.

Componente simbólico. Lean 4 verifica invariantes estructurales y descarta topologías inválidas mediante un núcleo formal determinista.

Componente evolutivo. La búsqueda reutiliza genotipos válidos y mejora progresivamente la calidad arquitectónica.

Interpretación. En este esquema, Lean 4 no opera solo como filtro binario. También actúa como un **oráculo simbólico** que restringe el espacio de búsqueda, rechaza regiones estructuralmente inválidas y favorece la reutilización de candidatos admisibles [5, 6, 7, 8].



Lectura de la figura. Figura 1 — Arquitectura conceptual del ciclo *generate–verify–evolve*. La imagen sintetiza la lógica interna de ArchiGenMS. Primero, el LLM produce un *genotipo* arquitectónico a partir de historias de usuario. Luego, Lean 4 actúa como **oráculo simbólico**, verificando invariantes estructurales. Finalmente, la etapa evolutiva reutiliza candidatos válidos y optimiza su calidad arquitectónica.

Significado. La figura no representa solo un flujo técnico, sino la interacción entre **generación estocástica**, **validación determinista** y **refinamiento evolutivo** [6, 9].

Hallazgo principal. La verificación formal estabiliza la búsqueda y permite que la optimización evolutiva mejore la calidad sin abandonar la validez estructural.

Indicador	Resultado
Escenarios evaluados	22
Reducción del fitness tras 5 generaciones	**44 %**
Convergencia del proceso evolutivo	Sí
Caso destacado	**g24-unibath**
Acoplamiento máximo en g24-unibath	**1**

Función objetivo. La optimización se guía mediante una función agregada sobre métricas computables en Lean 4:

$$Fitness(\mathcal{A}) = LCOM_{avg}(\mathcal{A}) + SGM_{max}(\mathcal{A}) + DispParams_{sum}(\mathcal{A}) + CouplingOut_{max}(\mathcal{A})$$

Menores valores indican mejor calidad arquitectónica.

Interpretación. Los resultados muestran que la verificación formal no bloquea la exploración, sino que estabiliza la búsqueda y favorece la convergencia sobre arquitecturas estructuralmente admisibles.

Representación algebraica. La arquitectura candidata propuesta por el LLM se formaliza como una estructura algebraica:

$$\mathcal{A} = \langle \mathcal{S}, \mathcal{C} \rangle, \qquad s = \langle name, Ops \rangle, \qquad o = \langle n, P \rangle$$

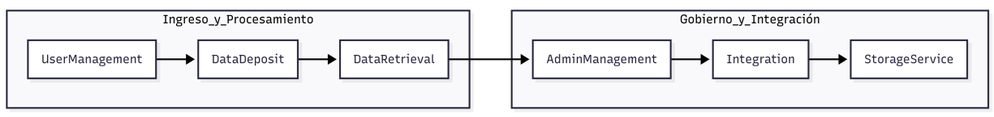
donde $\mathcal{S}$ es el conjunto de microservicios, $\mathcal{C}$ el conjunto de llamadas dirigidas entre servicios, Ops el conjunto de operaciones y P el conjunto de parámetros.

Invariantes estructurales. Sobre esta estructura, Lean 4 verifica un núcleo mínimo de invariantes:

$$\forall (u,v) \in \mathcal{C},\ u \in \mathcal{S}_{names}, \qquad \forall (u,v) \in \mathcal{C},\ v \in \mathcal{S}_{names}, \qquad \forall (u,v) \in \mathcal{C},\ u \neq v$$

Estos axiomas garantizan integridad referencial y anti-reflexividad.

Evaluación de calidad. A partir de ellos, la optimización se realiza únicamente sobre arquitecturas válidas, mientras la calidad se cuantifica mediante métricas formales como **LCOM**, **SGM**, **DispParams** y **CouplingOut** [10, 11].



Lectura de la figura. Figura 2 — Ejemplo de topología válida descubierta automáticamente. La imagen muestra una arquitectura concreta obtenida por ArchiGenMS para el caso `g24-unibath`.

Qué demuestra. Representa una instancia donde el ciclo neuro-simbólico produce una topología con **bajo acoplamiento saliente**, ausencia de auto-llamadas y separación explícita de responsabilidades.

Relevancia. La figura ejemplifica cómo el marco no se limita a generar una propuesta plausible, sino que converge hacia una configuración estructuralmente admisible y arquitectónicamente interpretable.

Extensiones inmediatas. Las extensiones más naturales del trabajo son:

- enriquecimiento del núcleo formal con invariantes arquitectónicos más expresivos;
- incorporación de **NFRs dinámicos** y propiedades temporales;
- definición de métricas formales más ricas para modularidad y refinamiento;
- exploración de modelos algebraicos más profundos para composición y descomposición de servicios.

Proyección teórica. A mediano plazo, una dirección prometedora consiste en estudiar abstracciones de inspiración categórica para representar servicios, dependencias y transformaciones composicionales de forma más rigurosa, fortaleciendo el puente entre IA generativa, verificación formal y teoría matemática de arquitecturas distribuidas.

[1] S. Newman, *Building microservices: designing fine-grained systems.* .ºReilly Media, Inc.", 2021.

[2] D. Narváez, N. Battaglia, A. Fernández, and G. Rossi, "Designing microservices using ai: A systematic literature review," *Software*, vol. 4, no. 1, p. 6, 2025.

[3] J. R. A. Pereira, D. Albuquerque, M. Perkusich, G. Rodríguez, J. A. Díaz-Pace, K. Gorgônio, and A. Perkusich, "Toward generating microservice architectures from textual requirements with large language models," in *Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS)*, pp. 79–89, SBC, 2025.

[4] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow," *Present and ulterior software engineering*, pp. 195–216, 2017.

[5] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, pp. 75–105, 2004.

[6] L. d. Moura and S. Ullrich, "The lean 4 theorem prover and programming language," in *International Conference on Automated Deduction*, pp. 625–635, Springer, 2021.

[7] E. M. Clarke and J. M. Wing, "Formal methods: State of the art and future directions," *ACM Computing Surveys (CSUR)*, vol. 28, no. 4, pp. 626–643, 1996.

[8] L. Belzner, T. Gabor, and M. Wirsing, "Large language model assisted software engineering: prospects, challenges, and a case study," in *International conference on bridging the gap between AI and reality*, pp. 355–374, Springer, 2023.

[9] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, pp. 1–61, 2012.

[10] O. Al-Debagy and P. Martinek, "A metrics framework for evaluating microservices architecture designs," *Journal of Web Engineering*, vol. 19, no. 3–4, pp. 341–370, 2020.

[11] D. Taibi and K. Systä, "A decomposition and metric-based evaluation framework for microservices," 2019.