DOI: 10.4206/retd.2023.n2-05

ARQUITECTURA DE SOFTWARE: UN MEDIO PARA LA TRANSDISCIPLINA EN TIEMPOS CAMBIANTES

Cristian Olivares Rodríguez
Dr. en Ingeniería
Investigador Instituto Informática
Facultad de Ciencias de la Ingeniería UACh

Martha Vidal Sepúlveda
Mg. Desarrollo Curricular
Docente UACh

Palabras clave: Arquitectura de software, inteligencia artificial, competencias digitales

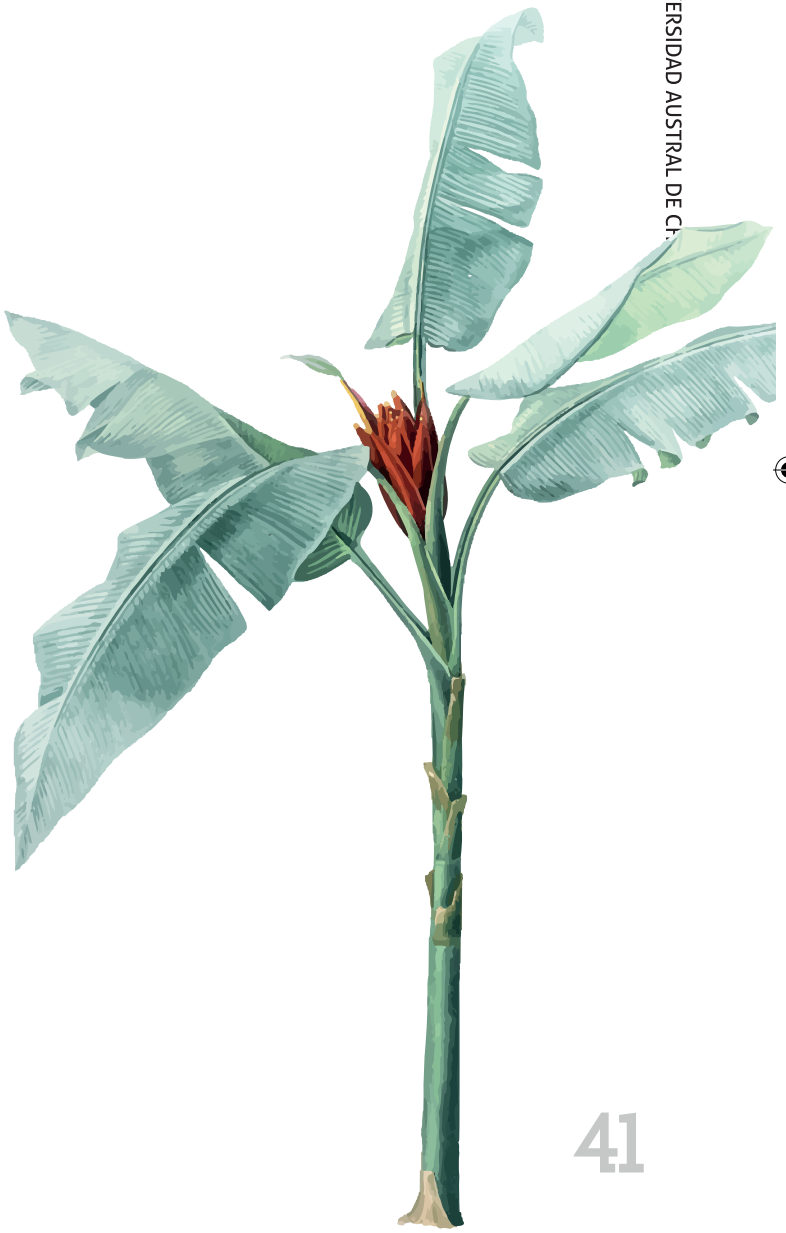
Actualmente, convivimos con gran variedad de plataformas que condicionan nuestras relaciones sociales y que son fabricadas a partir de métodos y herramientas propias de la industria del software (Sommerville, 2011). De manera más concreta, son elaboradas a partir de un proceso sistemático de desarrollo de software a través de la aplicación de enfoques, técnicas y herramientas que permitan lograr los objetivos para los cuales ha sido pensado el producto de software (Pressman, 2010). Así, la industria del software busca elaborar productos de calidad en relación con las necesidades de los clientes, las demandas actuales de la sociedad y la rapidez con la que emergen nuevas necesidades, exigiendo a los profesionales de las fábricas de software la capacidad de afrontar apropiadamente los problemas esenciales y accidentales que se presentan regularmente en esta industria (Brooks, 1987). Uno de los principales problemas esenciales del software es su invisibilidad; un pequeño cambio de uno de sus componentes puede hacerlo variar drásticamente o, peor aún, dificulta recorrerlo completamente para asegurar la correcta interacción entre sus componentes.

Desde la discusión propuesta por Brooks en la década de los ochenta acerca de la similitud de los proyectos de software con los hombres lobo, quienes formamos parte de esta disciplina hemos trabajado en la validación de nuevas balas de plata a lo largo de todo el ciclo de vida del producto de software, desde el análisis de las necesidades de los usuarios (Rojas et al., 2022) hasta la mantención de los productos (Ozkaya, 2019). Estas propuestas tienen el propósito de mantener bajo control los problemas esenciales y accidentales del producto a medida que avanza el proyecto. En este camino hacia el equilibrio entre control del proceso de construcción y rapidez en el desarrollo de software se han liberado frameworks de desarrollo, entre los cuales encontramos las propuestas de Google (Cantelon et al., 2014) y Facebook (Fedosejev, 2015). Estos frameworks cuentan con piezas de software

que pueden ser integradas en nuestros productos de software eliminando la necesidad de volver a escribirlas completamente, pero incrementando la dependencia en piezas de software desarrolladas por terceros.

La arquitectura de software, en tanto bala de plata, busca orquestar piezas de software de manera sistemática a través de patrones arquitectónicos, permitiendo que los diferentes interesados en el producto de software puedan razonar acerca de las relaciones y las responsabilidades de las estructuras que la componen a través de un lenguaje común (Bass et al., 2003). Sin embargo, los interesados tienen códigos e intereses muy dispares debido a sus características profesionales, por lo que se hace necesaria una nueva bala de plata que permita abordar la complejidad en la discusión sobre el lenguaje del dominio del problema sin perder los vínculos con el lenguaje técnico. Para ello, Evans (2004) ha propuesto una metodología de análisis del dominio que sistematiza la conformación iterativa e incremental de abstracciones conceptuales a partir del análisis de los códigos propios del dominio del problema, contribuyendo con la organización de los componentes arquitectónicos de la solución de software.

De este modo, en el actual proceso de desarrollo de software (fábrica) la arquitectura de software cobra mayor



relevancia, ya que en los productos de hoy conviven piezas de software externas con piezas desarrolladas específicamente para el dominio del problema a resolver. Mientras, el desarrollo dirigido por el dominio propuesto por Evans (2004) brinda la oportunidad de dar coherencia a los modelos arquitectónicos en relación con las entidades presentes en el lenguaje de los usuarios. En consecuencia, los modelos arquitectónicos compuestos por entidades abstractas diseñadas por los equipos de desarrollo son responsables de la ejecución de tareas específicas, para lo cual implementan interacciones abstractas con piezas de software tanto externas como específicas del dominio. Este artefacto de diseño abstracto del producto de software a partir de un lenguaje unificado (UML) permite que los interesados puedan discutir acerca de las características del producto de software, ya que las reglas y dinámicas del contexto disciplinar quedan representadas visualmente a partir de entidades.

En este contexto, el desarrollo de productos científico-tecnológicos avanza desde la co-construcción de la arquitectura de software como medio de interacción disciplinar, permitiendo el diálogo desde la abstracción y la complejización de la realidad observada. Si bien los estudios disciplinares han buscado la conformación de verdades sobre las cuales avanzar en la comprensión de nuestro entorno a partir de constructos propios de cada disciplina (Calvo, 2019), en la actualidad, la comunidad científica y las políticas públicas buscan la conformación de verdades a partir de la complejización de los problemas

desde el tratamiento transdisciplinar de los estudios sobre los cuales se espera registrar observaciones de los objetos de estudio para la validación de las preguntas de investigación, pero también para la observación sistemática de la realidad. Así, los productos de software se consolidan como espacios de comprensión de las dinámicas de subjetivación por medio de la interacción hombre-máquina. En este sentido, cobra mayor relevancia la arquitectura de software, siendo el artefacto abstracto que se consolida como un dispositivo de vigilancia de las conformaciones de subjetividades del mundo observado, ahora, desde la integración disciplinar en el proceso de co-construcción de los productos de software científico-tecnológicos.

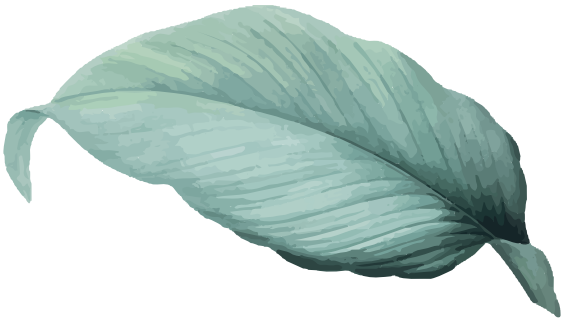
Inteligencia artificial generativa, ¿la nueva bala de plata?

La inteligencia artificial estudia métodos que permitan a los computadores resolver problemas de la misma manera en que podría hacerlo un humano. A la fecha, esta disciplina se ha enfocado en el desarrollo de productos de software a partir de cuatro perspectivas: 1) pensar como humano; 2) pensar racionalmente; 3) actuar como humano; y 4) actuar racionalmente (Russell y Norvig, 2004). Por ello, la inteligencia artificial implicaría –conforme a la primera perspectiva– el desarrollo de herramientas que sean capaces de pensar en el problema que se intenta resolver para proponer las estructuras necesarias para solucionarlo. Por su parte, el segundo enfoque estaría asociado con la elaboración de herramientas que piensen y apliquen los métodos correctos para la conformación de la solución apropiada al problema que se intenta resolver. Luego, el tercer enfoque implicaría la existencia de herramientas que imiten la forma en la cual los humanos construyen

productos de software para resolver problemas. El último enfoque, por tanto, se vincula con herramientas que actúan racionalmente en la construcción de las abstracciones que resuelven correctamente los problemas. En cualquiera de los casos, la industria del software se vería beneficiada con el desarrollo de herramientas que permitan construir abstracciones asociadas con las entidades del dominio a partir de un análisis correcto del dominio y la comprensión apropiada del problema.

A pesar de los avances de la inteligencia artificial, autores como Brooks (1987) la catalogan como un intento fallido de bala de plata para el proceso de desarrollo de software, debido a que la resolución de un problema por medio de un producto de software requiere la elaboración de abstracciones a través de un proceso creativo. No obstante, en los meses recientes hemos sido testigos de la eclosión de herramientas capaces de generar abstracciones a partir de un conjunto de instrucciones o prompts entregados por usuarios a través de una interfaz, a primera vista, simple y efectiva. De esta manera, la inteligencia artificial generativa ha llegado a nuestros hogares por medio de diversas plataformas, tales como Chat GPT de Open AI o Bard de Google. Estos motores conversacionales son capaces de generar textos a partir de prompts escritos en el idioma del interlocutor a partir de modelos del lenguaje entrenados con inmensas cantidades de datos. Pero, no solo son competentes produciendo distintos tipos de textos, incluso creativos, sino que también son capaces de generar piezas de código o, incluso, propuestas de arquitecturas de software que, fácilmente, pueden ser integradas como piezas de software por parte de desarrolladores expertos.

En este contexto, la inteligencia artificial generativa



parece ser la bala de plata esperada en la industria del software, aquella que nos permita abordar la complejidad en la creación de abstracciones que sean rápida y correctamente integradas en nuestros productos de software. Sin embargo, los problemas actuales son cada vez más complejos y requieren del involucramiento de una amplia diversidad de interesados provenientes de varias disciplinas para alcanzar una comprensión holística que nos lleve hacia la conformación de nuevas abstracciones y reglas asociadas con nuestra comprensión del entorno en el que vivimos.

Nuevas competencias digitales

Actualmente, somos testigos de la denominada guerra de plataformas (Scolari, 2022) en la cual la industria del software participa activamente para captar adeptos por medio de nuevas funcionalidades que prometen facilitar nuestro trabajo. Pero, también surgen cuestionamientos sobre los riesgos éticos del uso de la inteligencia artificial generativa. Chomsky (2023) plantea que la ausencia de pensamiento moral en la IA le permitiría falsear respuestas. Asimismo, esta amoralidad podría ser fuente propicia para la exacerbación de los sesgos presentes en la web (Kroll, 2015; Caplan y Boyd, 2016; Vidal-Sepúlveda et al., 2023). En esta discusión, se dibuja un usuario ingenuo e influenciable en tanto la inteligencia artificial se retrata como eficaz y seductora. Conforme a esta lógica, la humanidad estaría en riesgo de ser sometida bajo su influencia. Sin embargo, a la fecha, los usuarios se han mostrado cautelosos en el uso de la inteligencia artificial; quizás la tan anunciada masificación se ha pausado debido a las polémicas comunicacionales, o bien por la carencia de competencias digitales.

La competencia digital implica el uso seguro y crítico de las tecnologías de la sociedad de la información en los distintos ámbitos profesionales (Ferrari y Punie, 2013). Esta definición de competencia digital evoca la competencia crítica, una habilidad humana, pero no por ello automática. No se nace crítico, sino que es una competencia que se adquiere con la práctica constante. De acuerdo con este planteamiento, la competencia digital implica la formación crítica de las personas, por lo que para beneficiarse de la inteligencia artificial no solo es necesario contar con habilidades técnicas; para hacer de ella un uso efectivo y ético es necesario desarrollar la competencia crítica.

De este modo, la integración de la inteligencia artificial generativa en las prácticas de desarrollo de software científico-tecnológico de productos que contribuyan

con la conformación de verdades con un enfoque transdisciplinar impone la adquisición de nuevas competencias digitales. Pasaría, entonces, la arquitectura de software a ser el resultado de la interacción disciplinar propia de los estudios transdisciplinares con una máquina capaz de generar (proponer) entidades abstractas. Es así como las dinámicas de co-creación soportadas por máquinas inteligentes se consolidarán a partir de la efectividad de los científicos respecto de la generación de instrucciones precisas. Cabe entonces preguntarse ¿cuáles son las nuevas competencias digitales necesarias para el desarrollo de productos de software científico-tecnológicos? ¿Cuáles son los códigos lingüísticos necesarios para conformar una arquitectura de software correcta y comprensible por los diferentes campos disciplinares? Es claro que las máquinas inteligentes contribuyen con los procesos creativos y de producción a partir de la generación de insumos textuales y/o visuales, los que pueden ser refinados a partir de métodos científico-tecnológicos rigurosos. Entonces, el desafío radica en la consolidación de las nuevas competencias digitales que permitan la co-creación de entidades abstractas plasmadas en una arquitectura de software de quienes trabajamos en la generación de conocimiento en interacción con otros campos disciplinares.

Referencias

Bass, L., Clements, P., & Kazman, R. (2003). Software architecture in practice. Addison-Wesley Professional.

Brooks, F. P. (1987). Essence and accidents of software engineering. IEEE computer, 20(4), 10-19.

Calvo, M. (2019). Pensamiento complejo y transdisciplina. Sophia, colección de Filosofía de la Educación, 26(1), pp. 307-326.

Cantelon, M., Harter, M., Holowaychuk, T. J., y Rajlich, N. (2014). Node.js in Action (pp. 17-20). Greenwich: Manning.

Caplan, R., y Boyd, D. (2016). Who controls the public sphere in an era of algorithms. Mediation, Automation, Power, 1-19.

Chomsky, N. (31 de enero 2023). Noam Chomsky. The false promise of ChatGPT. New York Times. https://www.nytimes.com/2023/03/08/opinion/noam-chomsky-chatgpt-ai.html

Evans, E. (2004). Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional.

Fedosejev, A. (2015). React.js essentials. Packt Publishing Ltd.

Kroll, J. A. (2015). Accountable algorithms. (Doctoral dissertation, Princeton University).

Rojas, L., Olivares-Rodríguez, C., Alvarez, C., y Campos, P. G. (2022). OurRank: A Software Requirements Prioritization Method Based on Qualitative Assessment and Cost-Benefit Prediction. IEEE Access, 10, 131772-131787.

Russell, S. J., y Norvig, P. (2004). Inteligencia artificial: un enfoque moderno (No. 04; Q335, R8y 2004.).

Scolari, C. A. (2022). La guerra de las plataformas. Anagrama.

Sommerville, I. (2011). Software engineering (ed.). America: Pearson Education Inc.

Ozkaya, I. (2019). Are devops and automation our next silver bullet?. IEEE Software, 36(4), 3-95.
Pressman, R. S. (2010). A practitioner's approach. Software Engineering, 2, 41-42.

Vidal-Sepúlveda, M; Olivares-Rodríguez C. y Cárcamo-Ulloa, L. (2023, en prensa). "Básicamente, todo lo puedes encontrar ahí": Creencias, sesgos y estrategias de búsqueda de estudiantes universitarios en los motores de búsqueda. Revista Dígitos.