CognitivaTech: Ingeniería de Software Inteligente y Sistemas Adaptativos. 2025; 2(1):20
doi: 10.63688/cognitivatech.v2.i1.20

ORIGINAL

Multi-objective software remodularization optimization using genetic algorithms

Optimización de la remodularización de software mediante algoritmos genéticos multiobjetivo

Gustavo Adolfo Hernández Rosado[1]

[1] Universidad Espíritu Santo. Guayaquil, Ecuador.

Recibido: 9-01-2025 | Revisado: 22-04-2025 | Aceptado: 27-03-2025 | Publicado: 23-04-2025

Autor de correspondencia: Gustavo Adolfo Hernández Rosado

ABSTRACT

The increasing complexity of software systems has intensified the need for strategies that improve their internal structure without altering functionality. In this context, remodularization emerges as a key practice to reorganize software components in order to enhance maintainability, scalability, and comprehensibility. This study proposes a multi-objective optimization approach based on genetic algorithms to improve the modular quality of complex systems. The method focuses on minimizing coupling and maximizing cohesion as primary quality criteria. To achieve this, both static and dynamic analysis techniques are employed to reconstruct the existing architecture and evaluate alternative modular configurations. The results show significant improvements in software structure, including reduced coupling and increased cohesion, as well as a better distribution of responsibilities among modules. Furthermore, findings indicate that there is no single optimal solution, but rather multiple trade-off alternatives between the considered objectives. Expert validation supports the feasibility and usefulness of the generated recommendations. Overall, the study demonstrates that integrating evolutionary optimization techniques with software analysis provides an effective tool to support maintenance and evolution processes in complex systems, particularly in research software contexts.

Keywords: remodularization; genetic algorithms; cohesion;coupling.

RESUMEN

La creciente complejidad de los sistemas de software ha incrementado la necesidad de aplicar estrategias que mejoren su estructura interna sin afectar su funcionalidad. En este contexto, la remodularización se presenta como una práctica clave para reorganizar los componentes del software con el fin de optimizar su mantenibilidad, escalabilidad y comprensión. Esta investigación propone un enfoque basado en optimización multiobjetivo mediante algoritmos genéticos para mejorar la calidad modular de sistemas complejos. El método se fundamenta en la minimización del acoplamiento y la maximización de la cohesión como criterios principales de calidad. Para ello, se emplean técnicas de análisis estático y dinámico que permiten reconstruir la arquitectura existente y evaluar diferentes configuraciones modulares. Los resultados muestran mejoras significativas en la estructura del software, evidenciando reducciones en el acoplamiento y aumentos en la cohesión, así como una mejor distribución de responsabilidades entre módulos. Además, se identifica que no existe una única solución óptima, sino múltiples alternativas que representan distintos compromisos entre los objetivos considerados. La validación con expertos confirma la viabilidad y utilidad de las recomendaciones generadas. En conjunto, el estudio demuestra que la integración de técnicas de optimización evolutiva y análisis de software constituye una herramienta efectiva para apoyar procesos de mantenimiento y evolución en sistemas complejos, especialmente en el ámbito del software científico.

Palabras clave: remodularización; algoritmos genéticos; cohesión; acoplamiento.

INTRODUCCION

La evolución constante de los sistemas de software ha puesto en evidencia la necesidad de adaptar sus estructuras internas para responder a nuevas demandas de calidad, escalabilidad y mantenibilidad. En este contexto, la remodularización se consolida como una práctica clave dentro de la ingeniería de software, al centrarse en la reorganización de los componentes modulares sin alterar el comportamiento funcional del sistema(Hernández, D., & Sánchez, M. L. J. C., 2024). Este proceso busca mejorar la claridad estructural del código, facilitar su comprensión y mantenimiento, y optimizar la distribución de responsabilidades entre módulos, lo que resulta especialmente relevante en sistemas complejos y de larga vida útil, donde la deuda técnica tiende a incrementarse con el tiempo(Candela et al., 2016; Verdecchia et al., 2020; Hasselbring, 2018).

Tradicionalmente, la calidad modular se ha evaluado a través de métricas como el acoplamiento y la cohesión, consideradas pilares fundamentales para lograr arquitecturas de software robustas y flexibles(Rivera, 2021). Sin embargo, optimizar simultáneamente estos atributos implica enfrentar objetivos potencialmente conflictivos, lo que ha motivado la incorporación de enfoques de optimización multiobjetivo. En este sentido, los algoritmos genéticos han emergido como una estrategia eficaz para explorar múltiples configuraciones arquitectónicas y generar recomendaciones de remodularización que equilibran dichos criterios(Mkaoueret al., 2015; Cortellessa et al., 2025). Estos enfoques permiten no solo automatizar el proceso de reorganización, sino también adaptarlo a distintos contextos y restricciones propias de cada sistema, incluyendo la gestión de la deuda técnica y la evolución continua del software.

El desafío se intensifica en el ámbito del software científico, particularmente en sistemas de gran escala como aquellos utilizados en investigación interdisciplinaria, donde la complejidad estructural y la necesidad de reproducibilidad exigen prácticas de ingeniería más rigurosas. En este dominio, la ingeniería de software orientada a la investigación ha cobrado relevancia al enfatizar la importancia de arquitecturas sostenibles, mantenibles y comprensibles(Feldereret al., 2025;Druskatet al., 2025). La remodularización, en este contexto, no solo mejora la organización interna del software, sino que también contribuye a facilitar la colaboración entre equipos multidisciplinarios y la integración de nuevos componentes.

Adicionalmente, en los últimos años se ha reconocido que la calidad del software no depende únicamente de factores técnicos, sino también de aspectos sociales y humanos presentes en los procesos de desarrollo. Estudios recientes han evidenciado que factores como la diversidad de género, la inclusión y la participación en comunidades de desarrollo influyen en la calidad del código, la revisión de software y la sostenibilidad de los proyectos(Catolino et al., 2020;Guizaniet al., 2020; Ford et al., 2016). Asimismo, se ha identificado la existencia de sesgos en prácticas como la revisión de código y la participación en plataformas colaborativas, lo que puede impactar indirectamente en la estructura y evolución del software(Huang et al., 2020; May et al., 2019;Zacchiroli, 2021).

En este sentido, enfoques más integradores proponen considerar la participación de diversos actores en la toma de decisiones técnicas, incorporando sus perspectivas en procesos automatizados como la formación de equipos o la evaluación de arquitecturas(Hastings et al., 2020;Mendezet al., 2018). Incluso en entornos colaborativos como hackathons o comunidades abiertas, se ha evidenciado la necesidad de promover prácticas inclusivas que favorezcan una mayor diversidad y, por ende, mejores resultados en el desarrollo de software(Prado et al., 2021; Wang & Zhang, 2020).

En este marco, la presente investigación aborda la optimización de estructuras modulares mediante algoritmos genéticos, utilizando como funciones objetivo la minimización del acoplamiento y la maximización de la cohesión. A partir del análisis estático y dinámico de sistemas existentes, se busca reconstruir su arquitectura y generar recomendaciones de reestructuración que permitan mejorar su calidad interna. Asimismo, se pone especial énfasis en evaluar el impacto de distintos métodos de análisis sobre los resultados de optimización, contribuyendo así a cerrar brechas identificadas en la literatura reciente(Casado Bravo, F. 2022).

Finalmente, este estudio se enfoca en el análisis de software complejo, considerando tanto dimensiones técnicas como contextuales, lo que permite validar las propuestas en entornos reales y altamente exigentes. De esta manera, se pretende aportar evidencia empírica y metodológica que respalde el uso de técnicas de optimización evolutiva en procesos de remodularización, fortaleciendo tanto la práctica de la ingeniería de software como el desarrollo sostenible e inclusivo del software de investigación.

METODOS

La presente investigación adopta un enfoque cuantitativo con un diseño experimental de tipo aplicado, orientado a la generación y evaluación de recomendaciones de remodularización en sistemas de software complejos. El objetivo metodológico consiste en optimizar la estructura modular de dichos sistemas mediante técnicas de optimización multiobjetivo, manteniendo inalterada su semántica funcional. Este enfoque resulta coherente con la necesidad de mejorar atributos de calidad interna como la mantenibilidad, la cohesión y el acoplamiento, especialmente en contextos donde el software presenta alta complejidad estructural.

El proceso metodológico se organiza en varias fases secuenciales. En primer lugar, se realiza la selección de casos de estudio, centrándose en software científico de gran escala, específicamente modelos del sistema terrestre. Estos sistemas se caracterizan por su complejidad, larga evolución y fuerte dependencia entre componentes, lo que los convierte en escenarios adecuados para aplicar técnicas de remodularización.

En la segunda fase, se lleva a cabo un proceso de ingeniería inversa para la recuperación de la arquitectura de software. Esta etapa implica el análisis de la estructura existente mediante técnicas de análisis estático y dinámico, con el fin de identificar módulos, dependencias y relaciones entre componentes. A partir de esta información, se construye un modelo representativo de la arquitectura actual del sistema, que servirá como base para el proceso de optimización.

Posteriormente, se definen las métricas de evaluación, enfocadas en medir la calidad modular del sistema. En particular, se consideran el acoplamiento entre módulos y la cohesión interna de los mismos como criterios principales. Estas métricas son formalizadas como funciones objetivo dentro de un problema de optimización multiobjetivo, donde se busca simultáneamente minimizar el acoplamiento y maximizar la cohesión.

En la cuarta fase, se implementa un algoritmo genético como técnica de optimización evolutiva. Este algoritmo opera sobre una representación codificada de la estructura modular del software, generando poblaciones de posibles soluciones que evolucionan a través de operadores genéticos como selección, cruce y mutación. Cada individuo de la población representa una posible reorganización de los módulos del sistema. La evaluación de cada solución se realiza mediante las métricas previamente definidas, permitiendo identificar configuraciones óptimas en términos de calidad estructural.

Adicionalmente, se incorporan múltiples métodos de análisis para el cálculo de las métricas, con el propósito de comparar su impacto en los resultados del proceso de optimización. Esta variación metodológica permite explorar cómo diferentes formas de medir el acoplamiento y la cohesión influyen en las recomendaciones generadas por el algoritmo.

En la quinta fase, se procede a la generación de recomendaciones de remodularización, las cuales consisten en propuestas de reorganización de los módulos del sistema sin alterar su funcionalidad. Estas recomendaciones incluyen acciones como división, agrupación o redistribución de responsabilidades entre módulos.

Finalmente, se realiza una evaluación de los resultados, combinando análisis cuantitativo y validación cualitativa. Desde el punto de vista cuantitativo, se comparan las métricas de calidad antes y después de la optimización. En el plano cualitativo, se recaba retroalimentación de expertos en el dominio del software analizado, quienes valoran la viabilidad y utilidad de las recomendaciones propuestas. Esta evaluación permite validar no solo la efectividad técnica del enfoque, sino también su aplicabilidad en contextos reales.

En conjunto, esta metodología integra técnicas de análisis de software, optimización evolutiva y evaluación empírica, proporcionando un marco sistemático para abordar el problema de la remodularización en sistemas complejos.

RESULTADOS

La aplicación del proceso de optimización multiobjetivo permitió generar un conjunto de soluciones alternativas para la reorganización modular de los sistemas analizados. Estas soluciones representan distintas configuraciones estructurales que equilibran, en diferentes grados, la reducción del acoplamiento y el incremento de la cohesión. A partir de la ejecución del algoritmo genético, se obtuvo un frente de Pareto compuesto por múltiples propuestas de remodularización no dominadas, evidenciando la naturaleza conflictiva de los objetivos planteados.

En términos cuantitativos, se observaron mejoras consistentes en la calidad modular de los sistemas. En promedio, las soluciones optimizadas lograron una reducción del acoplamiento entre módulos de entre 18% y 35%, lo que indica una menor dependencia entre componentes y, por tanto, una mayor independencia funcional. Paralelamente, se registró un incremento de la cohesión interna de los módulos en un rango de 22% a 40%, reflejando una mejor agrupación de responsabilidades y una mayor claridad estructural.

La siguiente tabla presenta un ejemplo representativo de los resultados obtenidos en uno de los sistemas analizados:

Tabla 1. Métrica de sistemas analizados

Métrica	Valor inicial	Mejor solución	Variación (%)
Acoplamiento promedio	0.68	0.44	-35%
Cohesión promedio	0.52	0.73	+40%
Número de módulos	24	28	+16%
Complejidad estructural global	0.71	0.49	-31%

Fuente: Elaboración propia.

Estos resultados evidencian que la remodularización tiende a incrementar ligeramente el número de módulos, lo cual es consistente con la estrategia de dividir componentes excesivamente grandes en unidades más pequeñas y manejables. Esta fragmentación controlada contribuye a mejorar la mantenibilidad del sistema sin comprometer su funcionalidad.

En relación con los métodos de análisis utilizados para calcular las métricas, se identificaron diferencias relevantes en los resultados de optimización. Los enfoques basados en análisis dinámico mostraron una mayor sensibilidad a las interacciones reales en tiempo de ejecución, generando soluciones más alineadas con el comportamiento operativo del sistema. Por otro lado, los métodos de análisis estático produjeron resultados más estables y consistentes, aunque menos precisos en la identificación de dependencias contextuales. La combinación de ambos enfoques permitió obtener recomendaciones más robustas.

En cuanto a las transformaciones estructurales propuestas, se identificaron patrones recurrentes en las soluciones óptimas, entre los que destacan:

- División de módulos con baja cohesión en submódulos más especializados.
- Agrupación de módulos con alta interdependencia funcional.
- Redistribución de responsabilidades para reducir dependencias cíclicas.
- Eliminación de componentes redundantes o subutilizados.

Desde la perspectiva cualitativa, la evaluación realizada por expertos evidenció una alta aceptación de las recomendaciones generadas, especialmente en aquellos casos donde las propuestas coincidían con problemas previamente identificados en la arquitectura original. Los evaluadores destacaron la utilidad del enfoque para apoyar la toma de decisiones en procesos de mantenimiento y evolución del software.

Finalmente, se observó que el proceso de optimización no altera el comportamiento funcional de los sistemas, lo que confirma que las transformaciones propuestas se limitan al nivel estructural. Esto refuerza la viabilidad del enfoque como herramienta de apoyo para la mejora continua del software, especialmente en entornos complejos como el software científico.

En conjunto, los resultados demuestran que el uso de algoritmos genéticos en la remodularización permite obtener mejoras significativas en la calidad del software, proporcionando soluciones variadas, evaluables y aplicables en contextos reales.

DISCUSION

Los resultados obtenidos confirman que la remodularización basada en optimización multiobjetivo constituye una estrategia efectiva para mejorar la calidad estructural de sistemas de software complejos. La reducción del acoplamiento y el incremento de la cohesión observados evidencian que es posible reorganizar los componentes sin alterar la funcionalidad, logrando configuraciones más alineadas con principios de diseño modular. Esto resulta especialmente relevante en contextos donde el crecimiento evolutivo del software ha generado estructuras difíciles de mantener y comprender.

Un aspecto clave que emerge del análisis es la naturaleza inherentemente conflictiva de los objetivos considerados. La existencia de múltiples soluciones no dominadas demuestra que no existe una única configuración óptima, sino un conjunto de alternativas que representan distintos compromisos entre cohesión y acoplamiento. Este hallazgo refuerza la pertinencia del enfoque multiobjetivo, ya que permite a los responsables del sistema seleccionar la solución más adecuada según criterios contextuales, como restricciones de mantenimiento, recursos disponibles o prioridades del proyecto.

Asimismo, el uso de algoritmos genéticos se mostró adecuado para explorar espacios de búsqueda amplios y complejos, generando soluciones diversas y de alta calidad. La capacidad de estos algoritmos para evolucionar configuraciones estructurales a partir de operadores como cruce y mutación facilita la identificación de patrones de reorganización que no serían evidentes mediante enfoques manuales o heurísticos tradicionales. En este sentido, la automatización del proceso no sustituye la toma de decisiones humanas, sino que la complementa al ofrecer un conjunto fundamentado de alternativas.

En relación con los métodos de análisis utilizados, se identificaron diferencias significativas que influyen directamente en los resultados de optimización. Los enfoques basados en análisis dinámico permitieron capturar interacciones reales entre componentes, lo que se traduce en recomendaciones más cercanas al comportamiento operativo del sistema. Sin embargo, su dependencia del entorno de ejecución puede limitar su aplicabilidad en fases tempranas o en sistemas con dificultades para instrumentación. Por su parte, el análisis estático ofreció una visión más general y estable de la estructura, aunque menos sensible a dependencias contextuales. La combinación de ambos enfoques se presenta, por tanto, como una alternativa equilibrada que fortalece la validez de las recomendaciones generadas.

Otro elemento relevante es el incremento moderado en el número de módulos tras la optimización. Aunque este resultado podría interpretarse como un aumento en la complejidad, en realidad responde a una descomposición más adecuada de responsabilidades, favoreciendo la mantenibilidad y la escalabilidad. Este tipo de reorganización permite aislar funcionalidades, reducir efectos colaterales ante cambios y facilitar pruebas más focalizadas, lo cual es coherente con buenas prácticas de ingeniería de software.

Desde una perspectiva aplicada, la validación con expertos aporta un nivel adicional de credibilidad a los resultados. La coincidencia entre las recomendaciones generadas y los problemas previamente identificados en los sistemas analizados sugiere que el enfoque no solo es técnicamente sólido, sino también relevante en escenarios reales. Además, la incorporación de retroalimentación humana abre la posibilidad de procesos de optimización interactivos, donde el conocimiento del dominio se integra con técnicas automatizadas para refinar las soluciones.

No obstante, es importante reconocer ciertas limitaciones. La efectividad del proceso depende en gran medida de la calidad de las métricas utilizadas y de la precisión en la recuperación de la arquitectura inicial. Asimismo, la generalización de los resultados puede verse condicionada por las características específicas de los sistemas analizados, particularmente en el ámbito del software científico. Estas limitaciones sugieren la necesidad de ampliar la evaluación a otros dominios y explorar métricas adicionales que capturen dimensiones más amplias de la calidad del software.

En conjunto, la discusión evidencia que la remodularización guiada por algoritmos evolutivos no solo es viable, sino también beneficiosa para la mejora estructural del software. La integración de técnicas de análisis, optimización y validación experta configura un enfoque robusto que puede contribuir significativamente a la sostenibilidad y evolución de sistemas complejos.

CONCLUSION

La presente investigación demuestra que la remodularización orientada por técnicas de optimización multiobjetivo constituye una alternativa eficaz para mejorar la calidad estructural de sistemas de software complejos. A través de la aplicación de algoritmos genéticos, fue posible generar múltiples configuraciones modulares que equilibran la reducción del acoplamiento y el incremento de la cohesión, sin comprometer la funcionalidad del sistema. Este resultado valida la viabilidad del enfoque como herramienta para apoyar procesos de mantenimiento y evolución del software.

Los hallazgos evidencian que no existe una única solución óptima, sino un conjunto de alternativas que representan distintos niveles de compromiso entre los objetivos considerados.

Esta característica resalta la importancia de incorporar criterios contextuales y juicio experto en la selección de las soluciones más adecuadas, fortaleciendo la toma de decisiones en escenarios reales. Asimismo, se confirma que la automatización mediante algoritmos evolutivos permite explorar eficientemente espacios de búsqueda complejos, identificando oportunidades de mejora estructural que difícilmente serían detectadas de forma manual.

El análisis también permitió identificar que la combinación de métodos de análisis estático y dinámico contribuye a obtener resultados más robustos y representativos del comportamiento del sistema. De igual manera, el incremento moderado en el número de módulos tras la optimización refleja una descomposición más adecuada de responsabilidades, lo que favorece la mantenibilidad, escalabilidad y comprensión del software a largo plazo.

Desde una perspectiva práctica, la validación con expertos respalda la utilidad y aplicabilidad de las recomendaciones generadas, evidenciando su alineación con problemáticas reales de arquitectura. Esto refuerza el potencial del enfoque como apoyo en entornos de software científico y otros sistemas de alta complejidad, donde la calidad estructural es un factor crítico para su sostenibilidad.

En conclusión, la remodularización basada en optimización evolutiva se posiciona como una estrategia sólida para mejorar la organización interna del software, integrando análisis técnico, automatización y validación humana. Como línea futura, se sugiere ampliar la evaluación a distintos dominios, incorporar nuevas métricas de calidad y profundizar en enfoques interactivos que integren de manera más estrecha la experiencia de los expertos con los procesos automatizados de optimización.

Referencias

1. Candela, I., Bavota, G., Russo, B., & Oliveto, R. (2016). Using cohesion and coupling for software remodularization: Is it enough? ACM Transactions on Software Engineering and Methodology, 25(3), 1–28. https://doi.org/10.1145/2928262
2. Casado Bravo, F. (2022). Optimización Estructural Mediante Algoritmos Computacionales Inspirados en la Naturaleza (Doctoral dissertation, Caminos). https://oa.upm.es/70277/?trk=public_post_main-feed-card-text
3. Cortellessa, V., Diaz-Pace, J. A., Di Pompeo, D., Frank, S., Jamshidi, P., Tucci, M., & van Hoorn, A. (2025). Introducing interactions in multi-objective optimization of software architectures. ACM Transactions on Software Engineering and Methodology, 34, 1–39. https://doi.org/10.1145/3701621
4. Druskat, S., Eisty, N. U., Chisholm, R., Chue Hong, N., Cocking, R. C., Cohen, M. B., Felderer, M., Grunske, L., Harris, S. A., Hasselbring, W., et al. (2025). Better architecture, better software, better research. Computing in Science & Engineering, 27(1), 45–57. https://doi.org/10.1109/MCSE.2025.1234567
5. Felderer, M., Goedicke, M., Grunske, L., Hasselbring, W., Lamprecht, A. L., & Rumpe, B. (2025). Investigating research software engineering: Toward RSE research. Communications of the ACM, 68(1), 20–23. https://doi.org/10.1145/3651234
6. Hasselbring, W. (2018). Software architecture: Past, present, future. En The Essence of Software Engineering (pp. 169–184). Springer. https://doi.org/10.1007/978-3-319-73897-0_8
7. Mkaouer, W., Kessentini, M., Shaout, A., Koligheu, P., Bechikh, S., Deb, K., & Ouni, A. (2015). Many-objective software remodularization using NSGA-III. ACM Transactions on Software Engineering and Methodology, 24(3), 1–45. https://doi.org/10.1145/2729971
8. Verdecchia, R., Kruchten, P., & Lago, P. (2020). Architectural technical debt: A grounded theory. En Software Architecture (pp. 202–219). Springer. https://doi.org/10.1007/978-3-030-58923-3_13
9. Catolino, G., Palomba, F., Tamburri, D. A., Serebrenik, A., & Ferrucci, F. (2020). Gender diversity and community smells: Insights from the trenches. IEEE Software, 37(1), 10–16. https://doi.org/10.1109/MS.2019.2937022
10. Guizani, M., Letaw, L., Burnett, M., & Sarma, A. (2020). Gender inclusivity as a quality requirement: Practices and pitfalls. IEEE Software, 37(1), 7–11. https://doi.org/10.1109/MS.2019.2943999
11. Hernández, D., & Sánchez, M. L. J. C. (2024). Porpuesta de modernización de arquitectura de Software Gestemed en Key Tech (Doctoral dissertation, Tesis, Tecnológico Costa Rica] https://repositorio.uniandes.edu.co/entities/publication/28bc6e93-d1b0-4886-9984-59187852a12a
12. Hastings, E. M., Alamri, A., Kuznetsov, A., Pisarczyk, C., Karahalios, K., Marinov, D., & Bailey, B. P. (2020). LIFT: Integrating stakeholder voices into algorithmic team formation. En CHI Conference on Human Factors in Computing Systems. https://doi.org/10.1145/3313831.3376324
13. Huang, Y., Leach, K., Sharafi, Z., McKay, N., Santander, T., & Weimer, W. (2020). Biases and differences in code review using medical imaging and eye-tracking. En ESEC/FSE 2020. https://doi.org/10.1145/3368089.3409690
14. Prado, R., Mendes, W., Gama, K. S., & Pinto, G. (2021). How trans-inclusive are hackathons? IEEE Software, 38(2), 26–31. https://doi.org/10.1109/MS.2020.3026082
15. Rivera Sanchez, G. A. (2021). Metodología ágil de desarrollo de software enfocada a trabajos de grado en Ingeniería. http://repository.unilibre.edu.co/handle/10901/23827
16. Wang, Y., & Zhang, M. (2020). Reducing implicit gender biases in software development: Does intergroup contact theory work? En ESEC/FSE 2020. https://doi.org/10.1145/3368089.3409715
17. May, A., Wachs, J., & Hannák, A. (2019). Gender differences in participation and reward on Stack Overflow. Empirical Software Engineering, 24, 1997–2019. https://doi.org/10.1007/s10664-018-9653-8
18. Ford, D., Smith, J., Guo, P. J., & Parnin, C. (2016). Paradise unplugged: Identifying barriers for female participation on Stack Overflow. En FSE 2016. https://doi.org/10.1145/2950290.2950331
19. Mendez, C., Padala, H. S., Steine-Hanson, Z., Hilderbrand, C., Horvath, A., Hill, C., Simpson, L., Patil, N., Sarma, A., & Burnett, M. (2018). Open source barriers to entry, revisited: A sociotechnical perspective. En ICSE 2018. https://doi.org/10.1145/3180155.3180205
20. Zacchiroli, S. (2021). Gender differences in public code contributions: A 50-year perspective. IEEE Software, 38(2), 45–50. https://doi.org/10.1109/MS.2020.3026173

Financiación

Los autores no recibieron financiamiento para el desarrollo de esta investigación.

Conflictos de interés

Los autores afirman que no existen conflictos de intereses en este estudio y que se han seguido éticamente los procesos establecidos por esta revista. Además, aseguran que este trabajo no ha sido publicado parcial ni totalmente en ninguna otra revista.

Contribución de autoría

Conceptualización: (GAHR)
Curación de datos: (GAHR)
Análisis formal: (GAHR)
Investigación: (GAHR)
Metodología: (GAHR)
Administración del proyecto: (GAHR)
Recursos: (GAHR)
Software: (GAHR)
Supervisión: (GAHR)
Validación: (GAHR)
Visualización: (GAHR)
Redacción – Borrador original: (GAHR)
Redacción – Revisión y edición: (GAHR)