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.

Submitted: 9-01-2025 | Revised: 22-04-2025 | Accepted: 27-03-2025 | Published: 23-04-2025

Corresponding Author: 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.

INTRODUCTION

The continuous evolution of software systems has underscored the need to adapt their internal structures to meet new requirements for quality, scalability, and maintainability. In this context, remodularization has become a key practice in software engineering, focusing on the reorganization of modular components without altering the system's functional behavior (Hernández, D., & Sánchez, M. L. J. C., 2024). This process aims to improve the structural clarity of the code, facilitate its understanding and maintenance, and optimize the distribution of responsibilities among modules. This is particularly relevant in complex and long-lived systems, where technical debt tends to accumulate over time (Candela et al., 2016; Verdecchia et al., 2020; Hasselbring, 2018).

Traditionally, modular quality has been assessed using metrics such as coupling and cohesion, which are considered fundamental for achieving robust and flexible software architectures (Rivera, 2021). However, simultaneously optimizing these attributes involves addressing potentially conflicting objectives, motivating the adoption of multi-objective optimization approaches. In this regard, genetic algorithms have emerged as an effective strategy for exploring multiple architectural configurations and generating remodularization recommendations that balance these criteria (Mkaouer et al., 2015; Cortellessa et al., 2025). These approaches not only automate the reorganization process but also adapt it to different contexts and system-specific constraints, including the management of technical debt and the ongoing evolution of software.

The challenge is heightened in the field of scientific software, particularly in large-scale systems such as those used in interdisciplinary research, where structural complexity and the need for reproducibility demand more rigorous engineering practices. In this domain, research-oriented software engineering has gained importance by emphasizing the need for sustainable, maintainable, and understandable architectures (Felderer et al., 2025; Druskat et al., 2025). In this context, remodularization not only improves the internal organization of software but also facilitates collaboration among multidisciplinary teams and the integration of new components.

Additionally, in recent years, it has been recognized that software quality depends not only on technical factors but also on social and human aspects present in development processes. Recent studies have shown that factors such as gender diversity, inclusion, and participation in development communities influence code quality, software review, and project sustainability (Catolino et al., 2020; Guizani et al., 2020; Ford et al., 2016). Furthermore, the existence of biases in practices such as code review and participation in collaborative platforms has been identified, which can indirectly impact the structure and evolution of software (Huang et al., 2020; May et al., 2019; Zacchiroli, 2021).

In this regard, more integrative approaches propose considering the participation of diverse stakeholders in technical decision-making, incorporating their perspectives into automated processes such as team formation or architecture evaluation (Hastings et al., 2020; Mendez et al., 2018). Even in collaborative environments such as hackathons or open communities, the need to promote inclusive practices that foster greater diversity and, consequently, improved outcomes in software development has been demonstrated (Prado et al., 2021; Wang & Zhang, 2020).

Within this framework, the present research addresses the optimization of modular structures using genetic algorithms, employing as objective functions the minimization of coupling and the maximization of cohesion. Based on the static and dynamic analysis of existing systems, the aim is to reconstruct their architecture and generate restructuring recommendations to improve their internal quality. Special emphasis is also placed on evaluating the impact of different analysis methods on optimization results, thereby contributing to addressing gaps identified in recent literature (Casado Bravo, F. 2022).

Finally, this study focuses on the analysis of complex software, considering both technical and contextual dimensions, which enables the validation of the proposals in real and highly demanding environments. In this way, the intention is to provide empirical and methodological evidence supporting the use of evolutionary optimization techniques in remodularization processes, thereby strengthening both the practice of software engineering and the sustainable and inclusive development of research software.

METHODS

This research adopts a quantitative approach with an applied experimental design, aimed at generating and evaluating remodularization recommendations in complex software systems.

The methodological objective is to optimize the modular structure of these systems using multi-objective optimization techniques, while maintaining their functional semantics unchanged. This approach aligns with the need to improve internal quality attributes such as maintainability, cohesion, and coupling, especially in contexts where the software exhibits high structural complexity.

The methodological process is organized into several sequential phases. First, case studies are selected, focusing on large-scale scientific software, specifically Earth system models. These systems are characterized by their complexity, long-term evolution, and strong interdependence among components, making them suitable scenarios for the application of remodularization techniques.

In the second phase, a reverse engineering process is conducted for software architecture recovery. This stage involves analyzing the existing structure using static and dynamic analysis techniques to identify modules, dependencies, and relationships between components. Based on this information, a representative model of the current system architecture is constructed, which serves as the foundation for the optimization process.

Subsequently, evaluation metrics are defined, focusing on measuring the modular quality of the system. In particular, coupling between modules and their internal cohesion are considered as the primary criteria. These metrics are formalized as objective functions within a multi-objective optimization problem, where the goal is to simultaneously minimize coupling and maximize cohesion.

In the fourth phase, a genetic algorithm is implemented as an evolutionary optimization technique. This algorithm operates on an encoded representation of the software's modular structure, generating populations of possible solutions that evolve through genetic operators such as selection, crossover, and mutation. Each individual in the population represents a potential reorganization of the system's modules. The evaluation of each solution is performed using the previously defined metrics, enabling the identification of optimal configurations in terms of structural quality.

Additionally, multiple analysis methods are incorporated for the calculation of the metrics, with the aim of comparing their impact on the results of the optimization process. This methodological variation allows for the exploration of how different approaches to measuring coupling and cohesion influence the recommendations generated by the algorithm.

In the fifth phase, recommendations for remodularization are generated, consisting of proposals for reorganizing the system's modules without altering its functionality. These recommendations include actions such as splitting, grouping, or redistributing responsibilities among modules.

Finally, an evaluation of the results is conducted, combining quantitative analysis and qualitative validation. From a quantitative perspective, quality metrics are compared before and after optimization. On the qualitative side, feedback is gathered from experts in the domain of the analyzed software, who assess the feasibility and usefulness of the proposed recommendations. This evaluation enables validation not only of the technical effectiveness of the approach but also of its applicability in real-world contexts.

Taken together, this methodology integrates software analysis techniques, evolutionary optimization, and empirical evaluation, providing a systematic framework to address the problem of remodularization in complex systems.

RESULTS

The application of the multi-objective optimization process enabled the generation of a set of alternative solutions for the modular reorganization of the analyzed systems. These solutions represent different structural configurations that balance, to varying degrees, the reduction of coupling and the increase of cohesion. From the execution of the genetic algorithm, a Pareto front composed of multiple non-dominated remodularization proposals was obtained, highlighting the conflicting nature of the stated objectives.

In quantitative terms, consistent improvements in the modular quality of the systems were observed. On average, the optimized solutions achieved a reduction in coupling between modules of 18% to 35%, indicating lower dependency between components and, therefore, greater functional independence. In parallel, an increase in the internal cohesion of the modules was recorded in the range of 22% to 40%, reflecting better grouping of responsibilities and greater structural clarity.

The following table presents a representative example of the results obtained in one of the analyzed systems:

Table 1. Metrics of analyzed systems

Metric	Initial value	Best solution	Variation (%)
Average coupling	0.68	0.44	-35%
Average cohesion	0.52	0.73	+40%
Number of modules	24	28	+16%
Global structural complexity	0.71	0.49	-31%

Source: Own elaboration.

These results indicate that remodularization tends to slightly increase the number of modules, which is consistent with the strategy of splitting excessively large components into smaller, more manageable units. This controlled fragmentation contributes to improving the maintainability of the system without compromising its functionality.

Regarding the analysis methods used to calculate the metrics, relevant differences were identified in the optimization results. Approaches based on dynamic analysis showed greater sensitivity to actual runtime interactions, generating solutions more aligned with the operational behavior of the system. In contrast, static analysis methods produced more stable and consistent results, although they were less precise in identifying contextual dependencies. The combination of both approaches allowed for more robust recommendations.

With respect to the proposed structural transformations, recurrent patterns were identified in the optimal solutions, among which the following stand out:

- Splitting modules with low cohesion into more specialized submodules.
- Grouping modules with high functional interdependence.
- Redistribution of responsibilities to reduce cyclic dependencies.
- Elimination of redundant or underutilized components.

From a qualitative perspective, the expert evaluation demonstrated a high level of acceptance of the generated recommendations, particularly in cases where the proposals aligned with issues previously identified in the original architecture. The evaluators emphasized the usefulness of the approach in supporting decision-making processes related to software maintenance and evolution.

Finally, it was observed that the optimization process does not alter the functional behavior of the systems, confirming that the proposed transformations are restricted to the structural level. This finding reinforces the feasibility of the approach as a support tool for the continuous improvement of software, especially in complex environments such as scientific software.

Taken together, the results demonstrate that the use of genetic algorithms in remodularization enables significant improvements in software quality, providing diverse, assessable, and applicable solutions in real-world contexts.

DISCUSSION

The results obtained confirm that remodularization based on multi-objective optimization is an effective strategy for improving the structural quality of complex software systems. The observed reduction in coupling and increase in cohesion demonstrate that it is possible to reorganize components without altering functionality, achieving configurations that are more consistent with modular design principles. This is particularly relevant in contexts where the evolutionary growth of software has resulted in structures that are difficult to maintain and understand.

A key aspect emerging from the analysis is the inherently conflicting nature of the objectives considered. The existence of multiple non-dominated solutions indicates that there is no single optimal configuration, but rather a set of alternatives representing different trade-offs between cohesion and coupling. This finding underscores the relevance of the multi-objective approach, as it enables system stakeholders to select the most appropriate solution according to contextual criteria, such as maintenance constraints, available resources, or project priorities.

Similarly, the use of genetic algorithms proved suitable for exploring large and complex search spaces, generating diverse and high-quality solutions.

The ability of these algorithms to evolve structural configurations through operators such as crossover and mutation facilitates the identification of reorganization patterns that may not be apparent through manual or traditional heuristic approaches. In this regard, the automation of the process does not replace human decision-making, but rather complements it by providing a well-founded set of alternatives.

Regarding the analysis methods used, significant differences were identified that directly influence the optimization results. Approaches based on dynamic analysis enabled the capture of actual interactions between components, resulting in recommendations that more closely reflect the operational behavior of the system. However, their dependence on the execution environment may limit their applicability in early phases or in systems with instrumentation challenges. In contrast, static analysis provided a more general and stable view of the structure, although it was less sensitive to contextual dependencies. The combination of both approaches thus emerges as a balanced alternative that enhances the validity of the generated recommendations.

Another relevant aspect is the moderate increase in the number of modules after optimization. Although this result could be interpreted as an increase in complexity, it actually reflects a more appropriate decomposition of responsibilities, promoting maintainability and scalability. This type of reorganization allows functionalities to be isolated, reduces side effects in response to changes, and facilitates more focused testing, which is consistent with good software engineering practices.

From an applied perspective, validation with experts provides an additional level of credibility to the results. The alignment between the generated recommendations and the issues previously identified in the analyzed systems suggests that the approach is not only technically sound but also relevant in real-world scenarios. Furthermore, the incorporation of human feedback opens the possibility for interactive optimization processes, where domain knowledge is integrated with automated techniques to refine solutions.

Nevertheless, certain limitations should be acknowledged. The effectiveness of the process depends largely on the quality of the metrics used and the accuracy of recovering the initial architecture. Similarly, the generalizability of the results may be influenced by the specific characteristics of the analyzed systems, particularly in the field of scientific software. These limitations suggest the need to extend the evaluation to other domains and to explore additional metrics that capture broader dimensions of software quality.

Overall, the discussion indicates that remodularization guided by evolutionary algorithms is not only viable but also beneficial for the structural improvement of software. The integration of analysis, optimization, and expert validation techniques constitutes a robust approach that can contribute significantly to the sustainability and evolution of complex systems.

CONCLUSIONS

This research demonstrates that remodularization guided by multi-objective optimization techniques is an effective alternative for improving the structural quality of complex software systems. Through the application of genetic algorithms, it was possible to generate multiple modular configurations that balance the reduction of coupling and the increase of cohesion, without compromising system functionality. This result validates the feasibility of the approach as a tool to support software maintenance and evolution processes.

The findings indicate that there is no single optimal solution, but rather a set of alternatives representing different levels of trade-off among the considered objectives. This characteristic underscores the importance of incorporating contextual criteria and expert judgment in selecting the most appropriate solutions, thereby enhancing decision-making in real-world scenarios. Furthermore, the results confirm that automation through evolutionary algorithms enables efficient exploration of complex search spaces, identifying opportunities for structural improvement that would be difficult to detect manually.

The analysis also revealed that the combination of static and dynamic analysis methods contributes to obtaining more robust and representative results of the system's behavior. Additionally, the moderate increase in the number of modules after optimization reflects a more appropriate decomposition of responsibilities, which supports maintainability, scalability, and long-term software comprehension.

From a practical perspective, validation with experts supports the usefulness and applicability of the generated recommendations, demonstrating their alignment with real-world architectural issues.

This reinforces the potential of the approach as support in scientific software environments and other highly complex systems, where structural quality is a critical factor for sustainability.

In conclusion, remodularization based on evolutionary optimization is positioned as a robust strategy to improve the internal organization of software, integrating technical analysis, automation, and human validation. As a future direction, it is suggested to expand the evaluation to different domains, incorporate new quality metrics, and further develop interactive approaches that more closely integrate expert experience with automated optimization processes.

REFERENCES

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

FUNDING

The authors received no funding for the development of this research.

CONFLICT OF INTEREST

The authors declare that there are no conflicts of interest in this study and that the processes established by this journal have been ethically followed. Furthermore, they confirm that this work has not been published, either partially or in full, in any other journal.

AUTHORSHIP CONTRIBUTIONS

Conceptualization: (GAHR)
Data curation: (GAHR)
Formal analysis: (GAHR)
Investigation: (GAHR)
Methodology: (GAHR)
Project administration: (GAHR)
Resources: (GAHR)
Software: (GAHR)
Supervision: (GAHR)
Validation: (GAHR)
Visualization: (GAHR)
Writing – original draft: (GAHR)
Writing – review & editing: (GAHR)