Uso de revisiones de código modernas en pequeñas organizaciones

*Use of Modern Code Reviews in Small Organizations*

***Resumen* — En la actualidad, las revisiones de código modernas se han convertido en una práctica ampliamente adoptada en proyectos de software a gran escala. Esta modalidad de revisión ha demostrado beneficios significativos, como la mejora de la calidad del código, fomentar la colaboración en el desarrollo de software y aumentar la comprensión del código por parte de los miembros del equipo. Además, han sido objeto de interés en la comunidad de ingeniería de software en la última década. El objetivo de esta investigación fue identificar en qué medida se usan las revisiones de código modernas en pequeñas organizaciones o proyectos, y cuáles son los beneficios asociados. Los resultados revelaron poca evidencia documentada del uso de revisiones de código modernas en el contexto de pequeñas organizaciones.**

***Palabras Clave - revisión de código moderno; pequeñas organizaciones; entidades muy pequeñas.***

***Abstract* — Currently, modern code reviews have become a widely adopted practice in large-scale software projects. This review approach has shown significant benefits, such as improving code quality, fostering collaboration in software development, and enhancing team members' understanding of the code. Additionally, they have garnered interest within the software engineering community in the past decade. The aim of this research was to identify the extent to which modern code reviews are used in small organizations or projects, and what associated benefits they bring. The results revealed limited documented evidence of the use of modern code reviews in the context of small organizations.**

***Keywords - modern code review; small organizations; very small entities.***

1. Introducción

En el ámbito del desarrollo de software, las revisiones de código han evolucionado de su enfoque original [1] en la detección de errores a una práctica más amplia que busca mejorar la calidad del código, fomentar la colaboración y facilitar el aprendizaje en proyectos de gran escala [2]. A diferencia de los métodos formales de inspección de código, las revisiones de código modernas (RCM) se llevan a cabo de manera asíncrona y se apoyan en herramientas que agilizan el proceso, siguiendo la metodología adoptada por la organización.

En un estudio reciente se muestran que los principales temas que se han investigado sobre las RCM están relacionados con el efecto de métodos o herramientas en las revisiones, el impacto en la calidad del producto, del proceso y en aspectos humanos como el nivel de experiencia de los revisores, la confiabilidad en los resultados de la revisión y la imparcialidad de la revisión [4].

Sin embargo, en revisiones sistemáticas de literatura como Dávila [3] y Badampudi [4] no han informado del uso de RCM en el contexto de pequeñas organizaciones o pequeños proyectos. Por lo que estamos interesados en explorar su efectividad en este contexto y las aproximaciones para su implementación. Por lo tanto, se llevó a cabo un mapeo sistemático en cuatro bases de datos relevantes en ingeniería de software. Para validar nuestros resultados, verificamos los estudios empíricos identificados en una revisión sistemática de literatura [3].

Los resultados principales de esta investigación en proceso sugieren poca evidencia respecto del estudio de revisiones de código en el contexto de pequeñas organizaciones. Los cinco artículos primarios que seleccionamos mencionan brevemente el uso de inspecciones tradicionales, sin hacer referencia a RCM.

El documento se estructura en primer lugar con una descripción del proceso de las RCM y una breve presentación del ISO/IEC 29110: perfil básico [5] en la Sección II. A continuación, se expone la metodología utilizada en el mapeo sistemático en la Sección III, seguida de la presentación de los resultados en la Sección IV. Posteriormente se discuten las implicaciones y significados de los hallazgos y se abordan las amenazas a la validez del estudio en la Sección V. Por último, se presentan las conclusiones del trabajo en la Sección VI.

1. Antecedentes

En el proceso de revisión de código moderno, existen variaciones según las metodologías y herramientas utilizadas por cada organización. En el trabajo realizado por Dávila [3], se describe un proceso típico de RCM que consta de dos fases. La primera fase involucra la planificación y configuración de la revisión. Comienza cuando se introduce un cambio en una herramienta que soporta el proceso de la RCM, y se seleccionan y notifican a los revisores correspondientes [3]. La segunda fase es la ejecución de la revisión de código, donde se inspecciona el código, se agregan comentarios sobre los cambios realizados y se toman decisiones respecto a si se rechaza, se acepta o si se requieren correcciones [3].

Un marco de referencia apropiado para definir una pequeña organización puede ser el estándar internacional ISO/IEC 29110 - Perfil Básico [5]. Este estándar se enfoca en mejorar los procesos y la calidad de los productos en empresas o proyectos de desarrollo de software que constan de hasta 25 personas. Estas organizaciones se clasifican como entidades muy pequeñas o VSE por sus siglas en inglés (*Very Small Entities*). En el caso específico del perfil básico, se refiere a un solo equipo trabajando en un solo proyecto.

El proceso de implementación de software en el ISO 29110 incluye tareas de verificación y validación como parte de la evaluación de todos los productos de trabajo generados en el proceso de implementación de software. En particular, señala que los defectos son identificados, corregidos y documentados. Con esto en cuenta, las RCM podrían ser útiles como una práctica para llevar a cabo la verificación del código de software.

1. Metodología

El mapeo sistemático de literatura se basó en Petersen et al. 2008 [6] y permitió identificar los artículos primarios que mencionan el tema de revisión de código en pequeñas organizaciones.

En la búsqueda en bases de datos, se seleccionaron cuatro bases de datos relevantes en el área de computación e ingeniería de software: IEEE Xplore, ACM DL, Scopus y Web of Science (WOS). Se utilizó una cadena de búsqueda (Fig. 1) que se basa en [3] para identificar los términos relacionados con las revisiones de código. Además, incluimos términos relacionados con el contexto de pequeñas organizaciones. Por otra parte, no se excluyeron aquellos términos referentes a otras modalidades de la revisión de código para no restringir demasiado el alcance de la investigación. La consulta se realizó sobre los campos título, resumen y palabras clave de Scopus y WOS, y se aplicó en todos los metadatos en IEEE y ACM.

|  |
| --- |
| ***Query:*** *("code review" OR "code inspection" OR "software inspection" OR "formal inspection" ) AND (29110 OR VSE OR "very small entity" OR "small company" OR "small organization" OR "small project")* |

Figura 1. Cadena de búsqueda.

Para la selección de artículos se aplicaron los siguientes criterios de inclusión: el artículo está escrito en inglés, menciona el tema de revisión de código en cualquiera de sus modalidades (e.g., inspección o revisión de código), menciona que la revisión de código se aplicó o está orientada para su uso en proyectos pequeños de software u organizaciones pequeñas. Por otra parte, los criterios de exclusión son artículos que no están publicados en inglés, revisiones asociadas a la gestión del proyecto, revisiones de código en proyectos grandes. Se realizó un análisis de los títulos, resúmenes y palabras claves de los artículos para determinar si cumplían con los criterios. Aquellos que lo cumplían fueron analizados en su totalidad.

TABLA I. Resultados de búsqueda en las bases de datos.

|  |  |  |
| --- | --- | --- |
| Base de datos | URL | # Artículos |
| IEEEXplore | http://ieeexplore.ieee.org | 0 |
| ACM DL | http://portal.acm.org | 76 |
| Scopus | https://www.scopus.com | 4 |
| Web of Science | https://www.webofscience.com | 1 |
| **Total** |  | **81** |

En cuanto a la caracterización y extracción de información relevante de los artículos seleccionados, se utilizaron los siguientes criterios de análisis: tamaño de la organización o proyecto donde se realizó la práctica, modalidad de revisión, si describen o citan el proceso de revisión realizado y el objetivo del estudio.

Para validar los resultados del mapeo sistemático presentado en este trabajo, realizamos un análisis de los estudios experimentales reportados en una revisión sistemática de literatura [3]. De 37 artículos reportados en la categoría de evaluación, siete son estudios experimentales en donde participaron estudiantes o profesionales del desarrollo de software.

1. Resultados
   1. *Búsqueda en base de datos.*

Los resultados de las consultas en cada base de datos se muestran en la Tabla 1. A pesar de la flexibilidad en los criterios de inclusión, solo cinco de los 81 estudios fueron seleccionados. Los resultados de la clasificación de los artículos primarios se presentan en la Tabla II. La columna “describe” indica si el artículo presenta detalles del proceso para realizar la revisión.

Los artículos seleccionados datan principalmente del período entre 2005 y 2010, lo que sugiere que son revisiones en la modalidad tradicional de Fagan o inspecciones formales (IF). Algunos artículos citan a Fagan al introducir el concepto de revisión o inspección [7,8,10], mientras que otros describen el proceso de revisión en la fase de análisis de requisitos [9] y en la fase de implementación de código [11] con elementos propios de las inspecciones formales.

Los artículos analizados señalan propuestas para mejorar el proceso de inspección [7,10,11], así como evaluación de métodos relacionados con el uso de inspecciones [9, 10]. Sin embargo, aunque se menciona el uso y la aplicabilidad en contextos de proyectos pequeños, los tamaños de las organizaciones involucradas en los estudios no se presentan con claridad.

En solo dos de los artículos revisados, se puede determinar el tamaño de las organizaciones en función de la cantidad de participantes. En el trabajo de Walia [9], se realizó un experimento en el que se revisaron cuatro artefactos elaborados por dos equipos de ocho estudiantes y otros dos equipos de seis. El objetivo era evaluar el uso del método de captura-recaptura para estimar la cantidad de defectos en un artefacto. En el trabajo de Phongpaibul [8], se compararon las inspecciones de software con el desarrollo en pares a través de un experimento con ocho grupos de cinco estudiantes y dos grupos conformados por profesionales. Ambos estudios pueden resultar relevantes para pequeñas organizaciones, ya que se indica que en los proyectos participaron menos de 25 personas. Sin embargo, se usan inspecciones formales y no RCM. Además, las experiencias reportadas se han realizado, en la mayoría de los casos, en contextos educativos.

TABLA II. Clasificación de artículos primarios.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cita | Tamaño(s) | Tipo | Describe | Objetivo |
| [7] | N/A | IF | Sí | Proponer un marco de procesos de mejora en IF |
| [8] | 5 personas | IF | Sí | Comparar dos modalidades de revisión |
| [9] | 6 y 8 personas | IF | Sí | Evaluar un método de estimación en IF |
| [10] | Proyectos entre 250 y 869 persona-hora | IF | Sí | Presentar una métrica para cuantificar la calidad del proceso de IF |
| [11] | Grande | IF | Sí | Presentar el desarrollo de una herramienta de soporte para el proceso de IF |

Por otro lado, en el trabajo de Nair [10], se presenta el desarrollo de una métrica para cuantificar la calidad del proceso de inspección, pero el tamaño de los proyectos se describe en términos de personas-horas, por lo que no es posible determinar si se ajustan a la definición de pequeñas organizaciones. De manera similar, en el artículo de Holzmann [11], se propone una herramienta automatizada para las revisiones de código en las inspecciones, pero solo se menciona que la organización en la que se reportan resultados positivos es "grande", sin proporcionar ningún marco de referencia. Por último, en el trabajo de Harjumaa [7], se presenta un método basado en patrones de diseño para mejorar el proceso de revisión en la industria, pero no se menciona el tamaño del proyecto del experimento.

Por lo tanto, los resultados de la revisión sistemática nos indican que los estudios primarios que seleccionamos no muestran evidencia que se utilicen las revisiones de código modernas.

* 1. *Búsqueda en una revisión sistemática de literatura*

Dado el número limitado de artículos primarios identificados en el mapeo sistemático, verificamos nuestros resultados con los artículos experimentales identificados en [3]. Los siete estudios experimentales, a excepción de [17], presentan alguna herramienta, método o enfoque adicional con el objetivo de mejorar la efectividad y eficacia de las RCM, como se muestra en la Tabla III.

Para evaluar las propuestas de los estudios, se utilizaron técnicas de minería de repositorios de código abierto (OSS) en cuatro de los artículos seleccionados. Estos estudios complementaron sus análisis con experimentos que involucraron a participantes estudiantes [13, 14, 15], mientras que en uno de ellos se incluyeron tanto estudiantes como profesionales de la industria [18]. En el artículo [12], se llevó a cabo una encuesta a participantes con experiencia en la industria, junto con un experimento que involucró a estudiantes. Además, en el artículo [16] se presentó un análisis basado únicamente en un repositorio de código abierto, mientras que en [17] se realizó un experimento exclusivamente con estudiantes.

Para abordar la complejidad de los cambios a revisar, las propuestas de herramientas incorporaron funcionalidades adicionales para las RCM. Estas incluyeron la extracción del contexto de control y flujo de datos alrededor del código [12], el uso de analizadores semánticos [13, 18], la identificación y particionamiento de cambios compuestos [14], mejoras en la visualización de diferencias [15], y la identificación de cambios sobresalientes [16].

TABLA III. Clasificación de estudios empíricos.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cita | Tamaño(s) | Tipo | Describe | Objetivo |
| [12] | OSS de 400 KLOC | RCM | No | Propuesta de herramienta para comprender y revisar mejor los cambios |
| [13] | N/A | RCM | No | Propuesta para medir el impacto de los cambios por revisar |
| [14] | N/A | RCM | No | Propuesta de herramienta para la descomposición de cambios por revisar |
| [15] | N/A | RCM | No | Propuesta de herramienta para visualizar los cambios |
| [16] | N/A | RCM | No | Propuesta de herramienta para revisar cambios |
| [17] | 500 LOC | RCM | No | Medir el impacto de la gamificación en el proceso de RCM |
| [18] | N/A | RCM | No | Propuesta de herramienta para revisar cambios |

Al igual que en los resultados de los estudios primarios seleccionados, el tamaño de los proyectos no se muestra con claridad. Si bien se mencionan los participantes en los experimentos y se proporciona información sobre el tamaño de los cambios a revisar o del repositorio en términos de líneas de código (LOC) [13, 14, 15, 16, 18], o en términos de la cantidad de *commits* en el repositorio [17, 18], no se ofrece información detallada sobre la cantidad de personas involucradas en los proyectos de los repositorios y los experimentos no mencionan la cantidad de personas que trabajaron en los proyectos de desarrollo de software.

1. Discusión

Encontramos pocos artículos primarios que hablen del tema de revisiones en pequeñas organizaciones. En los cinco artículos primarios seleccionados se cita a Fagan o presentan características propias de las inspecciones tradicionales. Sin embargo, dentro del conjunto de artículos primarios seleccionados no encontramos evidencia de que se hayan aplicado las RCM en el contexto de pequeñas organizaciones. Por otra parte, el análisis que hicimos en los estudios experimentales reportados en [3] mostró falta de datos que permitan categorizar el tamaño de la organización o proyecto.

Respecto de las amenazas a la validez, la selección de artículos se basó en una cadena de búsqueda aborda distintos tipos de revisión [3] y agregamos términos relacionados con las pequeñas organizaciones. El proceso de búsqueda automática identificó pocos estudios. Por lo tanto, se requiere una búsqueda más exhaustiva, considerando la técnica de bola de nieve, para validar los resultados de este estudio. Por otra parte, el proceso de selección de artículos fue validado por el segundo autor. Además, la extracción y análisis de datos también fue validada por el segundo autor.

1. Conclusiones

Este trabajo presenta un mapeo sistemático para identificar en qué medidas se usan las revisiones de código modernas en pequeñas organizaciones. El análisis de los estudios primarios seleccionados fue validado consultando revisiones sistemáticas de literatura sobre revisiones de código modernas. Los resultados evidencian que existe poca información reportada de su uso en este contexto.

Como un trabajo futuro se propone realizar una investigación exhaustiva con un alcance más amplio para confirmar si existe una brecha en el conocimiento en este campo. Además, dada el escaso número de estudios de RCM en pequeñas organizaciones, es deseable estudiar empíricamente cómo las pequeñas organizaciones están realizando las revisiones de código y cómo mejorar sus procesos de revisión para mejorar la calidad de sus productos.

Referencias Bibliográfica

1. Fagan, M. E. (1973). Advances in software inspections to reduce errors in program development. lBM Systems.
2. Bacchelli, A., & Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. In 2013 35th International Conference on Software Engineering (ICSE) (pp. 712-721). IEEE.
3. Davila, Nicole & Nunes, Ingrid. (2021). A systematic literature review and taxonomy of modern code review. Journal of Systems and Software. 177. 110951. 10.1016/j.jss.2021.110951.
4. Deepika Badampudi, Michael Unterkalmsteiner, and Ricardo Britto. 2023. Modern Code Reviews—Survey of Literature and Practice. ACM Trans. Softw. Eng. Methodol. 32, 4, Article 107 (May 2023), 61 pages. https://doi.org/10.1145/3585004
5. ISO/IEC TR 29110-5-1-2:2011 – Software engineering – Lifecycle Profiles for Very Small Entities (VSEs) –Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile, International Organization for Standardization/International Electrotechnical Commission: Geneva, Switzerland. Available at no cost from ISO at: http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html
6. Petersen, Kai & Feldt, Robert & Mujtaba, Shahid & Mattsson, Michael. (2008). Systematic Mapping Studies in Software Engineering. Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering. 17.
7. Lasse Harjumaa (2005). A pattern approach to software inspection process improvement. , 10(4), 455–465. doi:10.1002/spip.239
8. Phongpaibul, Monvorath; Boehm, Barry (2006). [ACM Press the 2006 ACM/IEEE international symposium - Rio de Janeiro, Brazil (2006.09.21-2006.09.22)] Proceedings of the 2006 ACM/IEEE international symposium on International symposium on empirical software engineering - ISESE '06 - An empirical comparison between pair development and software inspection in Thailand. , (), 85–. doi:10.1145/1159733.1159749
9. Walia, Gursimran Singh; Carver, Jeffrey C. (2008). [ACM Press the Second ACM-IEEE international symposium - Kaiserslautern, Germany (2008.10.09-2008.10.10)] Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '08 - Evaluation of capture-recapture models for estimating the abundance of naturally-occurring defects. , (), 158–. doi:10.1145/1414004.1414031
10. Nair, T.R. Gopalakrishnan; Suma, V. (2010). A paradigm for metric based inspection process for enhancing defect management. ACM SIGSOFT Software Engineering Notes, 35(3), 1–. doi:10.1145/1764810.1764827
11. Gerard J. Holzmann (2010). SCRUB: a tool for code reviews. , 6(4), 311–318. doi:10.1007/s11334-010-0136-x
12. Zhang, T., Song, M., Pinedo, J., Kim, M., 2015. Interactive code review for systematic changes. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. In: ICSE ’15, Vol. 1, IEEE Press, pp. 111–122. http://dx.doi.org/10.1109/ICSE.2015.33.
13. Hanam, Q., Mesbah, A., Holmes, R., Aiding code change understanding with semantic change impact analysis. In: 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME). 2019, pp. 202–212.
14. Tao, Y., Kim, S., 2015. Partitioning composite code changes to facilitate code review. In: 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. In: MSR ’15, IEEE Press, pp. 180–190. http://dx.doi.org/10.1109/MSR.2015.24.
15. Huang, K., Chen, B., Peng, X., Zhou, D., Wang, Y., Liu, Y., & Zhao, W. 2018. ClDiff: generating concise linked code differences. Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering - ASE 2018. doi:10.1145/3238147.3238219
16. Huang, Y., Jia, N., Chen, X., Hong, K., Zheng, Z., 2018. Salient-class location: Help developers understand code change in code review. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. In: ESEC/FSE 2018, Association for Computing Machinery, New York, NY, USA, pp. 770–774. http://dx.doi.org/10.1145/3236024.3264841.
17. Khandelwal, S., Sripada, S.K., Reddy, Y.R., 2017. Impact of gamification on code review process: An experimental study. In: Proceedings of the 10th Innovations in Software Engineering Conference. In: ISEC ’17, ACM, New York, NY, USA, pp. 122–126. http://dx.doi.org/10.1145/3021460.3021474.
18. Menarini, M., Yan, Y., Griswold, W.G., 2017. Semantics-assisted code review: An efficient tool chain and a user study. In: 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE Press, pp. 554–565. http://dx.doi.org/10.1109/ASE.2017.8115666.