*Data and Metadata*. 2024; 3:.233
doi: 10.56294/dm2024.233

**ORIGINAL**

# Technical debt management in academic environments: perspectives and challenges of a software development team

## Gestión de la deuda técnica en entornos académicos: perspectivas y desafíos de un equipo de desarrollo de software

Edwin Fabricio Lozada Torres[1,2], Rodrigo Cadena Martínez[3], María Angélica Pico Pico[1]

[1]Universidad Regional Autónoma de Los Andes – UNIANDES. Carrera de Software, Ambato, Ecuador.
[2]Estudiante de Doctorado en Informática, Universidad Americana de Europa, C. P. 77500, Quintana Roo, México.
[3]Departamento de Informática, Universidad Americana de Europa, C. P. 77500, Quintana Roo, México.

**Citar como**: Lozada Torres EF, Cadena Martínez R, Pico Pico MA. Technical debt management in academic environments: perspectives and challenges of a software development team. Data and Metadata. 2024; 3:.233. https://doi.org/10.56294/dm2024.233

**Enviado:** 11-01-2024 **Revisado:** 16-07-2024 **Aceptado:** 11-12-2024 **Publicado:** 12-12-2024

**Editor:** Adrián Alejandro Vitón-Castillo

**Autor para la correspondencia:** Edwin Fabricio Lozada Torres

## ABSTRACT

Technical debt are those convenience tasks that developers perform to obtain short-term benefits but that in the future can produce activities that are difficult to perform or more costly. The impact of technical debt is perceived in all project resources, becoming a present and future problem in software development, affecting the quality of the software. This study uses a qualitative analysis, through interviews with a software development team, explores the knowledge and perception of technical debt, the practices used to manage it, the factors that influence its appearance and accumulation, the impact that development suffers, as well as the team and the maintainability of the software, as a result, ideas and solutions are proposed to address it effectively. The study was carried out with a qualitative investigative approach, which was based on the study of a single case with which the experiences and practices in the perception and management of technical debt are investigated. The data obtained reveal that technical debt affects the normal development of the development team and that it is present in the activities carried out by the team. Although the development team has a perception of the debt, it does not address it directly, and it is necessary to implement good practices to identify it and integrate it into the software development process. Addressing technical debt with the implementation of a formal process to manage it will ensure that scheduled time is controlled and resources are optimized, which will result in a culture of good practices that will ensure product quality, improving team performance and guaranteeing software maintainability.

**Keywords:** Software Development; Development Team; Technical Debt; Technical Debt Management.

## RESUMEN

La deuda técnica son aquellas tareas a conveniencia que realizan los desarrolladores para obtener beneficios a corto plazo pero que a futuro puede producir actividades difíciles de realizar o más costosos. El impacto de la deuda técnica se la percibe en todos los recursos del proyecto, convirtiéndose en un problema presente y futuro del desarrollo de software afectando en la calidad del software. Este estudio, utiliza un análisis cualitativo, mediante entrevistas a un equipo de desarrollo de software, explora el conocimiento y la percepción de la deuda técnica, las prácticas que se utilizan para gestionarla, los factores que influyen en su aparecimiento y acumulación, el impacto que sufre el desarrollo, así como el equipo y la mantenibilidad del software, como resultado se proponen ideas y soluciones para abordarla de manera efectiva. El estudio se efectuó con un enfoque investigativo cualitativo, que se basó en el estudio de un caso único con el cual se indaga las experiencias y prácticas en la percepción y gestión de la deuda técnica. Los datos obtenidos

revelan que la deuda técnica afecta el normal desenvolvimiento del equipo de desarrollo y que se encuentra presente en las actividades que realiza el equipo. Si bien el equipo de desarrollo tiene una percepción de la deuda no la aborda de manera directa siendo necesario implementar buenas prácticas para identificarla e integrarla en el proceso de desarrollo de software. Abordar la deuda técnica con la implementación de un proceso formal para gestionarla logrará que se controle el tiempo programado y se optimicen los recursos, lo que se derivará en una cultura de buenas prácticas que asegurará la calidad del producto mejorando el rendimiento del equipo y garantizando la mantenibilidad del software.

**Palabras clave:** Desarrollo de Software; Equipo de Desarrollo; Deuda Técnica; Gestión de la Deuda Técnica.

## INTRODUCCIÓN

La deuda técnica son aquellas tareas a conveniencia que realizan los desarrolladores para obtener beneficios a corto plazo pero que a futuro puede producir actividades difíciles de realizar o más costosos. El impacto de la deuda técnica se la percibe en todos los recursos del proyecto, convirtiéndose en un problema presente y futuro del desarrollo de software afectando en la calidad del software.

El termino deuda técnica genera ambigüedad en muchos desarrolladores, esta expresión se la asocia con todos los problemas derivados de la falta de una planificación adecuada o la baja calidad en el desarrollo del software. Si bien existen técnicas y herramientas que aportan en una detección a tiempo de la deuda, la ausencia de literatura e información técnica suficiente crean desconcierto para la interpretación y control la deuda técnica a tiempo (Stochel et al., 2020).

Si bien es cierto que, la evolución de las tecnologías ha tenido una relación directa con el crecimiento de la deuda técnica, al mismo tiempo mientras más tiempo ha sido utilizado un software requiere de un mayor mantenimiento debido a nuevos requerimientos, entornos regulatorios, nuevos ambientes, cambios tecnológicos, etc., estas actividades generan deuda extra que a la larga se hace imposible pagarla (Digkas et al., 2022).

En relación con los beneficios que se obtienen al aplicar atajos al desarrollar software, estos se derivan en la deuda técnica, que, a futuro representan costos extras al realizar los cambios en el producto. La utilización de herramientas y enfoques que permitan priorizar la deuda contribuyen a una adecuada gestión mejorando la comunicación tanto dentro del equipo como con el cliente (Reboucas de Almeida, 2019).

En efecto, la deuda técnica es un fenómeno que afecta a todas las fases de modelamiento del software, siendo el diseño de la arquitectura donde se producen patrones en la mayoría de las ocasiones consientes que deben tomar los arquitectos de software con la finalidad de lograr beneficios a corto plazo con las evidentes deficiencias a largo plazo. El impacto producido por la deuda técnica arquitectónica es evidente, por lo que luego de localizarla, los arquitectos de software deben utilizar estrategias de pago con la finalidad de reducir las consecuencias en el diseño del software (Pérez et al., 2019).

En este sentido, identificar y clasificar la deuda técnica es el punto inicial para tratar de controlarla, muchos desarrolladores se centran en la búsqueda de defectos en la codificación para lo cual utilizan varias técnicas y herramientas automatizadas. También, es importante admitir la deuda técnica y utilizar como técnica escribir comentarios en el código fuente para describirla, para esto se utiliza la similitud semántica que permite una búsqueda optima de errores (Flisar y Podgorelec, 2019).

La deuda técnica también puede surgir como resultado de tomar decisiones durante el diseño por conveniencia afectando la evolución del software. En esta fase se puede incurrir en diferentes tipos de deuda como es el caso de la arquitectónica, por requerimientos, por codificación, pruebas, etc. Es importante tomar las acciones adecuadas para determinar el momento oportuno en que se debe asumir la deuda (Rosser y Norton, 2021).

Asimismo, el desarrollo de software crítico no está exento de la deuda técnica, este tipo de software por su característica puede ocasionar daños irreversibles en el ámbito, económico, humano o de infraestructuras con cualquier falla. Es por lo que su desarrollo exige altos estándares de control y calidad requiriendo regulaciones críticas en la arquitectura y codificación ya que deben pasar por procesos de certificación antes de entrar en producción. Estos controles limitan el aparecimiento de deuda técnica, pero esta surge en la mantenibilidad, denotando un esfuerzo y costos muy altos para la refactorización y la recertificación (Besker et al., 2019).

Por otro lado, la percepción y gestión de la deuda técnica depende de la metodología utilizada en el desarrollo de software, sin embargo, independientemente del modelo utilizado las técnicas para prevenirla y pagarla son similares. La diferencia de utilizar modelos ágiles, tradicionales o la combinación de los dos se sitúa en como los desarrolladores monitorean y controlan los efectos de la deuda técnica en las fases y tareas del desarrollo, la prevención de la deuda es una actividad que no se puede dejar de lado independientemente de la metodología seleccionada para el desarrollo (Rios et al., 2021).

De ahí que, el desarrollo ágil se caracteriza por el suministro continuo de valor a los clientes con entregas

programadas de productos funcionales. Para cumplir con este objetivo los desarrolladores utilizan varios tipos de arquitecturas como es el caso de los microservicios. Como toda arquitectura, los microservicios también generan deuda técnica por la comunicación y las conexiones entre los servicios siendo necesario una planificación para el pago de la deuda generada (de Toledo et al., 2019).

Dentro de este marco, la falta de documentación técnica es una de las debilidades en los proyectos de desarrollo de Software. Si bien los modelos ágiles requieren de una mínima documentación que beneficia en el ahorro de tiempo, la ausencia o presencia casi nula de documentación dan como resultado una acumulación de la deuda técnica. Los desarrolladores conocen la importancia de documentar por lo que plantean adaptar las prácticas de documentación en las fases distintas fases de desarrollo (Behutiye et al., 2020).

Además, la fase del diseño del software es una de las etapas donde surge la deuda técnica. Una base de datos incorrectamente diseñada por la falta de la aplicación de los criterios formales de normalización introduce deuda técnica en el desarrollo. La calidad en el diseño de la base de datos influye en la calidad y rendimiento de los datos, por lo que la deuda generada por su normalización debe gestionarse con la aplicación de un marco adecuado (Albarak et al., 2020).

Dentro de este orden de ideas, un aspecto importante para tomar en cuenta en el desarrollo de aplicaciones modernas es el de la seguridad, es fundamentar implementar acciones en todas las etapas de desarrollo para blindar la aplicación. El endeudamiento asociado a la seguridad surge desde el diseño arquitectónico hasta la codificación, siendo necesario tomar medidas de seguridad y la aplicación de pruebas exhaustivas para precautelar consecuencias que afecten a la industria del software y los usuarios protegiendo la integridad y confidencialidad de los datos (Martinez et al., 2021).

En cuanto al impacto de la deuda técnica, esta es subestimado por los desarrolladores provocando que el software se encarezca y que se entregue productos de baja calidad. El buscar beneficios a corto plazo en el desarrollo tiene como consecuencias un esfuerzo extra en las acciones correctivas que debe aplicar el equipo para garantizar el funcionamiento adecuado del software. Los problemas de la deuda técnica inician en la definición de requisitos y el diseño, contagiando a la arquitectura, la codificación y la aplicación de las pruebas (Bi et al., 2021).

Con respecto a, la codificación errónea y de mala calidad, son unas de las principales causas para el surgimiento de la deuda técnica, esta deuda que en la mayoría de los casos se da por ahorrar tiempo en el desarrollo se debe pagar muchas veces con intereses cuando el software se encuentra en producción al tener que agregar funcionalidades y características y corregir errores. La cuantificación y medición de la deuda técnica aportan para la gestión de la deuda técnica asegurando un buen nivel de mantenibilidad y el pago a tiempo de la deuda (Nikolaidis et al., 2021).

No obstante, los beneficios obtenidos inicialmente al incurrir en deuda técnica se terminan pagando muy caro y con intereses a largo tiempo. Acumular deuda técnica da como resultado el lanzamiento de un producto de mala calidad y muy por debajo del nivel esperado. Detectar la deuda a tiempo requiere de diferentes estrategias como es el caso de la identificación de code smells y de anti-patrones en el diseño del software. Determinar las estrategias que aporten con técnicas para la eliminación de la deuda contribuirá para que le proyecto sea entregado en los tiempos previstos y de acuerdo con los requerimientos del cliente (Ramirez Lahti et al., 2021).

Como se ha indicado, la deuda técnica se genera por factores internos por lo que se debe gestionarla aplicando mediciones y utilizando herramientas para controlar eficazmente los aspectos técnicos. También los factores externos inciden en la generación de deuda como es el caso de las políticas, normativas y disposiciones del estado. Además, aspectos empresariales como infravalorar la gestión técnica de la deuda, la comunicación inadecuada sobre la estrategia empresarial, cambios en la cultura organizacional y la comunicación inadecuada sobre las tendencias tecnológicas entre otros son componentes críticos que contribuyen a la deuda técnica (Brenner, 2019).

La gestión de la deuda técnica es un proceso fundamental en el desarrollo de Software, sin embargo, no existe un marco formal que sea aceptado y aplicado por los desarrolladores. Si bien existen varios trabajos sobre marcos para la gestión todos estos difieren unos de otros como es el caso del marco basado en el cuerpo de conocimiento de gestión de proyectos PMBOK proporcionando un enfoque empírico para la gestión de la deuda técnica (Ramasubbu y Kemerer, 2019).

En el desarrollo dentro del ámbito universitario, la deuda técnica es evidente por el contexto dinámico de los proyectos de software y debido a la presión de los usuarios por obtener productos funcionales en el menor tiempo posible, esto sumado con la poca experiencia en la gestión de deuda técnica en los equipos desarrollo, desencadena en un producto de baja calidad e imposible de mantener (Alfayez et al., 2018).

Este estudio, utiliza un análisis cualitativo, mediante entrevistas a un equipo de desarrollo de software, explora el conocimiento y la percepción de la deuda técnica, las prácticas que se utilizan para gestionarla, los factores que influyen en su aparecimiento y acumulación, el impacto que sufre el desarrollo, así como el equipo y la mantenibilidad del software, como resultado se proponen ideas y soluciones para abordarla de manera efectiva.

https://doi.org/10.56294/dm2024.233

## MÉTODO

El estudio se efectuó con un enfoque investigativo cualitativo, que se basó en el estudio de un caso único con el cual se indaga las experiencias y prácticas en la percepción y gestión de la deuda técnica. El contexto del estudio se desenvolvió en un ámbito de desarrollo de software para una institución educativa de nivel superior. Se exploran los fenómenos en el campo real para comprender detalladamente el problema.

El equipo de desarrollo objeto del estudio se encuentra compuesto por el director del departamento de desarrollo y cinco desarrolladores. El equipo seleccionado desarrolla software de gestión y académico en la institución de educación superior desde hace quince años atrás, por lo que poseen una amplia experiencia en este campo, siendo los responsables del mantenimiento y creación de todos los sistemas de la universidad.

La herramienta utilizada para la recolección de datos fueron las entrevistas semi-estructuradas, se diseñó una entrevista para el director del departamento y otra entrevista para los cinco desarrolladores con preguntas similares, pero aplicando diferentes enfoques. La duración de la entrevista fue de aproximadamente quince minutos por cada entrevistado y se aplicó en el lugar de trabajo de los entrevistados con el consentimiento del director del departamento y cada uno de los miembros del equipo.

La guía de entrevista se estructuró con temas relacionados al conocimiento, percepción, prácticas actuales, identificación, impacto y gestión de la deuda técnica, relacionadas al contexto de desarrollo en el cual se desempeña el equipo de desarrollo.

En el proceso de validación, se examinaron los datos recolectados, para lo cual se verificaron las transcripciones de las entrevistas, realizando un análisis temático y semántico, asegurando la consistencia y confiabilidad, luego se categorizó y codificó los datos recolectados.

## RESULTADOS

El resultado de los datos obtenidos en las entrevistas aplicadas al equipo de desarrollo de software en la institución de educación superior permitió obtener hallazgos significativos en el marco de la percepción e identificación de la deuda técnica, estos se presentan a continuación.

### Información de los entrevistados

El director del departamento de desarrollo también cumple las funciones de jefe del equipo. Inicialmente trabajó como desarrollador externo a la institución y en la actualidad ya forma parte de la universidad. Sus principales funciones son: la planificación y estrategia, coordinación y gestión de proyectos, el liderazgo del equipo, la colaboración interdepartamental, el control de calidad, la gestión de presupuesto y asegurar que todos los desarrollos de software cumplan con las normativas solicitadas por la institución y sus usuarios.

El equipo de desarrollo se encuentra compuesto por profesionales titulados como Ingenieros en Sistemas e Informática con experiencia en el desarrollo de software que va desde uno a doce años. Las principales funciones de los miembros del equipo son las siguientes: el desarrollo colaborativo, la revisión y mejora continua, la comunicación constante, la entrega incremental, la generación de documentación y el testeo.

### Conocimiento y percepción de la deuda técnica

El equipo tiene un conocimiento no técnico de la deuda técnica, sin embargo, lo percibe en la mayoría de las tareas que realizan en el desarrollo de software. En la tabla 1 se observa la clasificación de las respuestas obtenidas.

**Tabla 1.** Conocimiento y percepción de la deuda técnica

| Tema | Código | Categoría |
|---|---|---|
| Conocimiento de la deuda técnica. | MET-Ágil | Metodologías ágiles |
| | DT-Riesgo | Definición |
| | DT-Complejidad | |
| | DT-Compromete la calidad | |
| Percepción de la deuda técnica. | DT-Impacto | Impacto |
| | DT-Estrés | |
| | DT-Motivación | |
| | DT-Ineficiencia | |

**Fuente:** Entrevista aplicada al equipo de desarrollo

### Prácticas Actuales

El equipo no utiliza una metodología ágil específica, pero aplican los principios de agilidad en el desarrollo. No obstante, en todo momento emplean posibles soluciones para problemas cotidianos como estimaciones

https://doi.org/10.56294/dm2024.233

incorrectas o trabajo extra para corregir problemas. El análisis y codificación relacionadas a esta pregunta se observa en la tabla 2.

**Tabla 2.** Prácticas Actuales

| Tema | Código | Categoría |
|---|---|---|
| Identificación y manejo de la deuda técnica. | MET-Ágil | Metodologías ágiles |
| | MET-Adaptación | |
| | MET-Priorización | |
| Procesos para abordar la deuda técnica. | DT-Identificación | Gestión |
| | DT-Gestión | |
| | DT-Estimación | |
| | DT-Priorización | |

**Fuente:** Entrevista aplicada al equipo de desarrollo

**Impacto y Consecuencias**

El impacto en el equipo se evidencia el momento de los cambios en los requerimientos por el desconocimiento técnico de los usuarios afectando en el normal avance del desarrollo de los proyectos, en la tabla 3 se analizan las respuestas obtenidas por los entrevistados.

**Tabla 3.** Impacto y Consecuencias

| Tema | Código | Categoría |
|---|---|---|
| Situaciones en que la deuda técnica ha impactado. | DT-Factores-Externos | Causas |
| | DT-Cambios-Requisitos | |
| | DT-Presión-Usuarios | |
| Afectación en la eficiencia y calidad del desarrollo. | DT-Impacto | Impacto |
| | DT-Moral-Equipo | |
| | DT-Creatividad | |
| | DT-Repetición-Problemas | |

**Fuente:** Entrevista aplicada al equipo de desarrollo

**Soluciones**

Identificar y gestionar a tiempo la deuda técnica permitiría obtener código fuente de calidad y entregas a tiempo del producto. Estás prácticas se derivan en la implementación de estrategias para corregir errores y controlar la deuda a tiempo. En la tabla 4 se presentan los resultados y codificación de los datos obtenidos en la entrevista.

**Tabla 4.** Soluciones

| Tema | Código | Categoría |
|---|---|---|
| Prácticas a implementarse para reducir la deuda técnica. | DT-Moral | Impacto |
| | DT-Productividad | |
| | DT-Frustración | |
| | DT-Desmotivación | |
| | DT-Estrés | |
| | DT-Innovación | |
| Herramientas o procesos para gestionar la deuda técnica. | DT-Medidas | Soluciones |
| | DT-Estimaciones | |
| | DT-Mediciones | |
| | DT-Guía | |

**Fuente:** Entrevista aplicada al equipo de desarrollo

## DISCUSIÓN

Este estudio contribuye con resultados que permiten un mejor conocimiento de la deuda técnica en el desarrollo de software en entornos académicos y de gestión de instituciones de educación superior.

https://doi.org/10.56294/dm2024.233

Los hallazgos presentados revelan que la deuda técnica se encuentra presente en estos contextos de desarrollo, sin embargo, el equipo de desarrollo no tiene un conocimiento generalizado de las implicaciones de no asumir adecuadamente la deuda técnica.

En la actualidad existen muchas herramientas de apoyo a los desarrolladores para generar una codificación eficiente, no obstante, el equipo de desarrollo estudiado no utiliza ninguna de estas herramientas que le permitirían ahorrar sobrecarga en el trabajo y costos extras. En el estudio desarrollado por (Haertel et al., 2022), señala que determinar las herramientas automatizadas de análisis estático adecuadas optimiza la generación de código sin errores y minimiza la deuda técnica.

Si bien, el equipo de desarrollo está consciente que en algunos casos la codificación es pobre, no poseen actividades para controlar los code smells, estos olores son un sinónimo de codificación compleja o mal estructurada que a la larga genera problemas en las aplicaciones. Los autores (Boutaib et al., 2020), confirman que los code smells afectan la estructura de los datos existiendo varios tipos de olores que demandan ser identificados y corregidos técnicamente.

Por otro lado, el equipo de desarrollo trabaja con principios ágiles, sin adoptar un modelo específico, lo que produce que no tengan el tiempo para aplicar métricas que permitan tomar decisiones adecuadas con el surgimiento de algún tipo de problemas. El no utilizar métricas se refleja en una mala codificación que implica el uso obligado de refactorización para minimizar el impacto de la deuda (Iammarino et al., 2021).

Se observa que, aunque el equipo trabaja varios años en el ámbito educativo, los problemas internos por la desorganización en la gestión de la deuda técnica acrecientan su acumulación limitando el desarrollo de nuevos proyectos y por consiguiente el descontento de los miembros de la organización educativa. En la investigación desarrollada por (Sundelin, Gonzalez-Huerta, & Wnuk, 2020), destacan el trabajo extra que los desarrolladores deben realizar para pagar la deuda y si no se la realiza a tiempo genera interés que a la larga va a imposibilitar eliminar la deuda técnica.

El problema de la deuda técnica se encuentra latente en el equipo de desarrollo, afectando la moral del equipo al no poder avanzar con el cumplimiento de las tareas asignadas, generando frustración y estrés en los desarrolladores. Entregar el producto de acuerdo con el cronograma es el principal objetivo en el desarrollo de software como lo señalan los autores (Huang, Shihab, Xia, Lo, & Li, 2017), al realizarlo se cumple con el presupuesto y los requerimientos del cliente.

Los problemas descritos son una bomba de tiempo que requiere un marco formal para gestionar la deuda técnica. Un proceso estructurado abordará la deuda técnica proactivamente y previniendo trabajo y recursos extras que es un limitante en el equipo de desarrollo. En este sentido (Rios, y otros, 2021) señalan que gestionar la deuda técnica y la adopción de buenas prácticas garantizan el control de la deuda y la calidad del software.

## CONCLUSIONES

Este estudio exploró la percepción de la deuda técnica y las practicas utilizadas para gestionarla, por medio de una investigación cualitativa aplicando entrevistas a un equipo de desarrollo de una institución de educación superior. Se analizaron los factores para el surgimiento de la deuda técnica, loe elementos que aportan para su acumulación y el impacto en el equipo y la organización,

Los datos obtenidos revelan que la deuda técnica afecta el normal desenvolvimiento del equipo de desarrollo y que se encuentra presente en las actividades que realiza el equipo. Si bien el equipo de desarrollo tiene una percepción de la deuda no la aborda de manera directa siendo necesario implementar buenas prácticas para identificarla e integrarla en el proceso de desarrollo de software.

Aplicar estrategias para medir, monitorear y controlar la deuda técnica, con el uso adecuado de técnicas y herramientas de análisis de la deuda contribuirán en el desarrollo organizacional, la satisfacción de los miembros del equipo y el cumplimiento óptimo con los requerimientos de los usuarios.

Es fundamental la aplicación de prácticas formales para la gestión de la deuda técnica, se deben establecer estrategias para identificar, medir, monitorear y controlar la deuda, mediante una combinación de herramientas técnicas y procesos adaptados que abarque a todos los interesados. Estas acciones contribuirán en mejorar la calidad del Software aportando en la eficiencia del equipo, la satisfacción de los usuarios, y el desarrollo de la institución

Abordar la deuda técnica con la implementación de un proceso formal para gestionarla logrará que se controle el tiempo programado y se optimicen los recursos, lo que se derivará en una cultura de buenas prácticas que asegurará la calidad del producto mejorando el rendimiento del equipo y garantizando la mantenibilidad del software.

## REFERENCIAS

1. Albarak, M., Bahsoon, R., Ozkaya, I., & Nord, R. (2020). Managing Technical Debt in Database Normalization. IEEE Transactions on Software Engineering, 48(3), 755-772. https://doi.org/10.1109/TSE.2020.3001339

2. Alfayez, R., Behnamghader, P., & Sriso, K. (2018). An exploratory study on the influence of developers in technical debt. Proceedings of the 2018 International Conference on Technical Debt, 1-10. https://doi.org/10.1145/3194164.31941

3. Behutiye, W., Rodríguez, P., Oivo, M., Aaramaa, S., Partanen, J., & Abhervé, A. (2020). How agile software development practitioners perceive the need for documenting quality requirements: a multiple case study. Euromicro Conference on Software Engineering and Advanced Applications (SEAA)(93-100). https://doi.org/10.1109/SEAA51224.2020.00025

4. Besker, T., Martini, A., & Bosch, J. (2019). How Regulations of Safety-Critical Software Affect Technical Debt. Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 74-81. https://doi.org/10.1109/SEAA.2019.00020

5. Bi, F., Vogel-Heuser, B., & Xu, L. (2021). Frequency and Impact of Technical Debt Characteristics in Companies Producing Mechatronic Products. IEEE/ACM International Conference on Technical Debt (TechDebt), 26-35. https://doi.org/10.1109/TechDebt52882.2021.00012

6. Boutaib, S., Bechikh, S., Palomba, F., Elarbi, M., Makhlouf, M., & Said, L. (2020). Code smell detection and identification in imbalanced environments. Expert Systems With Applications. https://doi.org/https://doi.org/10.1016/j.eswa.2020.114076

7. Brenner, R. (2019). Balancing Resources and Load: Eleven Nontechnical Phenomena that Contribute to Formation or Persistence of Technical Debt. IEEE/ACM International Conference on Technical Debt (TechDebt), 38-47. https://doi.org/10.1109/TechDebt.2019.00013

8. de Toledo, S., Martini, A., Przybyszewska, A., & Sjøberg, D. (2019). Architectural Technical Debt in Microservices. A case study in a large company. IEEE/ACM International Conference on Technical Debt (TechDebt), 78-87. https://doi.org/DOI 10.1109/TechDebt.2019.00026

9. Digkas, G., Chatzigeorgiou, A., Ampatzoglou, A., & Avgeriou, P. (2022). Can Clean New Code Reduce Technical Debt Density? IEEE Transactions on Software Engineering, 48, 1705-1721. https://doi.org/10.1109/TSE.2020.3032557

10. Flisar, J., & Podgorelec, V. (2019). Identification of Self-Admitted Technical Debt Using Enhanced Feature Selection Based on Word Embedding. IEEE Access, 7, 106475-106494. https://doi.org/10.1109/ACCESS.2019.2933318

11. Haertel, C., Pohl, M., Staegemann, D., & Turowski, K. (2022). A Method for Comparing and Selecting Static Code Analysis Tools in Software Development Projects. Research Square. https://doi.org/http://dx.doi.org/10.21203/rs.3.rs-1546855/v1

12. Huang, Q., Shihab, E., Xia, X., Lo, D., & Li, S. (2017). Identifying self-admitted technical debt in open source projects using text mining. Empir Software Eng, 1-34. https://doi.org/10.1007/s10664-017-9522-4

13. Iammarino, M., Zampetti, F., Aversano, L., & Di Penta, M. (2021). An empirical study on the co-occurrence between refactoring actions and Self-Admitted Technical Debt removal. The Journal of Systems & Software, 178. https://doi.org/10.1016/j.jss.2021.110976

14. Martinez, J., Quintano, N., Ruiz, A., Santamaria, I., Martinez de Soria, I., & Arias, J. (2021). Security Debt: Characteristics, Product Life-Cycle Integration and Items. IEEE/ACM International Conference on Technical Debt (TechDebt), 1-5. https://doi.org/10.1109/TechDebt52882.2021.00009

15. Nikolaidis, N., Zisis, D., Ampatzoglou, A., Chatzigeorgiou, A., & Soudris, D. (2021). Experience With Managing Technical Debt in Scientific Software Development Using the EXA2PRO Framework. IEEE Access, 9, 72524-72534. https://doi.org/10.1109/ACCESS.2021.3079271

16. Pérez, B., Correal, D., & Astudillo, H. (2019). A Proposed Model-driven Approach to Manage Architectural Technical Debt Life Cycle. International Conference on Technical Debt (TechDebt), 73-77. https://doi.org/10.1109/TechDebt.2019.00025

17. Ramasubbu, N., & Kemerer, C. (2019). Integrating Technical Debt Management and Software Quality Management Processes: A Normative Framework and Field Tests. IEEE Transactions on Software Engineering, 45(3), 285-300. https://doi.org/10.1109/TSE.2017.2774832

18. Ramirez Lahti, J., Tuovinen, A.-P., & Mikkonen, T. (2021). Experiences on Managing Technical Debt with Code Smells and AntiPatterns. IEEE/ACM International Conference on Technical Debt (TechDebt), 36-44. https://doi.org/10.1109/TechDebt52882.2021.00013

19. Reboucas de Almeida, R. (2019). Business-Driven Technical Debt Prioritization. IEEE International Conference on Software Maintenance and Evolution (ICSME), 605-609. https://doi.org/10.1109/ICSME.2019.00096

20. Rios, N., Freire, S., Pérez, B., Castellanos, C., Correal, D., Mendonça, M., . . . Spínola, R. (2021). On the Relationship between Technical Debt Management and Process Models. IEEE Computer Society, 38(5), 56-64. https://doi.org/10.1109/MS.2021.3058652

21. Rios, N., Mendonça, M., Freire, S., Falessi, D., Pérez, B., Izurieta, C., . . . Spínola, R. (2021). On the Relationship between Technical Management and Process Models. IEEE Computer Society.

22. Rosser, L., & Norton, J. (2021). A Systems Perspective on Technical Debt. IEEE Aerospace Conference, 1-10. https://doi.org/10.1109/AERO50100.2021.9438359

23. Stochel, M., Chołda, P., & Wawrowski, M. (2020). On Coherence in Technical Debt Research. Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 367-375. https://doi.org/10.1109/SEAA51224.2020.00067

24. Sundelin, A., Gonzalez-Huerta, J., & Wnuk, K. (2020). The Hidden Cost of Backward Compatibility: When Deprecation Turns into Technical Debt - An Experience Report. 2020 IEEE/ACM International Conference on Technical Debt, 67-76.

## FINANCIACIÓN

Ninguna.

## CONFLICTO DE INTERESES

Los autores declaran que no existe conflicto de intereses.

## CONTRIBUCIÓN DE AUTORÍA

*Conceptualización:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Curación de datos:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Análisis formal:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Investigación:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Metodología:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Administración del proyecto:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Recursos:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Software:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Supervisión:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Validación:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Visualización:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Redacción – borrador original:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.
*Redacción – revisión y edición:* Edwin Fabricio Lozada Torres, Rodrigo Cadena Martínez, María Angélica Pico Pico.