Synergía ISSN L 2953-2973 Vol.4, No.1 mayo - octubre 2025
https://revistas.up.ac.pa/index.php/synergia

Synergía pp.349-363

Transparencia en DevOps: estrategia para fortalecer la cultura colaborativa y calidad en equipos de TIC

Transparency in DevOps: approach to enhance collaborative culture and improve software quality in ICT Teams.

Luis Domínguez Quintero

Universidad de Panamá, Facultad de Informática, Electrónica y Comunicación, Panamá.
luis.dominguezq@up.ac.pa https://orcid.org/0000-0002-3309-6745

Doris Z. Pinzón Castillo

Universidad de Panamá, Facultad de Informática, Electrónica y Comunicación. Panamá.
doris.pinzon@up.ac.pa https://orcid.org/0009-0008-6521-783X

Saily M. González Aguilar

Universidad de Panamá, Centro de Innovación Desarrollo Tecnológico y Emprendimiento, Panamá. saily.gonzalez@up.ac.pa https://orcid.org/0009-0001-3168-3161

*Autor de correspondencia: (luis.dominguezq@up.ac.p

Fecha de recepción: 28/02/2025 Fecha de aceptación: 24/04/2025

DOI: https://doi.org/10.48204/synergia.v4n1.7198

Resumen

Desarrollo y operaciones (DevOps) es un conjunto de principios y prácticas que mejora la colaboración entre los equipos de desarrollo y operaciones, agilizando el ciclo de vida del desarrollo de software mediante la automatización. La transparencia se considera un elemento clave para optimizar los procesos y fomentar la colaboración entre los equipos. Esta transparencia permite un acceso abierto a la información y facilita una comunicación efectiva, lo cual contribuye a mejorar la calidad del software y la toma de decisiones.
El objetivo de este documento es analizar el papel de la transparencia en DevOps como factor clave para mejorar la cultura colaborativa y la calidad del software en los equipos de TIC. Se emplea para este análisis una revisión documental basado en la técnica de fichaje con el que se consintió la

identificación de perspectivas de diversos autores sobre cómo la transparencia optimiza las revisiones y resoluciones tempranas de problemas.
Como resultado encontramos que los diversos autores manifiestan que la transparencia facilita los procesos en los diferentes ciclos de vida del desarrollo del software al permitir resolver problemas con anticipación, pero también resalta la necesidad de implementar métricas para acceder a la información, ya que el no hacerlo puede ser una desventaja más que una ventaja.

Palabras clave: calidad, DevOps, comunicación, toma de decisiones, transparencia.

Abstract

Development and Operations (DevOps) is a set of principles and practices that enhance collaboration between development and operations teams, streamlining the software development lifecycle through automation. Transparency is key in optimizing processes and fostering cooperation among teams. This transparency allows open access to information and facilitates effective communication, contributing to improved software quality and decision-making.
The objective of this document is to analyze the role of transparency in DevOps as a key factor in strengthening collaborative culture and enhancing software quality in ICT teams. This analysis is based on a documentary review using the indexing technique, which enabled the identification of various authors' perspectives on how transparency optimizes auditing, reviews, and early problem resolution.
As a result, we found different authors affirm that transparency facilitates processes across various stages of the software development lifecycle, allowing problems to be addressed in advance. However, they also highlight the need to implement metrics to access information, as failing to do so could turn transparency into a disadvantage rather than an advantage.

Keywords: quality, DevOps, communication, decision-making, transparency.

Introducción

DevOps es un conjunto de principios y prácticas que permite aumentar la colaboración entre el personal de desarrollo y el de operaciones. Para Millstein (2020) DevOps, es un movimiento cuyo objetivo principal es incorporar infraestructuras ágiles al proceso de desarrollo de software. Ello es posible por la automatización de la mayoría de los procesos involucrados con el desarrollo del software y minimizando la interacción o eliminando el trabajo manual realizado por los departamentos de desarrollo y operaciones (Guerriero et al., 2019). Lo anterior no sería viable sin la existencia de procesos que transparentan la comunicación y colaboración entre ambos equipos.

El término transparencia es asociado a procesos relacionados con el acceso de información resultante y lo que acontece en estos. Cuando los usuarios de un sistema de tecnología de información (TI),

tienen el interés en ser informados sobre cómo el sistema gestiona los datos y, en particular, sus datos personales, la transparencia garantiza una política abierta sobre el funcionamiento y el procesamiento de la información del sistema.

Las organizaciones competitivas de hoy en día implementan DevOps para agilizar sus procesos de desarrollo y despliegue rápido de software y, poco a poco, comienzan a entender la importancia que conlleva la transparencia en los procesos DevOps. No existe una estrategia clara de cómo poder implementarla dentro de una organización. Algunos autores como Luz *et al.*(2019), creen que una adecuada estrategia de cultura colaborativa ayuda en los procesos de adopción y automatización de DevOps, pues ello permite tener un lenguaje en común entre todos los departamentos, ya que si no se realiza, este afecta el desempeño de los equipos de TI y no se podrían implementar de manera adecuada DevOps. Pero, aun así, es tema que se encuentra en proceso de definición.

DevOps

DevOps se rige por un conjunto de principios que guían el ciclo de vida de un software, los cuales son conocidos por sus abreviatura CALMS (Wiedemann et al., n.d.): (a) Culture: Integración de mutua confianza, disponibilidad de aprender, mejoras continuas, constante flujo de información, mentalidad abierta a cambios y experimentación entre desarrolladores y operaciones. (b) Automation: Implementación de los flujos de deployment con altos niveles de automatización (Despliegue Continuo, Integración Continua) y automatización de las pruebas. (c) Lean: Aplicación de principios de Lean como la minimización del trabajo en procesos, como el acortamiento y amplificación de los ciclos de retroalimentación para identificar y minimizar los flujos de quiebre con el fin de incrementar la eficiencia. (d) Measurement: Monitoreo de las métricas, tales como las relacionadas con el negocio o las métricas de transacciones y otros indicadores principales de desempeño. (e) Sharing: El conocimiento en la organización, a través de sus límites organizacionales. Los miembros del equipo deberían aprender de las experiencias de los otros y comunicarlas de forma proactiva.

Hall (2024), demuestra que DevOps implica responsabilidades compartidas. Los equipos de desarrollo y operaciones son responsables del éxito o el fracaso de un producto. Se espera que los desarrolladores no solo crearán el producto y lo proporcionarán al equipo de operaciones, sino que también compartirán la responsabilidad de supervisarlo durante toda su vida, adoptando la

mentalidad de "lo que creas, lo gestionas", por tanto es probable que un desarrollador permita conocer cómo implementar una infraestructura de servidores, optimizaciones y configuraciones adecuadas para que funcione un software en una infraestructura determinada; de la misma forma, es posible que el equipo de operaciones desconozca todas los fases y aspectos necesarios de las etapas de programación. Por tal razón, el objetivo de la infraestructura DevOps es la de ser una cultura de constante cambio que promueva la comunicación entre los equipos, orientados a tener presente y entender: cuánto los factores humanos pueden incidir en el desarrollo.

Es fundamental que los métodos de ingeniería de requisitos integren los procesos empresariales, favorezcan la comunicación entre distintos roles y fomenten la reutilización de requisitos en ciclos de trabajo continuos (Hernández *et al.*, 2023). Para desarrollar productos alineados con las expectativas y necesidades de los clientes, se requiere una planificación, construcción, operación, implementación y evaluación constantes, lo que facilita la toma de decisiones y la identificación de nuevas oportunidades de negocio (Barcellos, 2020; Silveira *et al.*, 2022)

Desde la perspectiva del análisis de requerimientos, en lo referente a las herramientas de comunicación, es recomendable que estén alineados con metodologías ágiles. Sin embargo, si en una organización no se han implementado dichas metodologías, es posible iniciar la estandarización de la comunicación mediante el uso de herramientas automatizadas. Según Collins, (2017) existen herramientas automatizadas utilizadas dentro de los procesos de desarrollo de software, que pueden servir como un primer acercamiento para comenzar a estandarizar estos mecanismos de comunicación y transparencia entre los equipos, por ejemplo, una de las herramientas más utilizadas en el desarrollo de software actualmente, para automatizar muchas de las tareas que allí acontecen, son las herramientas de infraestructura como código dice Rahman *et al.* (2019), las cuales para Collins constan de un lenguaje de dominio específico (LDE) en el que se especifican las tareas y fases utilizadas en el desarrollo y despliegue de las aplicaciones o utilizar sistemas de tickets. Si bien estos no es una solución ideal es una primera aproximación para generar comunicación y colaboración.

La cultura colaborativa es el elemento esencial para la adopción de DevOps dentro de una organización y esta debe ser fomentada dentro de los equipos técnicos, desde el primer día que se comienza a desarrollar el software, con el fin de reducir los retrasos y conflictos que normalmente se darán en los equipos durante las constantes tareas de administración, configuración, provisionamiento de la infraestructura como en el despliegue y entrega del software, por lo que

además de fomentar la comunicación, compartir las responsabilidades, ayuda a dilucidar y resolver los problemas que se den durante el ciclo de vida del software (Collins, 2017), ya que lo importante no es hallar culpable, si llega a pasar algo, sino que entre todos se busca generar ideas que permitan resolver los diferentes retos y problemas que se irán presentando, ya sea realizando entrenamiento, tomando charlas técnicas, documentación y todo canal de comunicación que permita compartir conocimiento e información a lo largo de los procesos. Los integrantes del equipo aportan su experiencia en áreas específicas de la operación de carga de trabajo, pero deben gestionar de manera autónoma las tareas diarias y de emergencia, con apoyo externo cuando sea necesario. Además, tienen el deber de cumplir con los lineamientos organizacionales y fomentar la colaboración con otros equipos, promoviendo una cultura de conocimiento compartido (Microsoft, 2024).

En la Tabla 1, se muestran principios en los que se consideran deben estar basados la cultura colaborativa.

Tabla 1.

Principios de éxitos identificados para llevar una cultura colaborativa del DevOps

Cultura colaborativa	Descripción
Comunicación abierta y transparente	La comunicación es la base que mantiene unido a un equipo DevOps. Más que un simple intercambio de información se trata de crear un ambiente donde cada miembro se sienta escuchado y valorado.
Cultura de responsabilidad compartida	El éxito o el fracaso de un proyecto no recae en un solo equipo, sino en todos. Cada miembro aporta su esfuerzo para garantizar la calidad del producto y su correcta implementación.
Mejora continua y aprendizaje	La cultura DevOps se basa en la mejora continua, lo que significa que los equipos no solo deben adaptarse a los cambios, sino también buscar nuevas formas de hacer las cosas mejor cada día.

Fuente: Najarro Gamboa (2024).

La Transparencia

Cuando hablamos de transparencia, comúnmente la asociamos con el acceso a la información y el conocimiento de los algoritmos o procesos de un software como producto, lo cual es fundamental

para garantizar la privacidad y la protección de datos personales (Kaplan, 2020). Sin embargo, la transparencia también actúa como un requisito no funcional (RNF), ya que puede ser expresada como tal dentro de la ingeniería de requisitos en el desarrollo de software. La ingeniería de requisitos proporciona técnicas y herramientas para especificar, modelar, representar, implementar, medir y rastrear, tanto los requisitos funcionales como los no funcionales de un sistema. Estos requisitos no funcionales, como la transparencia, son esenciales para describir y asegurar las propiedades de calidad del sistema en cuestión.

Como la transparencia se comporta como un RNF, la misma puede ser modelada y definida de manera formal, pero debido a sus múltiples dimensiones, se requiere un enfoque diferente al utilizado para otros RNF. Por esta razón, consideramos que aplicar prácticas de la ingeniería de requisitos puede ayudar a modelar la transparencia, consintiendo integrarla como un requisito en el Ciclo de Vida de Desarrollo de Software. El enfoque que se presenta se centra en cómo modelar y validar la transparencia. Modelar la transparencia implica representarla de alguna forma, mientras que validarla significa medir si un sistema ofrece transparencia y hasta qué nivel (Grünewald et al., 2023).

Las métricas utilizadas para los RNF como mantenibilidad, reutilización, confiabilidad o seguridad pueden ser validadas mediante métodos formales. No obstante, estas métricas no definen específicamente la transparencia, lo cual representa un desafío. Aunque existen métricas formales para evaluar confiabilidad, seguridad y redundancia, no hay métricas específicas diseñadas para la transparencia. A raíz de este problema, en la literatura investigada, los autores sugieren que es necesario realizar estudios diversos de los requisitos no funcionales a través de encuestas a empresas que utilizan DevOps, con el fin de identificar en qué etapas del Ciclo de Vida de DevOps surgen problemas relacionados con la transparencia. Esto nos permitiría desarrollar métricas específicas para promover la transparencia en el desarrollo de software (Sánchez & Colomo, 2018) o bien formular estrategias para mitigar los problemas encontrados.

Una de las metodologías más utilizadas para definir métricas de software es el Estándar IEEE 1016, que consta de cinco pasos: 1. Establecer los requisitos; 2. Identificar e implementar las métricas; 3. Analizar los resultados; y 4. Validar las métricas. Nos centraremos en este trabajo en el segundo paso: la identificación de métricas. La transparencia es un concepto muy amplio, y asignar métricas

sería un trabajo muy laborioso, por lo que, basado en el estándar IEEE, primero se deben identificar la calidad de los factores y sub-factores que contribuyen a establecer la transparencia, los cuales son: (1) Informativeness (Informatividad): concierne a la habilidad de transportar una buena calidad de información y entender la excelencia de la información prevista. (2) Understandability (Comprensibilidad): representa la habilidad de que la información tenga un significado comprensible. (3) Accessibility (Accesibilidad): se refiere a categorías de instrumentos que permiten que algo sea fácil de obtener. (4) Validity (Validez) es la capacidad de tener mecanismos que admiten definir que el resultado de algo es correcto y preciso.

Implementación de la transparencia en DevOps

La transparencia, en el contexto de los procesos DevOps, desempeña un papel fundamental en la mejora de la calidad del software. Al garantizar un acceso abierto y comprensible a la información y procesos, las organizaciones pueden crear una cultura de confianza y colaboración entre los equipos de desarrollo y operaciones (Mishra & Otaiwi, 2020). En un entorno transparente, los desarrolladores, el equipo de control de calidad y el personal de operaciones pueden tomar decisiones más informadas, lo que mejora la eficacia y eficiencia en cada etapa del ciclo de vida del software (Lwakatare et al., 2019). Esta cultura colaborativa fomenta la comunicación fluida y la retroalimentación constante, factores claves para detectar problemas y proponer soluciones rápidamente (Khan *et al.*, 2022).

Uno de los aspectos que fortalece la transparencia en DevOps es la implementación de prácticas y métricas que faciliten la accesibilidad y claridad de la información. La adopción de herramientas como la infraestructura, código (IaC) y sistemas de gestión de versiones, garantiza que los miembros del equipo puedan revisar y comprender el estado actual del desarrollo y las configuraciones de infraestructura en todo momento (Rahman, 2018). Además, según estudios previos, la transparencia en la fase de desarrollo permite que los equipos mantengan un enfoque alineado en los objetivos del proyecto y logren entregar un producto de software de mayor calidad (Mishra & Otaiwi, 2020).

En este sentido, la transparencia puede ser vista como un requisito no funcional, ya que su implementación eficaz proporciona una base sólida para evaluar y mejorar los procesos de desarrollo y despliegue. La aplicación de métricas específicas, como la informatividad, la comprensibilidad, la

accesibilidad y la validez, permite modelar y evaluar la transparencia en el contexto de DevOps (Spagnuelo *et al.*, 2020). Por ejemplo, la accesibilidad a la información crítica durante la fase de pruebas garantiza que todos los involucrados tengan una visión clara del rendimiento del software, lo cual es crucial para evitar errores y mejorar la calidad del producto final (Rahman y Williams, 2019).

Para garantizar dicha transparencia, es necesario que todos los equipos de trabajo tengan un lenguaje en común que permita un proceso efectivo de colaboración y automatización, algunos autores como (García-Mireles *et al.*, 2024) sugieren que, dentro del ciclo de vida de desarrollo e implementación del software, se realicen procesos de transparencia en el ciclo de vida del software de DevOps. Por otro lado, cuando hablamos de transparencia en el software, nos referimos a todas las funcionalidades, conocidas y desconocidas del mismo (Grünewald *et al.*, 2023), pero aplicada a los procesos de DevOps, por lo que es necesario realizar una revisión de requerimientos no funcionales que nos lleven a realizar una métrica, que promueva la transparencia en el desarrollo de software (Hernández *et al.*, 2023), pero aplicado a los procesos de integración y despliegue continuo en DevOps. Debido al potencial de poder que ofrece la transparencia se puede incrementar el potencial de colaboración y aprendizaje de todo el conocimiento complejo de los equipos (Azad & Hyrynsalmi, 2023), ya que, si no se establecen mecanismos que promuevan la transparencia en DevOps, en vez de ser un dinamizador del potencial tecnológico de las empresas, sería un causante de problemas.

Por consiguiente, las fases de DevOps comparten muchas similitudes con las etapas del ciclo de vida del desarrollo del software, en lo concerniente a requerimientos no funcionales identificados, los cuales pueden utilizarse para promover la transparencia a lo largo de dicho ciclo, considerando, de esta manera, el software como un producto. Estos también son aplicables a los procesos involucrados en DevOps. En la Figura 1, explicamos cómo la transparencia apoyados en los requerimientos no funcionales potencian una cultura de colaboración y comunicación en las organizaciones en las diferentes fases de DevOps y, por consecuencia, la calidad del software programado o implementado.

La transparencia en DevOps se encuentra soportada en base a un conjunto de RNF, estos son: compresibilidad, informatividad, accesibilidad y validez, los cuales inciden en cada una de las etapas de DevOps, permitiendo así mejorar la colaboración y la comunicación entre los equipos de desarrollo y operaciones, pilares de la cultura colaborativa sobre los que se sustenta DevOps y el éxito de cada una de sus fases. No todos los factores de transparencia identificados se aplican a todas las etapas de

DevOps de igual manera, por lo que explicamos cómo estos factores inciden en cada una de las etapas de DevOps.

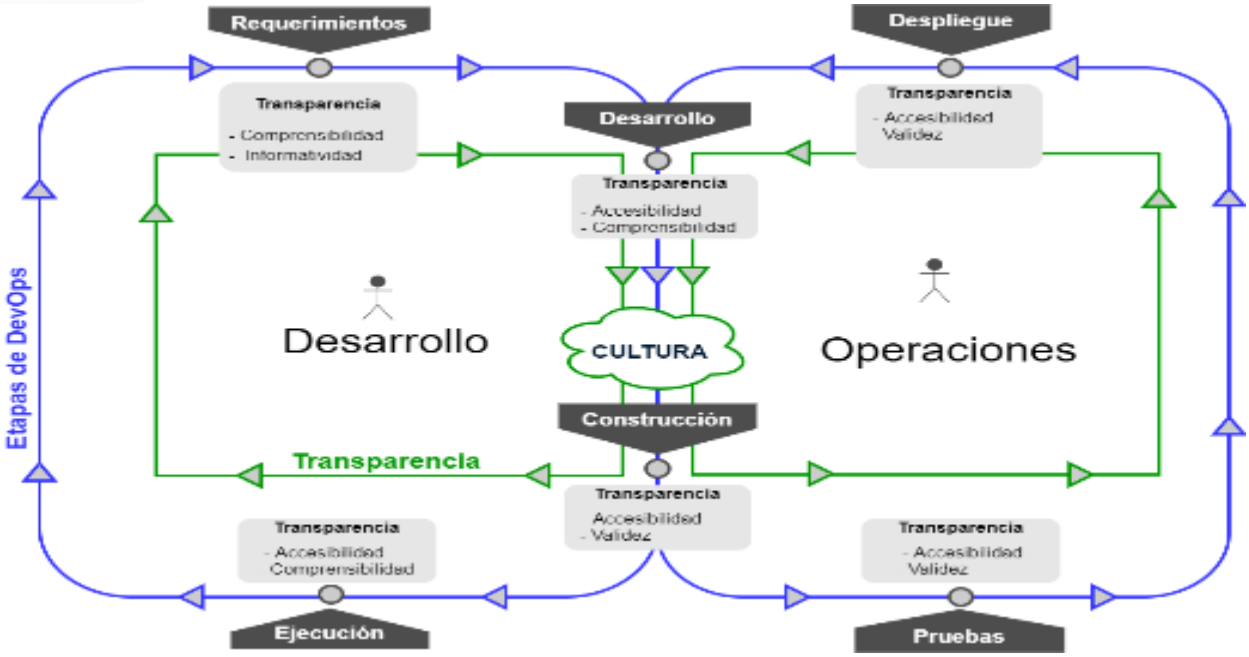

Figura 1.

Funcionamiento de la transparencia en DevOps

Una vez se establecen los Requerimientos (Requirements), el factor de la informatividad busca que, durante la realización de este proceso, se obtenga información precisa, actualizada, imparcial y coherente con la necesidad del cliente y la comprensibilidad en concordancia con la informatividad, permite que durante este proceso se garantice que, en la mayor parte del tiempo, siempre esté disponible la mínima información posible para entender lo que se requiere realizar o qué objetivos se desean alcanzar con suficientes detalles de forma clara y ordenada, utilizando la terminología adecuada, y usando metodologías ágiles.

Una vez los Requerimientos están identificados continuamos con el Desarrollo (Development) que, durante este proceso, la compresibilidad permite que la información recibida en la etapa de requerimiento proporcione a los desarrolladores los suficientes detalles sobre las tareas que deben

codificar de forma clara y ordenada. Además, se deben documentar las librerías, funciones, clases y componentes desarrollados donde la accesibilidad se garantiza mediante un sistema de control de versiones, que unifica y centraliza la evolución de los cambios iterativos de los softwares a nivel del código fuente y los servicios requeridos para su funcionamiento.

Cuando ya se tienen desarrollados algunos componentes del software, entramos en la construcción (Build). En esta etapa, se realizan las primeras compilaciones y pruebas, donde la validez exige que durante este proceso siempre se tiene que proporcionar datos o información sobre los entornos de compilación y construcción, para que la accesibilidad que viene desde la etapa de desarrollo permita ver todas las integraciones de componentes de software, que se dan en esta etapa.

En la etapa de Pruebas (Testing), la accesibilidad proporciona información sobre las diferentes pruebas que se estén realizando, para ver si están acorde con los requerimientos inicialmente establecidos y de manera conjunta con la validez, accede medir si los componentes de software preliminares están alcanzando la meta prevista.

Después que nuestro software ha sido testeado, de acuerdo con los requerimientos, en la etapa de despliegue (Deployment), el factor de la accesibilidad sirve de guía para identificar el desempeño de las tareas en el deployment y que esta información pueda usarse para configurar los entornos de producción y prueba realizados, por lo que se exige que se proporcione los *scripts* de infraestructura como códigos utilizados en el deployment para verificar su validez.

Para una adecuada ejecución (Execution), debemos garantizar la accesibilidad de información del entorno de producción, ya sea mediante la utilización de diversos instrumentos o reportes, con el fin de que la información sea vista en cualquier momento. Además, durante esta etapa, también se realizan actividades de monitoreo del sistema donde también es necesario la accesibilidad, pues exige que siempre esté disponible la información sobre el desempeño del sistema y que esta información sea en formato extraíble, así como la comprensibilidad, ya que es necesario que la información provea suficientes detalles sobre la monitorización y en un formato o notación fácil de entender, con el fin que de forma sencilla y rápida podamos ver los fallos o problemas de desempeño que se presenten en

un sistema. Además, es donde los equipos de desarrollo y operaciones evalúan si durante los procesos de DevOps se cumplió con los requerimientos definidos y planificar futuros cambios.

El propósito del estudio fue hacer una revisión documental basado en textos publicados por diversos medios que nos permiten poder comprender y exponer la importancia de los DevOPs en el proceso de desarrollo de software, llevado a cabo las diversas empresas, permitiendo erradicar el tabú de que cada proceso es independiente el uno del otro y dejando por sentado que la colaboración entre los desarrolladores de los diversos procesos genera más transparencia y viabilidad del producto final. De lo anterior, surge la pregunta de investigación ¿Los DevOPs influyen positivamente en la transparencia del desarrollo adecuado de cada uno de los pasos a seguir en el desarrollo del software?

Materiales y métodos

El estudio se enmarca en una investigación de tipo documental, recopilando información de fuentes bibliográficas y artículos científicos relacionados con DevOps, transparencia, calidad del software, otros. Se empleó la técnica de fichaje con la que se seleccionó y organizaron los datos relevantes, facilitando el análisis entre transparencia y cultura colaborativa.

Para este proceso, se realizó una revisión bibliográfica sistemática utilizando bases de datos reconocidas como web of Sciencie, Scopus, PubMed y Google Scholar. La estrategia de búsqueda incluyó términos claves específicos como aquellos que tenían que ver con el concepto principal del estudio, aspectos específicos del estudio y contexto o aplicación de este, combinado con operadores booleanos (AND, OR, NOT). Se establecieron filtros para restringir los resultados a publicaciones en inglés y español entre rango de años de 2017 a 2024, priorizando estudios con acceso abierto.

Los artículos preseleccionados se evaluaron en 2 etapas: primeramente, se hizo una revisión de títulos y resúmenes, y luego a través de análisis completo de los textos. Se utilizó una matriz de extracción de datos en Microsoft Excel con variables predefinidas como autor, año, metodología y resultados principales. Para garantizar la validez de los estudios que se seleccionaron, se aplicaron

herramientas de evaluación de calidad metodológica como PRISMA y CASP, asegurando un análisis riguroso que responde al objetivo de la investigación.

Resultados y discusión

El análisis comparativo de estudios claves en DevOps, ingeniería continua y cultura organizacional ha permitido identificar patrones comunes y diferencias significativas en su enfoque. En particular, se han analizado investigaciones relevantes sobre factores críticos de éxito en DevOps (Azad & Hyrynsalmi, 2023), modelos de ingeniería continua (Barcello, 2020) y el impacto de la cultura colaborativa en el desarrollo de software (Collins, 2017). Además, se han considerado metodologías para evaluar la calidad del software en entornos DevOps (García-Mireleset al. 2024) y estrategias para garantizar la transparencia en sistemas nativos en la nube (Grünewald *et al.*, 2023).

Este análisis ha revelado puntos de convergencias fundamentales como la importancia de la automatización, la colaboración interdepartamental y la mejora continua. Sin embargo, también se ha identificado enfoques divergentes. Mientras Kaplan (2020) enfatiza la necesidad de regulación y transparencia en los algoritmos utilizados en el desarrollo de software, Lwakatare *et al.*, (2019) exploran cómo estas prácticas son adoptadas en empresas tecnológicas, con un énfasis particular en la implementación pragmática de DevOps.

La transparencia emerge como un factor clave en este análisis comparativo, ya que influye en la confianza organizacional, la calidad del software y la sostenibilidad de las prácticas DevOps. La automatización y la mejora continua pueden generar opacidad sino se establecen mecanismos adecuados de supervisión y auditoría. En este sentido, las estrategias propuestas de Grünewald *et al.* (2023), para mejorar la visibilidad de los procesos en entornos nativos en la nube, cobran especial relevancia. Asimismo, la necesidad de marcos regulatorios y prácticas de documentación clara, como sugiere Kaplan (2020), es fundamental para garantizar que la automatización y la ingeniería continua no comprometan la ética y la responsabilidad en el desarrollo del software.

Aunque DevOps y la ingeniería continua comparten principios esenciales, la forma como se aborda la transparencia dentro de las prácticas de desarrollo no solo mejora la confianza y la calidad del

software, sino que también permite una adopción más responsable y sostenible de estos enfoques en la industria tecnológica.

Conclusiones

La transparencia en DevOps fortalece la colaboración entre equipos, mejora la comunicación y permite que la información fluya de manera abierta en cada fase de desarrollo. Esto crea un entorno de confianza donde las decisiones se toman con mayor seguridad y rapidez.

En términos de cultura colaborativa, compartir información en tiempo real facilita la alineación de objetivos y reduce la fricción entre equipos. Asimismo, se fomenta su trabajo más sincronizado, donde cada miembro entiende su rol y el impacto de sus acciones.

Sobre la calidad del software, la visibilidad de los procesos permite detectar errores antes de que lleguen a producción. Esto no solo reduce costos, sino que mejora la estabilidad y el rendimiento de los productos entregados.

En cuanto a las métricas, la disponibilidad de datos claros facilita la medición del desempeño y la identificación de áreas de mejora. Contar con información precisa permite ajustar procesos de manera ágil y mantener una evolución constante.

Este estudio se basa en un contexto específico, por lo que sus hallazgos pueden no ser aplicables a todas las organizaciones. Además, la implementación de transparencia depende de factores como la cultura empresarial y la tecnología disponible.

Referencias bibliográficas

Azad, N., & Hyrynsalmi, S. (2023). DevOps critical success factors — A systematic literature review. Information and Software Technology, 157, 107150. https://doi.org/10.1016/j.infsof.2023.107150

Barcellos, M. P. (2020). Towards a framework for continuous software engineering. ACM International Conference Proceedings Series, 626–631.

https://doi.org/10.1145/3422392.3422469

Collins, K. H. (2017). Building a collaborative culture. En R.I. Desourdis & K.H. Collins (Eds.), Human Collaboration in Homeland Security (DVD Included) (pp. 23–34.24). Nova Sciencie Publishers.

García-Mireles, G. A., Olivero, N. P., & Himer Avila-George. (2024). DevOps y la Medición de la Calidad del Producto de Software: Hallazgos Preliminares. RISTI - Revista Ibérica de Sistemas E Tecnologias de Informação, 53, 37–52. https://doi.org/10.17013/risti.53.37-52

Grünewald, E., Kiesel, J., Akbayin, S.-R., & Pallas, F. (2023). Hawk: DevOps-driven Transparency and Accountability in Cloud Native Systems. ArXiv (Cornell University). https://doi.org/10.48550/arxiv.2306.02496

Guerriero, M., Garriga, M., Tamburri, D. A., & Palomba, F. (2019). Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry. Proceedings - 2019 IEEE International Conference on Software Maintenance and Evolution, ICSME 2019, 580–589. https://doi.org/10.1109/ICSME.2019.00092

Hall, T. (2024). Cultura de DevOps. Atlassian. Recuperado de https://www.atlassian.com/es/devops/what-is-devops/devops-culture

Hernández, R., Moros, B., & Nicolás, J. (2023). Requirements management in DevOps environments: A multivocal mapping study. Requirements Engineering. https://doi.org/10.1007/s00766-023-00396-w

Kaplan, B. (2020). Seeing through health information technology: The need for transparency in software, algorithms, data privacy, and regulation. Journal of Law and the Biosciences, 7(1), lsaa062. https://doi.org/10.1093/jlb/lsaa062

Khan, M. S., Khan, A. W., Khan, F., Khan, M. A., & Whangbo, T. K. (2022). Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review. IEEE Access, 10, 14339–14349. https://doi.org/10.1109/ACCESS.2022.3145970

Luz, W. P., Pinto, G., & Bonifácio, R. (2019). Adopting DevOps in the Real World: A Theory, a Model, and a Case Study. Journal of Systems and Software. https://doi.org/10.1016/j.jss.2019.07.083

Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M., & Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. Information and Software Technology, 114, 217–230. https://doi.org/10.1016/j.infsof.2019.06.010

Microsoft. (2024). Recomendaciones para fomentar la cultura de DevOps. Microsoft Learn. https://learn.microsoft.com/es-es/azure/well-architected/operational-excellence/devops-culture

Millstein, F. (2020). DevOps: Paquete de 2 libros: Manual de DevOps y Adopción de DevOps. [s.n.].

Mishra, A., & Otaiwi, Z. (2020). DevOps and software quality: A systematic mapping. Computer Science Review, 38, 100308. ScienceDirect. https://doi.org/10.1016/j.cosrev.2020.100308

Najarro Gamboa, C. (2024). Cultura y mentalidad DevOps: Fomentando la colaboración y la comunicación. LinkedIn. https://www.linkedin.com/pulse/cultura-y-mentalidad-devops-fomentando-la-cesar-najarro-gamboa-bvipe/

Rahman, A. (2018). Anti-Patterns in Infrastructure as Code. Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation, ICST 2018, 434–435. https://doi.org/10.1109/ICST.2018.00057

Rahman, A., Mahdavi-Hezaveh, R., & Williams, L. (2019). A systematic mapping study of infrastructure as code research. Information and Software Technology, 108(November), 65–77.

Rahman, A., & Williams, L. (2019). Source code properties of defective infrastructure as code scripts. Information and Software Technology, 112(May), 148–163. https://doi.org/10.1016/j.infsof.2019.04.013

Sánchez-Gordón, M., & Colomo-Palacios, R. (2018). Characterizing DevOps culture: A systematic literature review. Communications in Computer and Information Science, 918, 3–15. https://doi.org/10.1007/978-3-030-00623-5_1

Silveira, C., Santos, V., Reis, L., & Mamede, H. (2022). CRESustain: Approach to include sustainability and creativity in requirements engineering. Journal of Engineering Research and Science, 1(8), 27–34. https://doi.org/10.55708/js0108004

Spagnuelo, D., Bartolini, C., & Lenzini, G. (2020). Qualifying and measuring transparency: A medical data system case study. Computers and Security, 91, 101717. https://doi.org/10.1016/j.cose.2020.101717

Wiedemann, B. Y. A., Forsgren, N., & Wiesche, M. (n.d.). *Research for practice: The DevOps phenomenon*. Communications of the ACM. https://cacm.acm.org/practice/research-for-practice-the-devops-phenomenon/