La organización de equipos de software en la búsqueda de la entrega continua: un enfoque teórico fundamentado

ABSTRACTO

Contexto: Para acelerar el tiempo de comercialización y mejorar la satisfacción del cliente, las organizaciones productoras de software han adoptado prácticas de entrega continua, lo que afecta las relaciones entre los profesionales del desarrollo y la infraestructura. Sin embargo, ninguna literatura sustancial ha abordado sustancialmente cómo la industria del software estructura la organización de los equipos de desarrollo e infraestructura.

Objetivo: En este estudio, investigamos cómo las organizaciones productoras de software estructuran sus equipos de desarrollo e infraestructura, específicamente cómo es la división del trabajo entre estos grupos y cómo interactúan.

Método: Después de una lluvia de ideas con 7 expertos de DevOps para formular mejor nuestra investigación y procedimientos, recopilamos y analizamos datos de 37 entrevistas semiestructuradas con profesionales de TI, siguiendo las pautas de Grounded Theory.

Resultados: Después de un análisis cuidadoso, identificamos cuatro estructuras organizativas comunes: (1) departamentos aislados, (2) DevOps clásicos, (3) equipos multifuncionales y (4) equipos de plataforma. También observamos que algunas empresas están haciendo la transición entre estas estructuras.

Conclusión: La principal aportación de este estudio es una teoría en forma de taxonomía que organiza las estructuras encontradas junto con sus propiedades. Esta teoría podría guiar a los investigadores y profesionales a pensar en cómo estructurar mejor a los profesionales de desarrollo e infraestructura en las organizaciones productoras de software.

1. INTRODUCCIÓN

Para seguir siendo competitivas, muchas corporaciones productoras de software buscan acelerar sus procesos de lanzamiento [1,2]. Las organizaciones pueden adoptar prácticas de entrega continua en su búsqueda por acelerar el tiempo de comercialización y mejorar la satisfacción del cliente [3]. Sin embargo, la entrega continua también conlleva desafíos, incluidos profundos impactos en varios aspectos de la práctica de ingeniería de software [4]. Con una canalización de implementación automatizada, uno puede, por ejemplo, cuestionar el papel de un ingeniero responsable únicamente de las nuevas implementaciones. Dado que las actividades de lanzamiento involucran muchas divisiones de una empresa (por ejemplo, desarrollo, operaciones y negocios), la adopción de la entrega continua afecta la estructura organizativa [3]. Dadas las transformaciones recientes, existe la necesidad de comprender mejor las estructuras organizativas

que la industria del software adopta para los empleados de desarrollo e infraestructura.1 Por estructura organizativa, nos referimos a la diferenciación (división del trabajo) y la integración (interacción) [5] de las actividades de operaciones (implementación de aplicaciones, configuración de infraestructura y operación de servicios en tiempo de ejecución) entre los grupos de desarrollo y operaciones. Sin embargo, no hay literatura sustancial que aborde cómo las organizaciones han estructurado sus grupos de desarrollo y operaciones. La literatura existente presenta algunas clasificaciones para las estructuras organizativas [6–10]. Aún así, la mayoría de estos estudios no se basan en evidencia empírica, lo que limita la comprensión de cómo los autores concibieron sus clasificaciones. Una excepción es el trabajo de Shahin et al. [10], cuyo enfoque fue comprender cómo las organizaciones organizan los equipos de desarrollo y operaciones para adoptar prácticas de entrega continua de manera óptima. Sin embargo, nuestra búsqueda no se centra en tales prácticas, ya que es más general sobre la estructuración de los profesionales del desarrollo y la infraestructura. Además, tenga en cuenta que las teorías sociales rara vez se confirman, sino que son corroboradas, confrontadas o evolucionadas por nuevos estudios [11-13]. Nuestro trabajo es único en el sentido de que (1) analiza una muestra diferente y (2) sigue un método de investigación diferente; además, (3) discutimos las estructuras descubiertas con más profundidad, y (4) destacamos las similitudes y diferencias entre las estructuras descubiertas por nosotros y Shahin et al. aportando así más robustez al conocimiento en el área. De esta manera, este artículo aborda las siguientes preguntas de investigación:

RQ1: ¿Qué estructuras organizativas adoptan las organizaciones productoras de software para estructurar las actividades de operaciones (implementación de aplicaciones, configuración de infraestructura y operación de servicios en tiempo de ejecución) entre los grupos de desarrollo e infraestructura?

RQ1.1: ¿Cuáles son las propiedades de cada una de estas estructuras organizativas?

RQ1.2: ¿Son algunas estructuras organizativas más propicias para la entrega continua que otras?

Para responder a estas preguntas, aplicamos la Grounded Theory [14], una metodología muy adecuada para generar teorías. El resultado principal de este enfoque de investigación es una taxonomía, que es nuestra teoría emergente. Una taxonomía es un sistema de clasificación que agrupa instancias similares para aumentar la eficiencia cognitiva de los usuarios, lo que les permite razonar sobre clases en lugar de instancias individuales [15]. Recopilamos datos preliminares en conversaciones de lluvia de ideas con siete especialistas, quienes nos ayudaron a comprender mejor la relevancia del problema y a dar forma a las preguntas que se harán en las entrevistas de seguimiento. Luego realizamos entrevistas semiestructuradas con 37 profesionales de TI. A partir del análisis de las entrevistas, descubrimos cuatro estructuras organizativas:

- **(i)** Departamentos tradicionales aislados, lo que dificulta la cooperación entre el desarrollo y las operaciones.
- **(ii)** DevOps clásico, centrado en la comunicación y la colaboración entre el desarrollo y las operaciones.
- **(iii)** Equipos multifuncionales, asumiendo la responsabilidad tanto del desarrollo de software como de la gestión de la infraestructura.

(iv) Equipos de plataforma, proporcionando servicios de infraestructura altamente automatizados para ayudar a los desarrolladores.

Para cada una de estas estructuras organizativas, identificamos propiedades básicas y suplementarias. Una organización clasificada como que adopta una estructura dada presentará la mayoría de las propiedades centrales asociadas con esa estructura. Las propiedades suplementarias, por el contrario, apoyan la explicación de patrones estructurales más detallados, y una organización puede (o no) exhibirlos. Este documento contribuye al área al presentar una taxonomía derivada sistemáticamente de las estructuras organizativas, basada en observaciones de campo recientes y empleando una metodología bien aceptada. En particular, nuestra taxonomía aporta los siguientes beneficios clave: *(i)* ayuda a los profesionales a diferenciar devOps clásicos de los equipos multifuncionales, que tradicionalmente se combinaban bajo el término DevOps [2,16], y (ii) destaca al equipo de la plataforma como una alternativa prometedora para las organizaciones. Además, nuestra taxonomía puede ayudar a los profesionales a discutir la situación actual de sus corporaciones, apoyando las decisiones sobre cambios estructurales. También apoya la comprensión del estado de una organización desconocida, lo que puede ayudar, por ejemplo, a los ingenieros en entrevistas de trabajo a evaluar la idoneidad de trabajar para una empresa determinada. Algunos de los hallazgos que discutimos en este documento se relacionan con el rendimiento de la entrega, un constructo compuesto por métricas cuantitativas, que explicamos en la Sección 2. Explicamos nuestro enfoque de investigación en la Sección 3, y presentamos nuestra taxonomía en detalle en la Sección 4. Después de esto, en la Sección 5 discutimos los comentarios de los entrevistados y nuestra evaluación. La Sección 6 discute el trabajo relacionado, mientras que la Sección 7 presenta las limitaciones de este trabajo. Por último, en la Sección 8 sacamos nuestras conclusiones y planes para la labor futura.

2. FONDO

El rendimiento de la entrega combina tres métricas: frecuencia de implementación, tiempo desde la confirmación hasta la producción y tiempo medio hasta la recuperación [17]. Se correlaciona con la capacidad organizativa de alcanzar tanto los objetivos comerciales (rentabilidad, productividad y cuota de mercado) como los objetivos no comerciales (eficacia, eficiencia y satisfacción del cliente) [18]. Utilizamos este constructo como una indicación de cuán exitosa ha sido una organización en la adopción de la entrega continua. Le pedimos a cada participante en nuestro estudio sobre cada una de estas métricas que definiera el desempeño de la entrega en el contexto del entrevistado. Sobre la base de una encuesta con 27.000 respuestas, Forsgren et al. [18] aplicaron el análisis de clústeres a estas métricas y descubrieron tres grupos: los de alto rendimiento se caracterizaron como aquellos con múltiples implementaciones por día, las confirmaciones que tardan menos de 1 h en llegar a la producción y los incidentes reparados en menos de 1 h. Los ejecutantes medianos que se implementaban una vez por semana a una vez por mes, tenían un tiempo desde el compromiso hasta la producción entre una semana y un mes, y tardaban menos de un día en reparar los incidentes. Los de bajo rendimiento presentaron las mismas características de los de rendimiento medio para la frecuencia de implementación y el tiempo desde la confirmación hasta la producción, pero tardan entre un día y una semana en reparar los incidentes. En nuestra investigación, estamos interesados en distinguir entre personas de alto y no alto rendimiento, no en identificar a los de rendimiento medio o bajo. Sin embargo, los grupos anteriores presentan un problema, porque existe una brecha en los valores utilizados

para identificar los clústeres de rendimiento alto y medio. Evitamos este problema al considerar a una organización como de alto rendimiento si (i) está dentro de los límites que limitan el grupo de personas de alto rendimiento definido anteriormente, o (ii) no viola más de un umbral de alto rendimiento por no más de un punto en la escala adoptada para la métrica. Las escalas para cada métrica son:

- **Frecuencia de despliegue:** múltiples despliegues por día; entre una vez por día y una vez por semana; entre una vez por semana y una vez por mes; entre una vez por mes y una vez cada seis meses; menos de una vez cada seis meses.
- **Tiempo desde la entrega hasta la producción:** menos de una hora; menos de un día; entre un día y una semana; entre una semana y un mes; entre un mes y seis meses; más de seis meses.
- **Tiempo medio de recuperación:** menos de una hora; menos de un día; entre un día y una semana; entre una semana y un mes; entre un mes y seis meses; más de seis meses.

3. DISEÑO DEL ESTUDIO

Esta sección presenta nuestro enfoque de investigación, incluido el proceso que utilizamos para recopilar y analizar datos.

3.1. TEORÍA FUNDAMENTADA

Nuestra investigación tiene como objetivo generar una teoría en forma de taxonomía para las estructuras organizativas en el contexto de la entrega continua. En términos generales, una teoría es un sistema de ideas para explicar un fenómeno [15]. Las taxonomías, por otro lado, son clasificaciones, es decir, colecciones de clases, en las que cada clase es una abstracción que describe un conjunto de propiedades compartidas por las instancias de la clase [15]. Si la taxonomía proporciona una explicación, puede considerarse una teoría para la comprensión: un sistema de ideas para dar sentido a lo que existe o está sucediendo en un dominio [15]. Al aplicar la "taxonomía" para denotar una formulación teórica, empleamos la "clasificación" en un sentido más amplio, incluidas las propuestas de clasificación sin formulación teórica. Grounded Theory (GT) es una metodología muy adecuada para generar taxonomías [15] y un enfoque de investigación ampliamente utilizado en ingeniería de software [19–22,16,23]. Una teoría fundamentada debe ajustarse a los datos, predecir, explicar, tener relevancia para el campo y ser modificable [14]. Su principal ventaja es fomentar la inmersión profunda en los datos, lo que puede proteger a los investigadores de instancias perdidas o de procesos de simplificación y racionalización excesiva [15]. GT también es adecuado para nuestros propósitos ya que, según Stol et al. [19], es adecuado para preguntas como ''¿qué está pasando aquí?''. En nuestro caso, queremos saber qué está pasando en las organizaciones productoras de software que están aprovechando la entrega continua. Dado que existen múltiples variantes gt, es importante indicar qué variante adoptamos. En este artículo, basamos nuestro enfoque en el libro seminal The

Discovery of Grounded Theory de Glaser y Strauss [14], que describe lo que se conoce como la "teoría clásica de Grounded" [19]. Si bien una teoría en sí misma debe surgir de los datos, Glaser y Strauss no rechazan las preguntas de investigación preestablecidas.2 El método comparativo constante es el método central para producir una teoría fundamentada. Se basa en el análisis riguroso de datos cualitativos, y se logra con la codificación, un proceso de condensación de datos originales en unas pocas palabras con relevancia conceptual, que dan lugar a conceptos teóricos. Glaser y Strauss no prescriben un formato de codificación preciso [14], lo que implica que el investigador anota conceptos y se adhiere a las siguientes reglas: *(i)* al anotar un concepto, comparar esta ocurrencia con las ocurrencias anteriores de los mismos conceptos o similares y *(ii)* mientras codifica, si surgen conflictos y reflexiones sobre nociones teóricas, escribir una nota sobre las ideas. Un memorándum es una nota no estructurada que refleja los pensamientos del investigador en un momento específico en el tiempo. Describimos con más detalle en la Sección 3.5 cómo aplicamos la codificación. Además del análisis riguroso de los datos cualitativos, GT también se basa en la sensibilidad teórica del investigador; es decir, su capacidad para tener una visión teórica de un área sustantiva. Nuestra sensibilidad teórica proviene de la experiencia directa en la industria de TI y nuestros trabajos anteriores sobre DevOps e ingeniería de software [24,25,23], especialmente una encuesta sobre la literatura de DevOps [26]. Esta sensibilidad teórica adquirida explica nuestra capacidad para plantear las preguntas de investigación de este trabajo. Este artículo también se basa en nuestro trabajo reciente, un resumen extendido que presenta brevemente las cuatro estructuras organizativas de nuestra taxonomía [27] y un breve documento que presenta la estructura del equipo de la plataforma solo [28]. En contraste, el documento actual describe en detalle las cuatro estructuras organizativas y sus propiedades. En GT, la recopilación y el análisis de datos se mezclan y se basan entre sí, por lo que la teoría emergente guía qué datos muestrear a continuación, considerando las brechas y preguntas sugeridas en el análisis anterior. Este proceso, llamado muestreo teórico, evita las nociones estadísticas habituales de los métodos de verificación, como la muestra significativa. En cambio, los investigadores deben establecer el propósito teórico de la muestra, definiendo múltiples grupos de comparación, maximizando la variación entre los grupos para identificar similitudes y minimizando la variación para determinar las diferencias. Abordamos el muestreo teórico principalmente mediante: *(i)* la valoración de la diversidad de personas y organizaciones en nuestra muestra, fortaleciendo la transferibilidad de nuestra teoría; y *(ii)* entrevistando a personas en contextos en los que podríamos explorar hipótesis débilmente respaldadas por la cadena de evidencia construida hasta ahora. Elaboramos más sobre las opciones de los participantes en la Sección 3.3. Idealmente, el investigador continúa el análisis hasta que se logra la saturación teórica, lo que significa que los nuevos datos ya no afectan significativamente la teoría. En este trabajo, alcanzar la saturación avanza nuestras publicaciones anteriores [28,27]. Nuestros criterios de saturación se describen en la Sección 3.6. Como una entidad en constante evolución, en lugar de un producto terminado, los nuevos datos siempre se pueden analizar para alterar o expandir una teoría fundamentada. En consecuencia, los profesionales podrían (y potencialmente lo harán) ajustar la teoría al aplicarla a sus escenarios concretos [14]. Por lo tanto, en este trabajo, presentamos una teoría emergente, en lugar de una totalmente validada, que explicamos más en la Sección 5.2. Aplicamos las técnicas de GT a los datos de las entrevistas con profesionales de TI. En las siguientes secciones, presentamos cómo elegimos nuestros temas y el diseño y análisis de estas entrevistas.

3.2. SESIONES DE LLUVIA DE IDEAS

Después de redactar nuestras preguntas de investigación, realizamos "sesiones de lluvia de ideas" con siete especialistas con experiencia en DevOps. Algunos de ellos han sido testigos de transformaciones de DevOps en grandes organizaciones, mientras que otros han dado forma activa a tales transformaciones en grandes y pequeñas empresas. El guión base para estas sesiones tuvo como objetivo obtener comentarios sobre nuestras preguntas de investigación y provocar la discusión de las preocupaciones planteadas por nuestra encuesta sobre la literatura de DevOps [26]. Las conversaciones fueron esenciales para que afináramos nuestras preguntas de investigación y nuestro enfoque de investigación. Estas sesiones también nos ayudaron a orientar mejor el guión de las siguientes entrevistas semiestructuradas hacia las preocupaciones aprendidas de estos expertos. Por lo tanto, los resultados concretos de esta fase fueron: las preguntas de investigación presentes en este texto y el guión de la entrevista (explicado en la Sección 3.4). No se aplicaron los procedimientos de análisis detallados en la Sección 3.5 para estas conversaciones preliminares, considerando que no estaban destinados a proporcionar respuestas a las preguntas de la investigación. Sin embargo, no descartamos las ideas teóricas proporcionadas por ellos. Por ejemplo, la noción de un equipo de plataforma comenzó a tomar forma en las sesiones de lluvia de ideas y, por lo tanto, influyó en el análisis posterior. Después de estas sesiones de lluvia de ideas, comenzamos las entrevistas semiestructuradas.

3.3. SELECCIÓN DE PARTICIPANTES

Enviamos alrededor de 90 invitaciones a entrevistas utilizando un enfoque de conveniencia: las primeras invitaciones fueron contactos cercanos en la red de nuestro grupo de investigación. También se estableció contacto con los participantes sugeridos por nuestros entrevistados y colegas. El único requisito era que el participante trabajara en un contexto industrial que haya adoptado la entrega continua o que esté implementando esfuerzos para lograrlo. Algunos participantes invitados no respondieron, y algunos demostraron interés en participar, pero no pudieron hacer tiempo para ello. Finalmente, entrevistamos a 37 profesionales de TI (≈ 41%). Siguiendo procedimientos éticos [29], todos los entrevistados y sus organizaciones son anonimizados en este documento. Grabamos las entrevistas para su posterior análisis, manteniendo los registros de audio bajo acceso restringido. Realizamos las entrevistas desde abril de 2019 hasta mayo de 2020. Nueve entrevistas se realizaron en persona, cinco de las cuales tuvieron lugar en la empresa del entrevistado, y 28 se realizaron en línea. Las sesiones duraron 50 minutos en promedio (mínimo de 24 y un máximo de 107 minutos). Empleamos varias estrategias para fomentar la diversidad y mejorar las posibilidades de comparación en nuestra muestra, según lo recomendado por las directrices gt [19]. Nuestro objetivo era incluir una amplia gama de perfiles de organizaciones y entrevistados. Por ejemplo, seleccionamos organizaciones de diferentes tamaños3: pequeñas (30%), medianas (32%) y grandes (38%); de diferentes tipos: privado (90%), gubernamental (5%) y otros (5%); y de diferentes sectores y países. Se incluyeron profesionales masculinos (73%) y femeninos (27%), y también se eligieron entrevistados con diferentes roles. La Tabla 1 describe a los participantes, presentando solo un perfil agregado de participantes para hacer frente a la anonimización [29,22]. La ubicación se refiere a la del equipo del entrevistado; tuvimos cuatro participantes trabajando de forma remota para equipos distribuidos globalmente. Al describir roles, el equipo de habilitadores indica un equipo técnico especializado que admite desarrolladores, pero sin poseer ningún servicio. Para las entrevistas con

consultores, el número de empleados se refiere al tamaño de las empresas que contrataron a los consultores (y no a los empleadores del consultor). Los entrevistados trabajaron en los siguientes dominios de negocio: IoT, finanzas, defensa, administración pública, justicia, bienes raíces, mapas, educación, Internet, big data, investigación, seguros, nube, juegos, comercio electrónico, telecomunicaciones, moda, relaciones internacionales, movilidad, automatización de oficinas (Tabla 1), consultoría de software, gestión de inventarios, automatización vehicular, gestión de equipos y soporte al desarrollo de software. Cinco de las empresas entrevistadas se consideran actualmente startups unicornio, y dos de ellas son gigantes tecnológicos. También seleccionamos a los participantes con fines teóricos en mente, aplicando así el muestreo teórico. Entrevistamos a participantes que trabajan en escenarios donde es particularmente difícil lograr una entrega continua (por ejemplo, IoT, juegos o sistemas de defensa), con el objetivo de comprender los límites y los eventuales casos de esquina. Después de la vigésima entrevista, buscamos más activamente personas: en un equipo multifuncional, en (o interactuando con) un equipo de plataforma, sin (o pocas) pruebas automatizadas, con sistemas monolíticos y etiquetados como "ingenieros de pila completa". Adoptamos estos criterios debido a hipótesis no bien respaldadas por nuestra cadena de evidencia en ese momento. Para apoyar nuestros hallazgos, incluimos extractos de estas conversaciones en nuestra cadena de evidencia (en el material complementario adjunto) y en este artículo. Los extractos están formateados en cursiva y entre comillas. Extractos y otros relatos se refieren a entrevistas utilizando tokens en el formato ''#IN''. Por lo tanto, "#I2" se refiere a la segunda entrevista, entrevistado u organización del entrevistado. Las sesiones de lluvia de ideas se indican como "#BN". Tales extractos y citas están destinados a hacer que los lectores "sientan que también estaban en el campo", una recomendación de GT [14].

3.4. REALIZACIÓN DE LAS ENTREVISTAS

Dado que nuestro objetivo es descubrir las estructuras organizativas existentes, y no verificar un conjunto preconcebido de estructuras en el campo, no sería adecuado usar solo preguntas cerradas; en su lugar, realizamos entrevistas semiestructuradas. Las entrevistas semiestructuradas mezclan preguntas cerradas y abiertas, a menudo acompañadas de preguntas de "por qué" y "cómo"; la entrevista puede desviarse de las preguntas planificadas, lo que permite la discusión de temas imprevistos [30], lo que se ajusta al propósito de la generación de teoría. Con las entrevistas semiestructuradas, también podríamos centrarnos en diferentes temas en diferentes conversaciones, según la relevancia de cada tema para cada contexto. Antes de comenzar las entrevistas, construimos un protocolo de entrevistas4 para guiar el proceso basado en nuestra experiencia previa con entrevistas [25], en otros trabajos relevantes [21,30] y en las pautas ofrecidas por un sitio web para periodistas, llamado ijnet.5 El protocolo de entrevista contiene las preguntas que impulsaron las entrevistas, que principalmente se derivaron de las sesiones de lluvia de ideas y nuestra encuesta de la literatura de DevOps [26], y, por lo tanto, también se basan en datos.6 Los temas abordados por las preguntas de la entrevista incluyen: (1) la empresa y el rol del entrevistado; (2) responsabilidad por la implementación, la construcción de nuevos entornos, los requisitos no funcionales, la configuración y el seguimiento de la supervisión y el manejo de incidentes, especialmente fuera del horario de atención; (3) rendimiento de la entrega (utilizando las métricas y escalas definidas en la Sección 2); (4) mejoras futuras en la organización; (5) efectividad de la comunicación entre equipos; (6) alineación entre equipos para el éxito de los proyectos; (7) descripción del equipo de DevOps o rol de DevOps, si existe; y (8) la política para

compartir personas especializadas (por ejemplo, expertos en seguridad y bases de datos) entre diferentes equipos. El protocolo de entrevista no es un documento estático. A medida que realizamos entrevistas, cambiamos la forma en que hacíamos algunas preguntas, nos enfocamos más en algunas preguntas y menos en otras, y creamos nuevas preguntas para explorar hipótesis crecientes. Aportamos algunas indicaciones sobre la evolución de las preguntas en el propio protocolo de entrevistas.

3.5. ANÁLISIS DE LAS ENTREVISTAS

Seguimos los principios básicos de la Teoría Fundamentada del método comparativo constante y la codificación, que están destinados a disciplinar la generación creativa de la teoría. Durante este proceso, creamos dos artefactos para cada entrevista: las **transcripciones** y los **códigos**. También creamos dos artefactos globales: la **hoja de comparación** y el **marco conceptual**. Finalmente, al analizar, comparar y usar todos estos artefactos, elaboramos nuestra **taxonomía**, que es la teoría en sí.

Transcripciones. Escuchamos cada grabación de audio y la transcribimos. No transcribimos la entrevista completa. En su lugar, resumimos las partes relevantes, excluyendo detalles menores y ruido sin sentido [31]. Por ejemplo, transcribimos la siguiente parte de una conversación: "Si rompes el SLA, hay consecuencias. Hay que mejorar las cosas; no puede volver al desarrollo de características hasta que el SLA se haya recuperado. Cualquier problema en el servicio final: el desarrollador está paginado. Si está relacionado con la infraestructura, los desarrolladores llaman al equipo de infraestructura. Y resolvemos juntos. Tratamos de ayudar de todos modos, porque al final del día, si los usuarios no pueden usar el sistema, todos sufrimos".

Códigos. Después de transcribir una entrevista, derivamos la codificación condensando las transcripciones de los entrevistadores en unas pocas palabras (la esencia de la entrevista en relación con nuestras preguntas de investigación). Cada entrevista tiene su lista de codificación, que representa la realidad particular de ese entrevistado. El fragmento de transcripción anterior, por ejemplo, condujo a la siguiente codificación:

> Los desarrolladores → poseen la disponibilidad de sus servicios
>
> Desarrollo de características de bloques de → SLA rotos
>
> SLA roto → desarrolladores de páginas Broken
>
> SLA → si es necesario, llame a infra

El material complementario "cadena de evidencia" presenta más ejemplos de la codificación que realizamos.

La hoja de comparación. Para apoyar la comparación constante de diferentes entrevistas y codificaciones, resumimos las principales características de cada entrevista en una hoja de cálculo. Llenamos las celdas con declaraciones concisas, con filas que representan entrevistas y columnas que incluyen las siguientes características: número de entrevista, estructura organizativa, propiedades suplementarias, rendimiento de entrega, observación, entrega continua, microservicios, nube, otros equipos, requisitos no funcionales (NFR), monitoreo / guardia, alineación, comunicación, cuello de botella, conflictos de responsabilidad, base de datos, seguridad, uso compartido de especialistas y equipo / rol de DevOps.

Marco conceptual. De la comparación constante de códigos de diferentes entrevistas surgieron

conceptos teóricos. Estos conceptos teóricos (Fig. 1) y sus relaciones forman el marco conceptual unificado, que constituye el entendimiento compartido entre los autores sobre los datos analizados [32]. Nuestro marco conceptual no es una representación de nuestra teoría, sino más bien un artefacto intermedio utilizado para consolidar, en un solo lugar, los conceptos producidos por el proceso de codificación, trabajando como la fuente de conceptos para la conformación de nuestra teoría. Mantuvimos una representación visual de nuestro marco conceptual a medida que la investigación evolucionaba. Proporcionamos todas sus versiones en los materiales complementarios. La Fig. 1 proporciona como ejemplo un fragmento de nuestro marco conceptual. En esa figura, los rectángulos representan conceptos abstraídos de los datos, mientras que los cuadros redondeados representan las propiedades de estos conceptos (es decir, un atributo o una característica de un concepto). A medida que evolucionamos el marco conceptual y llenamos nuestra hoja de comparación, desarrollamos nuestra teoría clasificando cada entrevista por su estructura organizativa y sus propiedades complementarias. El proceso de clasificación se basó en los conceptos proporcionados por el marco y en el análisis de similitudes y diferencias entre las entrevistas, resumidas en la hoja de comparación. A medida que evolucionamos nuestra comprensión con nuevas entrevistas, revisamos la clasificación de entrevistas anteriores para refinar nuestra teoría. Por ejemplo, después de la aparición de una propiedad suplementaria, comprobamos si podíamos clasificar las entrevistas anteriores con la nueva propiedad. Después de algunas entrevistas, un autor desarrolló la primera versión de las listas de codificación, la hoja de comparación, el marco conceptual y la taxonomía. Sobre la base de las transcripciones, otros dos autores revisaron a fondo estos artefactos, lo que desencadenó discusiones que afectaron su evolución. Al tomar una decisión que podría afectar nuestra taxonomía, involucramos a un cuarto autor. Un ejemplo de cómo evolucionamos y mejoramos nuestra taxonomía es cómo desarrollamos las propiedades suplementarias después de veinte entrevistas. Se llevaron a cabo rondas adicionales de análisis, discusiones y elaboración de teorías hasta la presentación de este artículo. Pasamos por este proceso de revisión interna para reducir el sesgo de un solo investigador que realiza análisis con ideas preconcebidas.

3.6. SATURACIÓN TEÓRICA

Consideramos el tamaño de nuestro marco conceptual como un indicador de cuánto hemos aprendido hasta ahora sobre nuestro tema de investigación. Definimos el tamaño del marco conceptual como el número de elementos en el diagrama que lo representa, que cuenta el número de conceptos, propiedades conceptuales y enlaces. Considere la Fig. 2-(a): el eje x representa el número de entrevistas, mientras que el eje y representa el número de elementos en nuestro marco conceptual. En esta figura, vemos que las últimas 15 entrevistas (40% de ellas) aumentaron el tamaño de nuestro marco conceptual en solo un 9%. Además, las tres últimas entrevistas no aumentaron el tamaño del marco conceptual en ningún grado. Esto sugiere que más entrevistas proporcionarían solo una ganancia insignificante para nuestro marco. Además de medir el tamaño del marco conceptual, un artefacto intermediario, también medimos el crecimiento de la taxonomía en sí, nuestro producto final. La Fig. 2-(b) muestra que en las primeras cinco entrevistas descubrimos nuestras cuatro estructuras organizativas. También muestra que para la entrevista 22, ya teníamos todas menos una de las propiedades suplementarias. La última propiedad suplementaria descubierta es un caso excepcional, y se aplicó solo para una entrevista. Por lo tanto, el crecimiento decreciente de la taxonomía sugiere que las

últimas entrevistas contribuyeron mucho menos a dar forma a nuestra teoría. Al unir estas dos observaciones de la Fig. 2, afirmamos haber alcanzado suficiente saturación para los propósitos de nuestra teoría.

3.7. RETROALIMENTACIÓN

GT tiene como objetivo formular una teoría que tenga relevancia para los profesionales, por lo que es crucial investigar también si los hallazgos tienen sentido para ellos [15]. Además, los profesionales pueden ayudar a identificar errores de taxonomía, como errores de inclusión y exclusión [15]. Por lo tanto, recopilamos comentarios sobre nuestra taxonomía de los participantes del estudio, utilizando una encuesta en línea.7 Los comentarios recibidos se presentan en la Sección 5.1.

4. LA TAXONOMÍA DE LAS ESTRUCTURAS ORGANIZATIVAS

En esta sección, respondemos a nuestras preguntas de investigación presentando nuestra taxonomía de la organización de los equipos de desarrollo e infraestructura, incluidas las estructuras organizativas que identificamos, junto con sus propiedades centrales y complementarias. Se espera que las propiedades principales se encuentren en corporaciones con una estructura determinada. Cuando corresponda, utilizamos las propiedades principales para analizar el rendimiento de la entrega. Las propiedades suplementarias refinan la explicación de una estructura, pero su asociación con las organizaciones no es obligatoria. Para cada entrevista, clasificamos la estructura organizativa observada. Dado que los patrones de diferenciación e integración entre el desarrollo y la infraestructura pueden variar para cada unidad desplegable [33], no es posible asignar una estructura única a una organización. Por lo tanto, la clasificación se aplica de acuerdo con el contexto del entrevistado. Además, observamos en algunos casos un proceso de adopción gradual de nuevos patrones estructurales mientras se abdica de los antiguos; clasificamos esto como transiciones de una estructura organizativa a otra. La Fig. 3 presenta las estructuras organizativas descubiertas, los elementos primarios de nuestra taxonomía, junto con las propiedades suplementarias, que califican los elementos señalados por las flechas. Los círculos agrupan propiedades suplementarias que se pueden aplicar igualmente para un elemento dado. La Tabla 2 muestra la clasificación (estructura organizacional y propiedades suplementarias) aplicada para cada entrevista, junto con el desempeño de entrega alcanzado, y el número de empleados en la organización correspondiente. Nuestra cadena completa de evidencia, agregada como material complementario, indica cuánto de nuestros hallazgos están respaldados por los datos que recopilamos. La cadena de evidencia vincula cada estructura organizativa y propiedad suplementaria con la codificación de apoyo, los memorandos y los extractos. Tal vinculación es fundamental para la credibilidad de los resultados de la investigación cualitativa [34,15,35]. Ahora presentamos cada una de las estructuras organizativas y sus propiedades centrales y complementarias. (Figura 2,3)

4.1. DEPARTAMENTOS AISLADOS

Con departamentos aislados, los desarrolladores y el personal de infraestructura están segregados entre sí, con poca comunicación directa entre ellos. Las fricciones se producen entre los silos, ya que los desarrolladores quieren entregar tanto como sea posible, mientras que las operaciones apuntan a la estabilidad y las entregas de bloques. El movimiento DevOps nació en 2008 [36] para

manejar tales problemas. Encontramos siete organizaciones que se adhieren a esta estructura, y otras seis que hacen la transición fuera de esta estructura. Si bien no surgieron propiedades suplementarias para esta estructura, encontramos siete propiedades centrales para corporaciones con departamentos aislados:

→ Los desarrolladores y operadores tienen **roles bien definidos y diferenciados**; como lo afirma #I20: "el muro fue muy claro: después de comprometernos, nuestro trabajo [como desarrolladores] se hizo". Por lo tanto, no hay conflictos con respecto a las atribuciones. Los roles y canalizaciones bien definidos pueden disminuir la necesidad de colaboración directa interdepartamental (#I10).

→ **Cada departamento se guía por sus propios intereses**, buscando la optimización local en lugar de la optimización global, un patrón antiguo y problemático [37]. El participante #I26 nos dijo''hay una gran guerra allí. . . los grupos de seguridad, gobernanza y auditoría aún deben estar convencidos de que la herramienta [Docker/Kubernetes] es buena''.

→ **Los desarrolladores tienen un conocimiento mínimo de lo que sucede en la producción** (#I26). Por lo tanto, el monitoreo y el manejo de incidentes son realizados principalmente por el equipo de infraestructura (#I5).

→ **Los desarrolladores a menudo descuidan los requisitos no funcionales (NFR),** especialmente la seguridad (#I5). En #I30, los conflictos entre los desarrolladores y el grupo de seguridad surgen del desacuerdo sobre las decisiones técnicas. En otros casos, los desarrolladores tienen poco contacto con el grupo de seguridad (#I26).

→ **las iniciativas de DevOps limitadas**, centradas en la adopción de herramientas, no mejoran la comunicación y la colaboración entre los equipos (#I30) ni difunden el conocimiento sobre las pruebas automatizadas (#I5, #I15). En #I30, un "equipo de DevOps" que mantiene la canalización de implementación se comporta como otro silo, a veces con cuellos de botella en la entrega [38]. (Tabla 2)
→ **Es menos probable que las organizaciones logren un alto rendimiento de entrega,** ya que los desarrolladores necesitan aprobación burocrática para implementar aplicaciones y evolucionar el esquema de la base de datos (#I5, #I30). El cuadro 3 muestra que sólo dos de las 13 organizaciones aisladas presentaban un alto rendimiento en la ejecución, y estas dos ya estaban haciendo la transición a otras estructuras. Sin embargo, observamos casos en los que el bajo rendimiento de entrega no fue un problema, como experimentos de investigación de corta duración (#I13) y lanzamientos de nuevas fases de un juego que no requieren cambios de código (#I10). Las directivas de aislamiento de red también pueden dificultar la implementación frecuente (#B1, #I7).

→ Observamos una **falta de automatización de pruebas adecuada** en muchas organizaciones (#I5, #I15, #I23 #I26). En #I26, los desarrolladores automatizan solo las pruebas unitarias. La organización #I15 estaba dejando la automatización de pruebas solo para las personas de control de calidad, lo que no es adecuado para TDD o pruebas unitarias. Aunque las organizaciones aisladas no son las únicas que carecen de automatización de pruebas (#I3, #I32, #I35), en esta estructura los desarrolladores pueden incluso ignorar su valor (#I5, #I23, #I37). Notamos que algunos de los escenarios observados eran más desafiantes para la automatización de pruebas, como los juegos.

4.2. DEVOPS CLÁSICOS

La estructura clásica de DevOps se centra en la colaboración entre los desarrolladores y el equipo de infraestructura. No elimina todos los conflictos, pero promueve un mejor ambiente para lidiar con ellos (#I34). Llamamos a esta estructura "DevOps clásico" porque entendemos que una cultura colaborativa es la principal preocupación de DevOps [23,26,39]. Clasificamos diez organizaciones en esta estructura. También observamos tres organizaciones en transición a esta estructura y tres en transición fuera de esta estructura. Las ocho propiedades principales observadas para las organizaciones que adoptan DevOps clásicos son las siguientes: (Tabla 3)
→ Observamos que, en entornos clásicos de DevOps, muchas prácticas fomentan una **cultura de colaboración**. Vimos el intercambio de la administración de bases de datos: el personal de infraestructura crea y ajusta la base de datos, mientras que los desarrolladores escriben consultas y administran el esquema de la base de datos (#I17). Escuchamos sobre la comunicación abierta entre los desarrolladores y el equipo de infraestructura (#I2, #I6, #I17, #I22, #I31, #I36). El participante #I2 destacó que: "Los equipos de desarrollo e infraestructura participan en el mismo chat; incluso parece que todos son parte del mismo equipo". Los desarrolladores también apoyan el producto en su producción inicial (#I31).→ Roles permanecen bien definidos, y a pesar de la colaboración en algunas actividades, generalmente no hay
conflictos sobre quién es responsable de cada tarea.
→ Los desarrolladores se sienten aliviados cuando pueden confiar en el equipo de infraestructura (#I17). El participante #I31 afirmó que su trabajo anterior en un equipo multifuncional tenía un entorno mucho más estresante que su posición actual en un equipo de desarrollo en un entorno clásico de DevOps. Por otro lado, el **estrés puede persistir en niveles altos para el equipo de infraestructura** (#I34), especialmente ''si la aplicación está mal diseñada y tiene bajo rendimiento'' (#I36).→ En esta estructura, el éxito del proyecto depende de la
alineación de diferentes departamentos, lo que no es trivial de lograr. En #B3, diferentes equipos entendieron los objetivos de la organización y las consecuencias de no resolver problemas, como calcular erróneamente cantidades del orden de millones de dólares. Además, #I7 descrito que la alineación surge cuando los empleados se centran en la resolución de problemas en lugar de las atribuciones de roles.

→ **los equipos de desarrollo e infraestructura comparten responsabilidades de NFR** (# 17). Por ejemplo, en #I2, ambos estaban muy preocupados por la baja latencia, un requisito principal para su aplicación.

→ Por lo general, **el personal de infraestructura es la primera línea de seguimiento de monitoreo y manejo de** incidentes (#I2, #I11, #I29, #I31 #I36). Sin embargo, si es necesario, los desarrolladores son convocados y colaboran (#I17, #I34). En #I34, las alertas de supervisión se dirigen al equipo de infraestructura, pero se copian a los desarrolladores. Sin embargo, en algunos casos, los desarrolladores nunca trabajan fuera de horario (#I2, #I22).

→ Humble espera que una cultura de colaboración entre los desarrolladores y el personal de infraestructura **prescinda de un "equipo de DevOps"** [38]. Entendemos que esta crítica se aplica a los equipos de DevOps con miembros dedicados, como vimos en #I30, ya que se comportan como nuevos silos. Sin embargo, encontramos en #I36 un equipo de DevOps bien administrado que trabaja como un comité para decisiones estratégicas, un foro para el liderazgo de diferentes

departamentos. También encontramos grupos de DevOps que trabajan como gremios (#I4, #I8), apoyando el intercambio de conocimientos entre diferentes departamentos [40].

→ la colaboración y la automatización de la entrega, valores críticos del movimiento DevOps, **no son suficientes para lograr un alto rendimiento de entrega**. De las 10 organizaciones clásicas de DevOps que no hacen la transición desde o hacia otras estructuras, solo tres presentaron un alto rendimiento de entrega (Tabla 3). Una posible razón es la falta de automatización adecuada de las pruebas (#I22, #I36) [41]. Otra limitación para el rendimiento de la entrega es la adopción de ventanas de lanzamiento (#I11, #I31, #I14, #I36), que buscan mitigar el riesgo de implementación al restringir la entrega de software a intervalos de tiempo periódicos. Las ventanas de lanzamiento se adoptan teniendo en cuenta el número masivo de usuarios (#I31) o la criticidad financiera del sistema (#I36). Las ventanas de lanzamiento también pueden ser el resultado de arquitecturas frágiles (#I37) o el estilo arquitectónico monolítico (#I11), ya que cualquier implementación tiene un mayor riesgo de afectar a todo el sistema.
Propiedades suplementariasPara
las organizaciones clásicas de DevOps, encontramos una propiedad suplementaria que describimos a continuación.
Infra como colaborador de desarrollo. El personal de infraestructura contribuye al código de la aplicación para optimizar el rendimiento, la fiabilidad, la estabilidad y la disponibilidad del sistema. Aunque esta aptitud requiere habilidades avanzadas de codificación de profesionales de infraestructura, es una estrategia adecuada para mantener sistemas a gran escala, como los que posee #I31.

4.3. *EQUIPOS MULTIFUNCIONALES*

En nuestro contexto, un equipo multifuncional asume la responsabilidad tanto del desarrollo de software como de la gestión de la infraestructura. Esta estructura se alinea con el lema de Amazon "Lo construiste, lo ejecutas" [42] y con los "escuadrones autónomos" en Spotify [40]. Esto le da más libertad al equipo, junto con una gran responsabilidad. Como describió el entrevistado #I1: "Es como si cada equipo fuera una miniempresa diferente, que tiene la libertad de administrar su propio presupuesto e infraestructura". Encontramos siete organizaciones con esta estructura, dos organizaciones en transición a esta estructura y una en transición fuera de ella. Las cuatro propiedades principales que se encuentran para los equipos multifuncionales son las siguientes:→

la independencia entre los equipos puede conducir a una desalineación. La falta de comunicación y estandarización entre los equipos multifuncionales dentro de una sola organización puede conducir a esfuerzos duplicados (#I28). Sin embargo, esto no siempre es un problema (#I1).→
Es difícil asegurarse de que un equipo tenga todas las habilidades necesarias. Por ejemplo, entrevistamos a dos equipos multifuncionales sin experiencia en infraestructura (#I3, #I32). El participante #I27 reconoce que "hay una falta de conocimiento" sobre la infraestructura, la automatización de la implementación y el monitoreo. Una posible razón para tal adversidad es que, como nos enseñó #I29, es difícil contratar especialistas en infraestructura y desarrolladores senior.→ Esperábamos que los equipos multifuncionales proporcionaran demasiado tiempo ocioso para los especialistas, en lugar de grupos centralizados de especialización. Sin embargo, no encontramos
evidencia de ociosidad para los especialistas. Desde #I16, escuchamos todo lo contrario: los

especialistas en infraestructura estaban demasiado ocupados para ser compartidos con otros equipos. Tener a los especialistas en infraestructura codificando características en su tiempo libre evita tal ociosidad (#I35).

→ **La mayoría de los equipos multifuncionales que entrevistamos estaban en organizaciones pequeñas** (Tabla 4), probablemente porque no tiene sentido crear múltiples equipos en organizaciones demasiado pequeñas.

Propiedades

complementariasDesfilicibistas de infraestructura dedicados. El equipo cuenta con personal especializado dedicado a tareas de infraestructura. En #I1, un empleado se especializa en infraestructura física y otro es "DevOps", que se encarga de la canalización de implementación y el monitoreo. En esta circunstancia, los especialistas en infraestructura se convierten en la primera línea para abordar incidentes y monitorear (#I28, #I35).

Desarrolladores con experiencia en infraestructura. El equipo cuenta con desarrolladores con conocimientos en gestión de infraestructura; estos profesionales también se llaman ingenieros de pila completa o incluso ingenieros de DevOps (#I25). El participante #I25 es un ingeniero de pila completa y afirmó *"conocer todas las* tecnologías involucradas (Tabla 4*): front-end, back-end e infraestructura; por lo tanto, soy la persona capaz de vincularlas todas y combatir incendios cuando sea necesario".* El participante #I29, un consultor, es escéptico con respecto a los ingenieros de pila completa y declaró que "estas personas no están a la altura de la tarea". Se quejó de que los desarrolladores generalmente desconocen cómo ajustar la aplicación, como configurar las conexiones de la base de datos.

Sin antecedentes infra. Los equipos de producto gestionan la infraestructura sin la experiencia correspondiente. Vimos este patrón en dos lugares. Una era una empresa muy pequeña y acababa de lanzar su aplicación, teniendo solo unos pocos usuarios (#I32) y sin estar seguro de contratar personas especializadas pronto. El entrevistado #I3 entiende que el trabajo de operaciones (por ejemplo, detectar errores durante las actualizaciones de firmware en dispositivos IoT e iniciar Máquinas virtuales de Amazon para nuevos clientes) es demasiado pequeño para los ingenieros de software, lo que lleva demasiado tiempo costoso. Así que la organización estaba planeando la creación de un sector de operaciones compuesto por una fuerza laboral más barata. El entrevistado #I19 argumentó que tal acuerdo no podía sostener el crecimiento de su empresa en el pasado.

4.4. EQUIPOS DE PLATAFORMA

Los equipos de plataforma son equipos de infraestructura que proporcionan servicios de infraestructura altamente autoacoplados que los desarrolladores pueden autoadministrar para la implementación de aplicaciones. El equipo de infraestructura ya no es un "equipo de soporte"; se comporta como un equipo de producto, con la ''plataforma'' como su producto y los desarrolladores como clientes internos. En este contexto, los especialistas en infraestructura necesitan habilidades de codificación; los equipos de productos tienen que operar sus servicios comerciales; y la plataforma maneja gran parte de las preocupaciones no funcionales. Encontramos cuatro organizaciones que adoptan plenamente este modelo y otras cuatro en el proceso de adoptarlo. También observamos el patrón del equipo de la plataforma en tres de las sesiones de lluvia de ideas. Las propiedades principales de los equipos de plataforma son las siguientes:

→ **Los equipos de productos son totalmente responsables de los requisitos no funcionales de sus servicios.** Se convierten en los primeros llamados cuando hay un incidente, que se escala a la gente de infraestructura solo si el problema se relaciona con un servicio de infraestructura (#I8, #I9, #I12, #I33).→ Aunque el equipo de producto se vuelve totalmente responsable de los NFR de sus servicios, no es una carga significativa que los desarrolladores intenten rechazar (#I33).
La plataforma en sí maneja muchas preocupaciones de NFR, como el equilibrio de carga, el escalado automático, la limitación y las comunicaciones de alta velocidad entre centros de datos (#I4, #I8, #I16 #I33). Como nos dijo #I33 participantes, "no necesitas preocuparte por cómo funcionan las cosas, simplemente funcionan". Además, observamos que las personas de infraestructura apoyan voluntariamente a los desarrolladores por el bien de la disponibilidad, el rendimiento y la seguridad de los servicios (#I9, #I14).
→ los equipos de producto se desacoplan de los miembros del equipo de la plataforma. Por lo general, la comunicación entre estos equipos ocurre cuando los desarrolladores y las personas de infraestructura se reúnen para resolver incidentes (#I8, #I9); cuando las personas de infraestructura proporcionan consultoría para que los desarrolladores dominen las preocupaciones no funcionales (#I9); o cuando los desarrolladores exigen nuevas capacidades de la plataforma (#I8, #I12). De esta manera, el desacoplamiento entre la plataforma y los equipos de producto no implica la ausencia de colaboración entre estos grupos.

→ **El equipo de infraestructura ya no es solicitado para tareas** operativas. Las tareas operativas son automatizadas por la plataforma. Por lo tanto, no se puede simplemente llamar a los miembros del equipo de la plataforma "operadores", ya que también diseñan la solución de infraestructura. Señalamos que, en otras industrias, ''operador'' es un título atribuido a los trabajadores de baja categoría.

→ La plataforma **evita la necesidad de que los equipos de productos cuenten con especialistas en infraestructura**. Los #I33 expresaron su deseo de comprender mejor lo que sucede "bajo el capó" de la plataforma, lo que indica qué tan bien la plataforma alivia al equipo de dominar las preocupaciones de infraestructura. Por otro lado, dado que los desarrolladores son responsables de la implementación, deben tener algunos conocimientos básicos sobre la infraestructura y la plataforma en sí.

→ **La plataforma puede no ser suficiente para hacer frente a requisitos particulares**. El participante #I16 declaró que "si mucha gente hace una funcionalidad similar, con el tiempo generalmente se integra a la plataforma. . . pero cada equipo tendrá algo muy especializado...'' para explicar la presencia del personal de infraestructura dentro del equipo, incluso con el uso de una plataforma, teniendo en cuenta la gran cantidad de máquinas virtuales que se administrarán.

→ Si la organización desarrolla una nueva plataforma para hacer frente a sus especificidades**, requerirá habilidades de desarrollo del equipo de infraestructura**. Sin embargo, incluso sin desarrollar una nueva plataforma, el equipo de infraestructura debe tener una "mentalidad de desarrollo" para producir scripts y utilizar la infraestructura como código [43] para automatizar la ruta de entrega (#I14). Una estrategia que observamos para satisfacer esta necesidad fue contratar desarrolladores anteriores para el equipo de infraestructura (#I14).→ Las cuatro organizaciones que han adoptado completamente la estructura del equipo de la **plataforma son de alto rendimiento**, mientras que ninguna otra estructura proporcionó tal nivel

de éxito (Tabla 3). Una explicación para tal relación es que esta estructura desacopla los equipos de infraestructura y producto, lo que evita que el equipo de infraestructura obstruya la ruta de entrega. Como afirma #I20: "Ahora los desarrolladores tienen autonomía para pasar de cero a producción sin tener que esperar a nadie". Esta estructura también contribuye a la confiabilidad del servicio al permitir que los equipos de productos manejen requisitos e incidentes no funcionales.

→ En la Tabla 4, entre las organizaciones entrevistadas con equipos de plataforma, ninguna tenía menos de 200 empleados. Dado que el ensamblaje de un equipo de plataforma requiere personal dedicado con conocimientos especializados, tiene sentido que dicha estructura no sea **adecuada para pequeñas empresas**.

Propiedades suplementarias

La descripción de una plataforma se puede refinar aplicando una de las siguientes propiedades suplementarias:

Fachada en la nube. En última instancia, la plataforma implementa aplicaciones en nubes públicas, como Amazon WS, Google Cloud o Azure. Aunque estas nubes permiten una implementación más fácil en comparación con la administración de servidores físicos, todavía ofrecen docenas de servicios y una multitud de configuraciones. La plataforma interna estandariza el uso de los proveedores de nube pública dentro de la organización, por lo que los desarrolladores no necesitan comprender muchos detalles sobre la nube (#I4, #I14, #I33); por lo tanto, «mejorar la usabilidad de la infraestructura [en la nube]» (#I16).

Plataforma privada personalizada. La plataforma está construida sobre servidores físicos internos (#B1, #I1, #I8, #I12, #I20), ocultando complejidades de infraestructura a los desarrolladores, como el uso e incluso la existencia de Kubernetes, una plataforma de código abierto utilizada para administrar el ciclo de vida de los contenedores Docker. (Fig. 4)

Plataforma interna de código abierto. La plataforma es un software de código abierto implementado en las instalaciones. Organización #I20 utiliza Rancher,8 una interfaz gráfica para que los desarrolladores interactúen con Kubernetes.

4.5. PROPIEDADES COMPLEMENTARIAS COMPARTIDAS

Esta sección presenta las propiedades suplementarias encontradas que no están vinculadas a una sola estructura organizativa. Estas propiedades son relevantes ya que se comparten entre múltiples estructuras organizativas, como se muestra en la Fig. 3.

Equipo habilitador. Un equipo habilitador proporciona consultoría y herramientas para los equipos de productos, pero no posee ningún servicio. La consultoría puede ser sobre rendimiento (#I18) o seguridad (#I9, #I16, #I31), por ejemplo. Las herramientas proporcionadas por los equipos facilitadores incluyen la canalización de implementación (#I4, #I30), mecanismos de alta disponibilidad (#I11), herramientas de monitoreo (#I12) y herramientas de seguridad (#I17). Los encontramos en todas las estructuras organizativas. Aprendimos el término "equipo facilitador" durante nuestra entrevista con #I11.

Con una plataforma. La organización posee una plataforma que puede proporcionar automatización de la implementación, pero no sigue los patrones de interacción humana y colaboración descritos por las propiedades principales de los equipos de la plataforma. El participante #I25 desarrolló un ''*IaaC autónomo para la integración e implementación con Google*

Cloud'', que proporciona las capacidades de una plataforma a otros desarrolladores del equipo. Sin embargo, dado que en este contexto hay un solo equipo multifuncional, no se puede llamar un "equipo de plataforma". Clasificamos la organización #I30 como una estructura aislada, incluso con un equipo de plataforma, ya que los desarrolladores y el equipo de la plataforma tienen una relación conflictiva. Las propiedades suplementarias de los equipos de plataforma también se pueden aplicar a organizaciones con una plataforma.

4.6. TRANSICIÓN

Las organizaciones no son estáticas. Identificamos nueve de ellos en transición de una estructura a otra. Teniendo en cuenta los flujos de transición en la Fig. 4, percibimos que *(i)* ninguna organización está haciendo la transición a departamentos aislados, *(ii)* la mayoría de las transiciones son de departamentos aislados, y *(iii)* ninguna organización está haciendo la transición fuera del equipo de la plataforma. Estas observaciones concuerdan con nuestras consideraciones teóricas sobre los problemas de las estructuras en silos y las promesas de los equipos de plataforma. No obstante, la transición de las estructuras organizativas es un esfuerzo difícil, como lo confirman algunos entrevistados. Aunque su organización hizo un excelente trabajo de transición a una estructura de plataforma y logró un alto rendimiento de entrega, el entrevistado #I20 afirma que el *"viejo mundo"* todavía coexiste con el *"nuevo".* De la misma manera, como informan Nybom et al. [6], hay algunos conflictos de responsabilidad y *"fuerzas disidentes"*: a algunos miembros del personal de operaciones no les gustan los desarrolladores con poderes administrativos, mientras que algunos desarrolladores no quieren tales poderes. El entrevistado declaró que *"todavía no están todos juntos".* Del mismo modo, el entrevistado # 21 declaró que *"Hay dos mundos ... uno nació en la nube, y es bueno que influya en el sistema heredado para que sea más robusto. Hay muchos mundos que deseamos unir. Sin embargo, necesitamos reescribir incluso la cultura; debemos restablecer todo''*. Estos ejemplos también muestran cómo la cultura es un factor crucial para el cambio.

4.7. RESUMEN

Cerramos la descripción de nuestra taxonomía destacando las diferencias clave entre sus estructuras organizativas. La Tabla 5 resume, para cada estructura, *(i)* la diferenciación entre grupos de desarrollo e infraestructura con respecto a las actividades de operaciones (implementación, configuración de infraestructura y operación de servicio en tiempo de ejecución); y *(ii)* cómo interactúan estos grupos (integración).

5. DISCUSIÓN

En esta sección, discutimos las sesiones de retroalimentación y nuestra teoría emergente a la luz de la retroalimentación recibida.

5.1. RETROALIMENTACIÓN

Enviamos a cada uno de los 37 primeros entrevistados un formulario de retroalimentación preguntando si el entrevistado estaba de acuerdo con la clasificación elegida (estructura organizativa y propiedades suplementarias) para su contexto utilizando la siguiente escala Likert: fuertemente de acuerdo, débilmente de acuerdo, no sé, débilmente en desacuerdo, muy en desacuerdo. En caso de desacuerdo, había campos de texto libre para la explicación. También

preguntamos a los entrevistados si percibían nuestra taxonomía como completa y si agregarían o eliminarían elementos. Finalmente, también dejamos un campo libre para comentarios generales. Enviamos el formulario en cuatro lotes de veinte, cinco, cinco y siete correos electrónicos repartidos en los últimos cinco meses de nuestro período de entrevistas; utilizamos la retroalimentación de manera incremental para refinar nuestra teoría. También adjuntamos a los correos electrónicos un resumen que describe nuestra taxonomía. Recibimos 11 respuestas. Nueve participantes estuvieron totalmente de acuerdo con la clasificación recibida en cuanto a la estructura orgánica, mientras que dos de ellos estuvieron débilmente de acuerdo con ella. Nadie estuvo en desacuerdo. Cinco participantes estuvieron totalmente de acuerdo con la clasificación recibida con respecto a las propiedades suplementarias, mientras que uno de ellos estuvo débilmente de acuerdo con ella. En cuanto a la exhaustividad de nuestro modelo, siete participantes estuvieron totalmente de acuerdo con él, tres de ellos estuvieron débilmente de acuerdo con él, y uno de ellos no lo sabía. Este resultado sugiere la resonancia de los participantes con nuestra teoría, que se refiere al grado en que los hallazgos tienen sentido para los participantes. Las respuestas de texto libre de los participantes fueron valiosas para refinar la taxonomía. Concebimos las propiedades suplementarias a partir del análisis de la primera ronda de retroalimentación. Los comentarios de un entrevistado nos ayudaron a mejorar nuestro resumen de taxonomía (refinamos una figura para expresar mejor la idea del equipo de la plataforma). Además, algunos participantes expresaron su preocupación por la forma en que las diferentes partes de la organización actúan bajo diferentes patrones y cómo evoluciona esto. Consideramos tales preocupaciones ya que nuestra clasificación no es para toda la organización, sino para el contexto del entrevistado en un momento dado.

5.2. EVALUANDO NUESTRA TEORÍA EMERGENTE

Según Ralph, las buenas taxonomías deberían aumentar la eficiencia cognitiva y ayudar al razonamiento facilitando más inferencias [15]. Sin embargo, evaluar si una taxonomía satisface tales criterios de calidad exige un tipo diferente de enfoque de investigación que el que empleamos; por ejemplo, un estudio de caso [34]. Aunque los estudios de caso pueden probar teorías [44], (Tabla 5) son más útiles para demostrar cómo una teoría actual es incompleta o inadecuada para explicar el caso observado [12,11]. Un solo estudio de caso no prueba una teoría social. Además, incluso después de décadas de pruebas sólidas que confirman las teorías, los nuevos trabajos aún pueden expandirlas [13]. Por otro lado, la Teoría Fundamentada (GT) se centra en generar teoría en lugar de validar hipótesis preconcebidas. Aunque los investigadores pueden verse tentados a tratar de validar su teoría tan pronto como nace, Glaser y Strauss advierten que los enfoques de verificación dificultan el desarrollo de la teoría demasiado pronto [14]. Por lo tanto, en este trabajo, presentamos solo pasos incipientes relacionados con la evaluación teórica: escuchar retroalimentación sobre nuestra taxonomía (Sección 5.1) y compararla con otras clasificaciones existentes (Sección 6). Sin embargo, tales pasos evidencian la resonancia de nuestra teoría. Al seguir el marco de Guba para la evaluación de la investigación naturalista [35], la resonancia de los profesionales y la literatura también proporciona cierta credibilidad (validez interna, cuán plausibles o verdaderos son los hallazgos) y confirmabilidad (objetividad, oportunidades para corregir el sesgo de investigación). El resto de los criterios de

Guba son la confiabilidad (confiabilidad), que es proporcionada por nuestra cadena de evidencia, y la transferibilidad (validez externa o generalizabilidad), que está respaldada por nuestra diversa selección de participantes. Aunque es inadecuado evaluar nuestra teoría emergente como un producto terminado, podemos evaluar nuestro proceso de generación de la teoría [14, 15]. Esto se puede hacer verificando el cumplimiento de las prescripciones de GT y las recomendaciones de Ralph para la generación de taxonomía [15]. Se describió la adherencia a las directrices GT (es decir, muestreo teórico, codificación, sensibilidad teórica y saturación teórica) en la Sección 3. A continuación, discutimos nuestra adhesión a las recomendaciones de Ralph. Una advertencia de Ralph es que la teoría debe explicar, no prescribir. De esta manera, es esencial tener en cuenta que aunque nuestra teoría está destinada a ser utilizada por los profesionales en entornos prácticos, como es el objetivo de una teoría fundamentada, la teoría en sí misma proporciona una explicación del mundo, no una guía para la acción. Un impacto más severo en nuestro trabajo proviene del consejo de Ralph para favorecer las observaciones de primera mano sobre las entrevistas debido a los sesgos de los entrevistados. Aunque reconocemos sus preocupaciones, en nuestro caso las estructuras organizativas son demasiado abstractas para comprenderlas solo mediante la observación, incluso la observación de reuniones, sin más conversaciones con las personas observadas. Por lo tanto, nuestro contexto difiere de otras situaciones de ingeniería de software, como la observación de una sesión de programación en pareja. Sin embargo, el anonimato reduce estos sesgos al favorecer una actitud abierta de los participantes. Nos encargamos de no hacer las preguntas de investigación directamente a los participantes, lo que obligaría a nuestras ideas preconcebidas sobre ellos [15]. Elaboramos cuidadosamente preguntas de segundo nivel [34], que son más objetivas que las preguntas de investigación. Nuestro protocolo de entrevista9 presenta cada pregunta de la entrevista, seguida de su justificación. Reconocemos la importancia de triangular los resultados con otros tipos de datos y con otros participantes en las mismas organizaciones. Sin embargo, ahorramos estas estrategias para la fase futura de nuestra investigación. Observar más escenarios es más valioso para esta fase inicial de elaboración de la teoría que entrevistar a más personas en cada organización, lo que llevaría a entrevistas en menos empresas. Ralph aún advierte que los investigadores deben tener cuidado al seleccionar los criterios de evaluación para evaluar el desarrollo de una teoría. Dado que existen múltiples criterios y no existe un conjunto de criterios universalmente aceptados, los investigadores deben elegir criterios que tengan sentido para la teoría emergente [15]. No obstante, aplicamos a nuestra investigación tres criterios cruciales sugeridos por Ralph: *(i)* el estudio empírico está bien ejecutado y claramente descrito, *(ii)* hay una cadena explícita de evidencia desde los resultados hasta los datos de apoyo (lo que no excluye la existencia de saltos intuitivos no replicables [15]), y *(iii)* la necesidad de la teoría propuesta debe ser clara. El presente texto y la cadena completa de pruebas, proporcionados como material complementario, deben ser suficientes para que el lector juzgue estas preocupaciones.

6. TRABAJO RELACIONADO

Investigaciones recientes han discutido los beneficios y desafíos de la entrega continua [3,45,46,1,47]. Entre los desafíos, Chen et al. incluyen cuestiones organizativas relacionadas con las tensiones entre grupos dentro de una organización [3], lo que exige estudios sobre la organización de equipos. Algunos de estos estudios se han centrado en cómo los desarrolladores migran de equipos o empresas [48,22,49]. No obstante, la literatura sobre los arreglos entre

equipos para administrar la infraestructura de TI en un contexto de entrega continua sigue siendo limitada. A continuación, discutimos esta literatura existente [2,6–10] comparando sus clasificaciones de relaciones entre equipos con nuestra taxonomía.

En uno de los escritos fundacionales sobre DevOps [2], Humble y Molesky comienzan criticando la estructura del departamento en silos y la gestión por proyecto. Luego siguen abogando por equipos multifuncionales y gestión por producto. Sin embargo, también sugieren prácticas para fortalecer la colaboración entre el desarrollo y las operaciones, lo que tiene sentido en la estructura clásica de DevOps. Tales prácticas incluyen operadores que asisten a ceremonias ágiles y desarrolladores que contribuyen a la resolución de incidentes. Humble y Molesky también prevén grupos de operación que ofrezcan servicios de soporte (por ejemplo, integración continua) e infraestructura como servicio a los equipos de productos, lo que se relaciona en nuestra taxonomía con los equipos facilitadores y la estructura del equipo de la plataforma.

Nybom et al. presentan tres enfoques distintos para la adopción de DevOps [6]: *(i)* asignar responsabilidades de desarrollo y operaciones a todos los ingenieros; *(ii)* componer equipos multifuncionales de desarrolladores y operadores; y *(iii)* crear un equipo de DevOps para unir el desarrollo y las operaciones. Sin embargo, el artículo trata sobre un estudio de caso que coincide solo con el primer enfoque; los desarrolladores emprendieron tareas operativas y se promovió la colaboración entre los departamentos de desarrollo y operaciones. De acuerdo con nuestra taxonomía, tal escenario fue un intento de migrar de departamentos aislados a DevOps clásicos. Sin embargo, a pesar de algunos beneficios percibidos, surgieron nuevas fuentes de fricción entre los departamentos, y varios empleados no estuvieron de acuerdo con el enfoque adoptado. Asociamos estos resultados subóptimos a la falta reportada de inversiones en automatización, lo que sugiere que intentar cualquier adopción de DevOps sin apuntar a la entrega continua no es prometedor.

El Informe sobre el estado de DevOps de 2018 encuestó a los encuestados sobre las estructuras organizativas utilizadas en sus viajes de DevOps [7], ofreciendo un conjunto cerrado de alternativas: equipos multifuncionales responsables de servicios o aplicaciones específicas, equipos de DevOps dedicados, equipos de TI centralizados con múltiples equipos de desarrollo de aplicaciones, equipos de ingeniería de confiabilidad del sitio y equipos de servicio que proporcionan capacidades de DevOps (por ejemplo, creación de entornos de prueba, monitoreo). Sin embargo, el texto no describe más a fondo esas opciones. Por lo tanto, asociar nuestras estructuras a las opciones presentadas por la encuesta sería una actividad propensa a errores.

Skelton y Pais presentan nueve "topologías de DevOps" y siete antipatrones [8], como la más informal de nuestras fuentes de comparación: una publicación de blog. La presentación de cada topología y antipatrono es corta, carente de detalles sobre cómo las corporaciones los aplican. Ahora presentamos las correspondencias entre las topologías/antipatrones de DevOps y nuestra taxonomía. Los silos de desarrollo y operaciones corresponden a nuestra estructura de departamentos en silos. Dev do not need ops corresponde a nuestros equipos multifuncionales sin antecedentes de infraestructura. En algunas organizaciones que adoptaron DevOps clásicos, vimos el cambio de marca del equipo de infraestructura a SRE o DevOps (#I2, #I6, #I31), pero esta situación no coincidió del todo con el anti-patrón de sysadmin renombrado ya que hubo cambios culturales en los casos observados. Las operaciones integradas en el equipo de desarrollo

corresponden a nuestros equipos multifuncionales con profesionales de infraestructura dedicados, aunque vimos resultados positivos con esta configuración en #I1. En una organización aislada (#I30), también observamos el silo del equipo de DevOps. La colaboración de dev y ops corresponde a nuestra estructura clásica de DevOps. Ops como infraestructura como servicio (plataforma) corresponde a nuestros equipos de plataforma. Consideramos que la topología del equipo SRE coincide con nuestras estructuras clásicas de DevOps con infra como colaborador de desarrollo, aunque la descripción de la topología no incluye esta actividad SRE de codificar la aplicación para mejorar sus requisitos no funcionales [50].

El *libro Team Topologies* [9], de los mismos autores de la publicación del blog DevOps topologies, presenta cuatro patrones dinámicos de equipos para corporaciones productoras de software: equipos alineados
con el flujo, que entregan software en un flujo constante alineado con el negocio; *equipos de subsistemas complicados*, con personas altamente especializadas que trabajan en un problema complicado; *equipos facilitadores*, que proporcionan consultoría para equipos alineados con el flujo en un dominio técnico o de producto específico; y *equipos de plataforma*, que brindan servicios internos para el autoservicio mediante equipos alineados con el flujo, abstraen la infraestructura y aumentan la autonomía para los equipos alineados con el flujo.

El equipo *alineado con la transmisión* corresponde a lo que llamamos un equipo de producto o equipo de desarrollo en este documento. *El complicado equipo de subsistemas* no se considera en nuestra taxonomía, ya que está relacionado con la división del trabajo dentro del desarrollo solamente. El *equipo facilitador* propuesto por Skelton y Pais está muy cerca de lo que también llamamos un equipo habilitador (por ejemplo, entrevistado #I18 estaba en un equipo habilitador que brindaba consultoría sobre el rendimiento para los equipos de productos); sin embargo, Skelton y Pais defienden que dicha consultoría debe estar limitada en el tiempo, lo cual es un aspecto ausente de nuestros equipos facilitadores observados. Finalmente, el *equipo de plataforma* propuesto por Skelton y Pais incluye nuestra noción de equipo de plataforma; sin embargo, nuestro concepto se restringe a los servicios relacionados con el entorno de ejecución de las aplicaciones, es decir, mientras que consideramos a los equipos que brindan servicios de pipeline como equipos facilitadores, Skelton y Pais los considerarían como equipos de plataforma.

Otra diferencia significativa de las topologías de equipo a nuestro trabajo es que el libro busca presentar *las cosas como deberían ser*, mientras tratamos de resumir *cómo son*. En este sentido, aunque reconocemos la existencia de la estructura *clásica de DevOps*, no está incluida en las topologías del equipo, ya que los autores desalientan el traspaso, que provoca. También observamos que los términos "equipo de plataforma" y "equipo habilitador" surgieron de nuestros entrevistados, sin la influencia directa del libro Teams Topologies.

La mayor parte del trabajo discutido hasta ahora [6,2,7,8] presenta conjuntos de estructuras organizativas sin una elaboración empírica de cómo se concibieron dichos conjuntos. El libro *Team Topologies* sugiere que las topologías propuestas surgieron de observaciones de campo, pero carecían de una metodología científica. De esta manera, Shahin et al. [10] presentan el trabajo más cercano al nuestro, con un conjunto de estructuras basadas en datos de campo y directrices científicas.

Shahin et al. [10] realizaron entrevistas semiestructuradas en 19 organizaciones y encuestaron a 93 profesionales para investigar empíricamente cómo se organizan los equipos de desarrollo y operación en la industria del software para adoptar prácticas de entrega continua. Encontraron cuatro tipos de estructuras de equipo: *(i)* equipos separados de Desarrollo y Operaciones con mayor colaboración, *(ii) equipos de Desarrollo y Operaciones separados con un facilitador (s) en el medio*; *(iii) un pequeño equipo de operaciones con más responsabilidades para el equipo de desarrollo*, y *(iv) ningún equipo de operaciones visible*. Las estructuras *i* y *iii* se asignan a DevOps clásicos en nuestra taxonomía. La estructura *ii* corresponde a la adopción de un equipo de DevOps como puente entre el desarrollo y las operaciones. Uno de nuestros entrevistados informó que tal patrón ocurrió en el pasado en su organización (#I4) y también observamos al equipo de DevOps como un comité que reúne al liderazgo de desarrollo y operaciones en otra organización (#I36); por lo tanto, es probable que los equipos de DevOps que sirven como puentes ya no sean comunes. Finalmente, la estructura *iv* asigna a equipos multifuncionales. Shahin et al. no identificaron la estructura de los equipos de la plataforma, la alternativa más prometedora que encontramos en relación con el rendimiento de la entrega. Además, su trabajo no presenta la noción de transición entre las estructuras.

Shahin et al. [10] también exploraron el tamaño de las empresas que adoptan cada estructura. Encontraron que la estructura i es adoptada principalmente por las grandes corporaciones, mientras que la estructura iv se observó principalmente en las pequeñas. Estos hallazgos son corroborados por nuestros datos en la Tabla 4: DevOps clásico fue menos observado en organizaciones pequeñas, mientras que los equipos multifuncionales no fueron frecuentes en organizaciones grandes. Sin embargo, otros facilitadores pueden participar en la adopción de una estructura organizativa. Una mejor comprensión de estos factores y sus fuerzas está en nuestros planes para el trabajo futuro.

En un artículo reciente, Shahin y Babar recomiendan agregar especialistas en operaciones a los equipos [41], lo que favorece los equipos multifuncionales con profesionales de infraestructura dedicados.

7. LIMITACIONES

El lector debe tener en cuenta las limitaciones típicas de las teorías taxonómicas. La limitación más importante es que rara vez hacen predicciones probabilísticas en términos de variables dependientes e independientes, como lo hacen las teorías de varianza, comunes en Física [15]. Por lo tanto, no es apropiado discutir, para una teoría fundamentada, el número de entrevistados en términos de muestreo estadístico.

Una limitación para la credibilidad y la confiabilidad [35] es el anonimato de nuestros participantes, ya que no es posible replicar las entrevistas con las mismas personas. Sin embargo, hay una compensación crucial aquí: algunos participantes pueden no aceptar una invitación para participar en una entrevista no anónima. Además, el anonimato hace frente al sesgo de deseabilidad social [51]; entrevistar a la misma persona después de algún tiempo no garantiza las mismas respuestas.

Aunque nos basamos en trabajos anteriores [18] para adoptar la construcción del rendimiento de la entrega, necesitábamos adaptar los umbrales originales debido a las razones

expuestas en la Sección 2. Tampoco diferenciamos a los de bajo rendimiento de los de rendimiento medio, ya que juzgamos que dicha diferenciación no ayudaría a distinguir cómo las estructuras contribuyen al rendimiento de la entrega, en parte porque los de bajo y medio rendimiento son mucho más similares entre sí que en comparación con los de alto rendimiento [52]. Por lo tanto, al cambiar algunas decisiones relacionadas con el rendimiento de la entrega, otro investigador podría llegar a conclusiones diferentes basadas en los mismos datos. Manejamos esta preocupación principalmente al establecer nuestras definiciones con respecto a este tema. Sin embargo, GT no garantiza que dos investigadores que trabajan en paralelo con los mismos datos logren resultados idénticos [14].

En un proceso ideal, diferentes investigadores analizarían las interpuntrices de forma independiente, evitando algún posible sesgo de acuerdo favorecido por las revisiones. Sin embargo, el costo del análisis independiente sería demasiado oneroso (analizar una entrevista lleva horas). Con respecto al proceso de revisión, en lugar de confiar en las transcripciones producidas por un solo autor, todos los investigadores podrían haber escuchado los registros originales. Sin embargo, tal enfoque también sería demasiado costoso en términos de compromiso de tiempo. Alternativamente, dos coautores participaron en algunas entrevistas iniciales para estandarizar los procedimientos de transcripción y evaluar cómo el primer autor realizó estas entrevistas.

Somos conscientes de que las grandes empresas suelen presentar grupos en diferentes niveles de madurez, y que dichos grupos podrían clasificarse de manera diferente si hubiéramos elegido diferentes entrevistados. Para verificar esto, entrevistamos a dos personas de la misma empresa (#I16 y #I18) que trabajan en equipos diferentes. Observamos que, de hecho, las pautas de organización no eran idénticas. Este efecto también ocurre cuando se hace la transición de una estructura a otra, ya que la transición puede ser un proceso largo y tomar diferentes ritmos en diferentes segmentos de la organización. Por lo tanto, el lector debe tener en cuenta que nuestras descripciones caracterizan los contextos de nuestros encuestados, no las organizaciones en su totalidad.

Finalmente, una característica particular de los equipos de desarrollo de software es que no son fijos; los desarrolladores a menudo entran y salen de un equipo. Por lo tanto, no hay razón para pensar que las estructuras organizativas propuestas son inmutables. En cambio, los lectores deben considerar nuestras estructuras organizativas al avanzar hacia un escenario de entrega continua. Después de sentirse cómodos y comenzar a sobresalir con desautoría continua, si es necesario, los profesionales podrían adaptar la estructura organizativa empleada para ser efectiva en otros contextos.

8. CONCLUSIÓN

En este artículo, presentamos una teoría fundamentada emergente que aborda la pregunta de investigación "¿Qué estructuras organizativas adoptan las organizaciones productoras de software para estructurar las actividades de operaciones (implementación de aplicaciones, configuración de infraestructura y operación de servicios en tiempo de ejecución) entre grupos de desarrollo e infraestructura?" Encontramos cuatro estructuras organizativas:

1. Departamentos aislados (con siete propiedades principales);

2. DevOps clásico (con ocho propiedades principales y una propiedad suplementaria);
3. Equipos multifuncionales (con cuatro propiedades básicas y tres propiedades suplementarias);
4. Equipos de plataforma (con nueve propiedades principales y tres propiedades complementarias).

Más allá de las siete propiedades suplementarias directamente relacionadas con una estructura cada una, también encontramos otras dos propiedades suplementarias aplicables a más de una estructura. Las propiedades básicas que describen cada estructura organizativa más las propiedades suplementarias responden a la pregunta de subinvestigación *"¿Cuáles son las propiedades de cada una de estas estructuras organizativas?"*

Además, este estudio responde a la pregunta de la sub-investigación *"¿Son algunas estructuras organizativas más propicias para la entrega continua que otras?"* planteando la hipótesis emergente de que la estructura del equipo de la plataforma es más adecuada para lograr una entrega continua dada su relación con el rendimiento de la entrega. También observamos que los departamentos aislados se relacionan más con organizaciones con un rendimiento menos que alto. Además, no hubo una relación aparente entre el rendimiento de la entrega y las otras estructuras (DevOps clásicos y equipos multifuncionales). No obstante, afirmamos críticamente que alcanzar un alto rendimiento de entrega no es una necesidad para todas las organizaciones productoras de software, ya que la adopción de equipos de plataforma no es adecuada para todas las organizaciones. Un tema prometedor para futuras investigaciones es investigar si el dominio de la organización influye en la necesidad de un alto rendimiento de entrega.

Nuestro trabajo tiene implicaciones para **la práctica**. Las empresas de software podrían darse cuenta de que hay varios tipos de estructuras organizativas que podrían adoptar para sobresalir en la entrega continua y planificar a qué estructura organizativa están interesadas en mudarse, maximizando sus posibilidades de tener éxito en la transición. Además, aclaramos los roles que los participantes tienen que desempeñar en cada estructura organizativa. Esta evidencia podría ayudar a los profesionales a cooperar con menos fricción hacia la transformación organizacional.

Nuestro trabajo también tiene implicaciones para la investigación. Los elementos de nuestra taxonomía, en la Fig. 3, proporcionan un vocabulario común para apoyar la formulación de nuevas preguntas de investigación. Por ejemplo, los investigadores pueden investigar el impacto de cada propiedad en otras perspectivas (por ejemplo, arquitectura de software, seguridad, administración de bases de datos). Otro esfuerzo valioso sería investigar la relación entre los elementos de nuestra taxonomía y la organización interna de los grupos de desarrollo y operación, especialmente los equipos de plataforma.

Además, observamos que la automatización de pruebas aún no se practica adecuadamente en muchas empresas de software, como también señalaron Olsson et al. [46], y que la falta de pruebas es un factor limitante para lograr un alto rendimiento de entrega. Esto sugiere oportunidades de investigación para proponer técnicas automatizadas de generación de pruebas, especialmente en entornos en los que las pruebas son intrínsecamente difíciles, como los juegos o IoT. Además, notamos que los participantes practicaban DevOps mientras mantenían una aplicación monolítica, lo que contrasta con la literatura que asocia fuertemente DevOps con

microservicios. Esto sugiere que los investigadores deberían investigar más a fondo las diferencias en la aplicación de DevOps a monolitos y sistemas basados en microservicios. Del mismo modo, dado que también observamos que algunos participantes mantenían un núcleo monolítico con microservicios periféricos y lograban un alto rendimiento de entrega con los microservicios, los investigadores podrían proponer técnicas y herramientas novedosas que podrían (semi) extraer automáticamente los servicios periféricos de las aplicaciones monolíticas.

Para el trabajo futuro, planeamos delinear mejor las fuerzas que impulsan a las organizaciones a elegir diferentes estructuras. También tenemos la intención de emplear nuestra taxonomía para discutir escenarios de corporaciones que aún no hemos visitado, lo que puede corroborar la resonancia de la teoría.