POBLACIÓN Y MUESTRA EN ESTUDIOS DE CASO EN INGENIERÍA DE SOFTWARE: MARCO Y GUÍA PRÁCTICA

PhD. Luis Roberto Pérez Rios
Posgrado SOE – UAGRM
https://orcid.org/0000-0002-8385-1016
Santa Cruz, Bolivia | luis.roberto@alenasoft.com

https://doi.org/10.23670 FT.2026.1.39

Recibido 08/05/2026 - Aceptado 29/05/2026

RESUMEN

La investigación en ciencias de la computación e ingeniería de software recurre con frecuencia al estudio de caso como diseño metodológico, pero la forma en que se declaran y justifican los conceptos de población y muestra en estos trabajos es a menudo imprecisa o directamente omitida, lo que debilita la credibilidad de sus hallazgos ante revisores formados en metodología cuantitativa. Este artículo de revisión examina el fundamento epistemológico del estudio de caso en ingeniería de software, analiza cómo los conceptos de población y muestra se reformulan en este contexto, y propone un marco operativo para declarar, justificar y delimitar tanto la población como el caso seleccionado en investigaciones de este tipo. Se revisan los marcos metodológicos de referencia canónicos del área y se articulan sus principios en una guía de cinco pasos aplicable a trabajos originales en las áreas de ingeniería de software, arquitectura de sistemas, seguridad informática y desarrollo de herramientas. El artículo concluye que la especificación rigurosa de la población y la justificación explícita del tipo de muestreo son condiciones necesarias para que un estudio de caso en ingeniería produzca conocimiento generalizable, y que esta generalización opera por vía analítica –no estadística–, lo que tiene consecuencias directas sobre cómo se redactan las secciones de metodología y discusión.

Palabras clave: estudio de caso, ingeniería de software, población y muestra, metodología de investigación, generalización analítica

ABSTRACT

Research in computer science and software engineering frequently relies on case study design, yet the way population and sample are declared and justified in these works is often imprecise or outright absent, undermining credibility with reviewers trained in quantitative methodology. This review article examines the epistemological foundation of case study research in software engineering, analyzes how the concepts of population and sample are reformulated in this context, 1Boletín Científico Fronteras Tecnológicas andproposes anoperational framework for declaring, justifying, and delimiting both the population and the selected case in this type of research. The canonical methodological frameworks of the field are reviewed and their principles are articulated into a five-step guide applicable to original work in software engineering, systems architecture, security, and tool development. The article concludes that rigorous population specification and explicit sampling justification are necessary conditions for a case study in engineering to produce generalizable knowledge, and that this generalization operates analytically –not statistically–, with direct consequences for how methodology and discussion sections are written.

Keywords: case study, software engineering, population and sample, research methodology, analytical generalization

INTRODUCCIÓN

La investigación empírica en ciencias de la computación e ingeniería de software ha experimentado en las últimas dos décadas una expansión significativa en el uso de métodos cualitativos y mixtos, entre los cuales el estudio de caso ocupa un lugar central (Runeson & Höst, 2009; Wohlin et al., 2024). Esta tendencia refleja un reconocimiento gradual de que muchos fenómenos relevantes en el desarrollo de software —la efectividad de una metodología, el impacto de una herramienta, la calidad arquitectónica de un sistema— no pueden ser estudiados adecuadamente mediante experimentos controlados o encuestas estadísticas, porque ocurren en contextos organizativos e históricos específicos que el control experimental eliminaría (Flyvbjerg, 2006). El estudio de caso permite, en cambio, preservar esa riqueza contextual y examinar el fenómeno en su entorno natural.

Sin embargo, la adopción de este método no ha sido acompañada siempre por una comprensión suficiente de sus fundamentos epistemológicos, y en particular de cómo los conceptos de población y muestra —centrales en la investigación cuantitativa— se reformulan cuando el objeto de estudio es un sistema de software, un proceso de desarrollo o una herramienta tecnológica, y no un grupo humano. En consecuencia, es común encontrar en publicaciones de ingeniería de software secciones de metodología que omiten por completo la justificación de por qué se seleccionó el caso estudiado, qué dominio de sistemas representa y a qué clase de afirmaciones generalizables da lugar el análisis. Esta omisión no es trivial: un revisor metodológicamente riguroso puede rechazar un trabajo bien ejecutado simplemente porque la sección de metodología no articula con claridad los límites de validez del estudio. La magnitud del problema ha sido documentada empíricamente: Baltes y Ralph (2022) revisaron sistemáticamente las prácticas de muestreo en investigación empírica de ingeniería de software y encontraron que el área presenta una crisis de generalización caracterizada por la prevalencia del muestreo de conveniencia no declarado y por la confusión sistemática entre representatividad estadística y analítica; Wohlin y Rainer (2022) identificaron, por su parte, que un número significativo de trabajos que se autoidentifican como estudios de caso no satisfacen los criterios metodológicos que definen el diseño. Ambos hallazgos confirman que el problema no es una percepción docente ni un riesgo potencial, sino una deficiencia activa y medible en la literatura del área.

La pertinencia de este trabajo radica en que la ingeniería de software se encuentra en una posición epistemológicamente ambigua: es una disciplina de ingeniería —en el sentido de que produce artefactos funcionales— pero sus objetos de estudio son en gran medida construcciones sociotécnicas cuyo comportamiento depende del contexto humano y organizacional en que operan (Shaw, 2003). Esta ambigüedad hace que ni el paradigma experimental de las ciencias naturales ni el paradigma interpretativo de las ciencias sociales sean completamente adecuados para todos sus problemas de investigación, y que el estudio de caso ocupe ese espacio intermedio donde la ingeniería y la ciencia social se encuentran.

El objetivo general del presente artículo es proporcionar al investigador en ciencias de la computación e ingeniería de software un marco teórico integrado y una guía operativa para declarar y justificar los conceptos de población y muestra en investigaciones basadas en estudio de caso. De este objetivo se derivan cuatro objetivos específicos: (OE1) revisar los fundamentos epistemológicos del estudio de caso en ingeniería de software y sus implicaciones para la selección y declaración del caso; (OE2) analizar la reformulación de los conceptos de población y muestra en el contexto del estudio de caso en ingeniería; (OE3) sintetizar los criterios de selección del caso derivados de los marcos metodológicos canónicos del área; y (OE4) proponer un procedimiento operativo de cinco pasos para declarar y justificar la población y la muestra en la sección de metodología de trabajos con diseño de caso.

METODOLOGÍA

El presente artículo constituye una revisión narrativa y focalizada de los marcos metodológicos canónicos sobre el estudio de caso como diseño de investigación, con énfasis específico en su aplicación al campo de la ingeniería de software. Este tipo de revisión —denominado también revisión teórica o conceptual en la literatura metodológica— se diferencia de la revisión sistemática (Kitchenham et al., 2007) en que no busca exhaustividad en la cobertura de un corpus de publicaciones mediante protocolos de búsqueda replicables, sino profundidad interpretativa en el análisis de las obras que han estructurado el debate metodológico en el área. La pertinencia de este enfoque es coherente con el propósito declarado del artículo: no catalogar el estado del arte, sino ofrecer al investigador en ciencias de la computación un marco teórico integrado y una guía operativa aplicable directamente en la redacción de sus secciones de metodología.

Fuentes de información

Las fuentes primarias de este artículo son las obras que han alcanzado estatus canónico en la investigación empírica en ingeniería de software: Yin (2018) como referencia central para el diseño del estudio de caso; Runeson y Höst (2009) y Runeson et al. (2012) para las guías específicas al área de software; Wohlin et al. (2024) para la experimentación empírica en ingeniería; y Eisenhardt (1989) para la teoría del muestreo teórico en estudios de caso. Se incorporaron adicionalmente obras que amplían el espectro epistemológico del análisis: Flyvbjerg (2006) sobre los malentendidos frecuentes en torno al método; Stake (1995) sobre la distinción intrínseco/instrumental del caso; Benbasat et al. (1987) sobre los criterios de defensibilidad metodológica; Lincoln y Guba (1985) sobre la transferibilidad como alternativa interpretativa a la generalización; Kitchenham et al. (2007) y Shaw

(2003) como marcos de referencia sobre la calidad de la investigación empírica en el campo; y Baltes y Ralph (2022) y Wohlin y Rainer (2022) como evidencia empírica reciente sobre la prevalencia y naturaleza de los problemas de muestreo en la literatura del área. El acceso a estas obras se realizó a través de SpringerLink, Wiley Online Library, SAGE Publications Digital Library, ACM Digital Library y Google Scholar.

Criterios de inclusión y exclusión

Los criterios de inclusión aplicados fueron: (a) obras con reconocido estatus canónico o amplia citación en la comunidad de investigación empírica en ingeniería de software, verificado por recuento de citas en Google Scholar y presencia en bibliografías de guías metodológicas publicadas en revistas del área; (b) publicaciones en revistas indexadas o libros editados por casas académicas reconocidas; y (c) obras que abordan directamente el diseño del estudio de caso o sus criterios de validez en contextos de ingeniería o ciencias sociales aplicadas. Los criterios de exclusión fueron: (a) obras de literatura gris sin revisión por pares; (b) publicaciones que abordan el estudio de caso únicamente como técnica pedagógica sin discusión metodológica; y (c) trabajos que no aportan elementos diferenciadores respecto de los marcos canónicos ya seleccionados, evitando redundancia sin ganancia analítica.

Palabras clave y estrategia de búsqueda

La identificación inicial de fuentes se realizó mediante búsqueda en Google Scholar y ACM Digital Library con los descriptores: case study research software engineering, empirical software engineering methodology, case selection criteria, theoretical sampling case study, analytic generalization, population and sample qualitative research, y sus equivalentes en español. La selección final de los marcos de referencia no se basó en el resultado cuantitativo de la búsqueda—dada la naturaleza narrativa de la revisión— sino en la convergencia de citas entre los propios marcos: las obras seleccionadas son aquellas que se citan mutuamente de forma sistemática en el sub-campo de la investigación empírica en ingeniería de software, lo que constituye un criterio de pertinencia teórica coherente con el enfoque de la revisión.

Limitaciones metodológicas

Al tratarse de una revisión narrativa y no sistemática, el proceso de selección de fuentes incorpora un grado de discrecionalidad que una revisión sistemática eliminaría mediante protocolos de búsqueda replicables. El principal riesgo asociado es el sesgo de confirmación: la selección de marcos que refuerzan la posición teórica del artículo en lugar de cuestionar sus premisas. Este riesgo se mitiga parcialmente por la incorporación de perspectivas alternativas —Lincoln y Guba (1985) como contraste interpretativo a Yin, Stake (1995) como complemento a la tradición positivista del diseño de caso— y por la declaración explícita de que la taxonomía de errores frecuentes tiene naturaleza observacional y no deriva de una revisión sistemática de publicaciones. Adicionalmente, la selección de obras canónicas incorpora fuentes publicadas hasta 2024 — incluyendo la edición actualizada de Yin (2018), la segunda edición de Wohlin et al. (2024) y los estudios empíricos recientes de Baltes y Ralph (2022) y Wohlin y Rainer (2022)—, lo que garantiza que el marco de referencia refleja el estado actual del debate metodológico en el área.

RESULTADOS Y DISCUSIÓN

El estudio de caso como diseño de investigación en ingeniería de software

Definición y características fundamentales: El estudio de caso es, según la definición ampliamente citada de Yin (2018), una investigación empírica que examina un fenómeno contemporáneo en profundidad y en su contexto real, especialmente cuando los límites entre el fenómeno y el contexto no son claramente evidentes. Esta definición tiene tres implicaciones metodológicas directas para la ingeniería de software. Primera: el estudio de caso es pertinente cuando el fenómeno —por ejemplo, la adopción de una práctica ágil, la presencia de deuda técnica, la efectividad de una herramienta de análisis estático— no puede separarse del contexto organizacional, técnico e histórico en que ocurre sin perder información esencial. Segunda: el estudio de caso no es un diseño de conveniencia para cuando no hay suficientes sujetos de estudio, sino un diseño específicamente adecuado para ciertas preguntas de investigación. Tercera: un estudio de caso puede ser completamente riguroso y producir conocimiento científicamente válido, pero esa validez opera a través de mecanismos diferentes a los de la investigación experimental.

Runeson y Höst (2009) ofrecen una clasificación del estudio de caso especialmente ajustada a la ingeniería de software, distinguiendo cuatro propósitos principales: exploratorio (generar hipótesis para investigaciones posteriores), descriptivo (documentar cómo ocurre un fenómeno), explicativo (establecer relaciones causales en contexto) y mejorativo (estudiar el efecto de una intervención). Cada uno de estos propósitos tiene implicaciones distintas sobre cómo se selecciona el caso, qué datos se recogen y cómo se interpretan los resultados. Un investigador que no declara explícitamente el propósito de su estudio dificulta al lector la evaluación de si el diseño metodológico es coherente con lo que el trabajo pretende demostrar. Esta observación metodológica tiene fundamento empírico: Wohlin y Rainer (2022) han examinado críticamente trabajos publicados en ingeniería de software que se autoidentifican como estudios de caso y encuentran que un número significativo no satisface los criterios que definen el método; la inconsistencia en el uso del término es, por tanto, un problema activo en el área, no meramente un riesgo potencial.

Stake (1995) complementa esta perspectiva desde la tradición interpretativa del estudio de caso: mientras que Yin (2018) enfatiza el rigor del diseño de investigación

y la validez de los constructos, Stake distingue entre el estudio de caso intrínseco —donde el caso importa por sí mismo, independientemente de cualquier generalización— y el instrumental —donde el caso sirve para iluminar un fenómeno más amplio—. En el contexto de la ingeniería de software, la mayoría de los estudios responden al segundo tipo: el sistema analizado es el medio para generar conocimiento sobre una clase de sistemas, no un fin en sí mismo. Esta distinción es operativamente relevante porque confirma la Tercera implicación de Yin (2018): la validez del estudio de caso no reside en la representatividad estadística del caso, sino en que los mecanismos estudiados sean los correctos para el fenómeno de interés, lo cual exige declarar con precisión el dominio de sistemas al que esos mecanismos son analíticamente extrapolables.

La cuestión de la generalización: La objeción más frecuente al estudio de caso como método científico es la que señala que sus conclusiones no son generalizables porque se basan en un solo caso —o en muy pocos casos— que no puede ser representativo estadísticamente de ninguna población. Esta objeción, aunque intuitivamente comprensible, confunde dos tipos de generalización que la metodología científica distingue con claridad (Yin, 2018). La generalización estadística es la que opera en los estudios con muestra representativa: se mide un atributo en n individuos seleccionados aleatoriamente del universo y se infiere que ese atributo tiene la misma distribución en toda la población, con un margen de error calculable. La generalización analítica, en cambio, es la que opera en los estudios de caso: el investigador demuestra que los hallazgos del caso son consistentes con una teoría o con un mecanismo causal conocido, y esa consistencia permite afirmar que el mismo mecanismo actuaría en otros contextos con características análogas, independientemente de cuántos casos se hayan estudiado.

Este principio tiene una consecuencia práctica directa para la sección de metodología: el investigador no debe justificar el número de casos con argumentos estadísticos, sino argumentar por qué el caso seleccionado es representativo del mecanismo o fenómeno que se estudia. Flyvbjerg (2006) señala que incluso un único caso —si está bien seleccionado— puede refutar una teoría establecida, lo que demuestra que el valor epistémico de un estudio de caso no es proporcional a su tamaño, sino a la calidad de la selección y del análisis. En la práctica de la ingeniería de software, esto significa que lo que importa no es cuántos sistemas se analizaron, sino si el sistema analizado fue seleccionado con criterios explícitos que lo hacen representativo del dominio de interés.

Reformulación de población y muestra en ingeniería de software

La población como dominio de fenómenos: En la investigación con encuesta o experimento, la población es el conjunto de individuos —personas, empresas, respuestas posibles— sobre los que se pretende generalizar los hallazgos. El investigador selecciona una muestra de ese conjunto, la estudia y extrapola las conclusiones al conjunto completo. Esta lógica presupone que los individuos de la población son comparables entre sí en los atributos relevantes y que la variabilidad observada en la muestra refleja la variabilidad real del conjunto.

En la investigación de ingeniería de software con estudio de caso, el objeto de estudio no es una persona ni una organización como tal, sino un artefacto o proceso de naturaleza técnica: un sistema de software, una arquitectura, una metodología de desarrollo, una herramienta o una práctica de equipo. En consecuencia, la población es el dominio de artefactos o procesos al que el hallazgo es potencialmente aplicable, es decir, la clase de sistemas o situaciones que comparten las características estructurales que hacen posible el fenómeno estudiado. Wohlin et al. (2024) describen esta reformulación con el concepto de objeto de estudio y contexto: la población es la clase de objetos y contextos sobre los que se afirma que la relación estudiada se sostiene.

Por tanto, definir la población en un estudio de ingeniería de software equivale a responder: ¿en qué clase de sistemas, procesos u organizaciones serían aplicables los hallazgos de este trabajo? Esta pregunta no es retórica; su respuesta delimita los límites de validez del estudio y protege al investigador de realizar afirmaciones más amplias de lo que el diseño permite. Si el investigador no responde esta pregunta explícitamente, el lector —y el revisor— la responderán por él, a menudo de forma más restrictiva.

La muestra como caso seleccionado: Dado que la generalización en el estudio de caso opera de forma analítica y no estadística, la muestra no necesita ser aleatoria ni representativa en términos estadísticos. Lo que sí debe ser es intencionalmente seleccionada, con criterios documentados que justifiquen por qué ese caso particular es adecuado para estudiar el fenómeno de interés. Eisenhardt (1989) denomina a este procedimiento muestreo teórico (theoretical sampling): el caso se selecciona por su potencial para revelar el mecanismo teórico que se estudia, no por su frecuencia en la población.En la práctica, esto significa que el investigador debe documentar explícitamente los criterios de inclusión del caso —qué características debe tener para ser objeto de estudio— y, cuando sea posible, los criterios de exclusión —qué características lo descalificarían—. Esta documentación cumple tres funciones: permite al lector evaluar si la selección fue apropiada para el propósito declarado, permite a investigadores futuros replicar el proceso con casos diferentes del mismo dominio, y protege al investigador de la acusación de selección oportunista —es decir, de haber elegido el caso porque era conveniente o familiar, no porque fuera metodológicamente adecuado.

Tipos de diseño de caso: único y múltiple: La decisión de estudiar un único caso o múltiples casos responde a consideraciones metodológicas distintas. Yin (2018) identifica cinco justificaciones para el diseño de caso

único: (a) el caso es decisivo (critical case) para probar o refutar una teoría bien formulada; (b) el caso es extremo o único en sus características (extreme case; (c) el caso es representativo (typical case) del fenómeno de interés; (d) el caso es revelador (revelatory case), es decir, se trata de un fenómeno antes inaccesible para la investigación; o (e) el caso es longitudinal, estudiado en múltiples puntos en el tiempo.

Para la ingeniería de software, el caso representativo es el justificante más habitual: se selecciona un sistema de software que sea típico de una clase más amplia, con el argumento de que los hallazgos sobre ese caso particular tienen valor como demostración de que el fenómeno existe y puede describirse. Este es el diseño apropiado, por ejemplo, cuando se demuestra la viabilidad de una metodología sobre un sistema concreto —el argumento no es que todos los sistemas del dominio tendrán exactamente los mismos resultados, sino que el proceso es aplicable a sistemas de ese tipo.

El diseño de casos múltiples, en cambio, es adecuado cuando se estudian variaciones del fenómeno en distintos contextos, cuando se busca fortalecer la generalización analítica mediante la replicación literal o teórica (Yin, 2018), o cuando las diferencias entre casos son ellas mismas parte del hallazgo. Wohlin et al. (2024) señalan que en ingeniería de software, los casos múltiples son frecuentes en estudios sobre prácticas de equipo —donde la variabilidad entre organizaciones es parte central del resultado— pero menos habituales en estudios sobre herramientas o metodologías, donde un caso representativo bien documentado puede ser suficiente para demostrar la viabilidad o eficacia de la propuesta.

Tipos de muestreo en estudios de caso en ingeniería

Muestreo intencional o por propósito El muestreo intencional —también denominado muestreo por propósito o purposive sampling— es el tipo más frecuente en estudios de caso en ingeniería de software. Consiste en seleccionar el caso que mejor satisface los criterios establecidos para el propósito del estudio. En consecuencia, la calidad del muestreo no se mide por el número de casos seleccionados, sino por la coherencia entre los criterios de selección y el propósito declarado de la investigación.

Los criterios típicos de muestreo intencional en ingeniería de software incluyen: complejidad representativa del sistema —suficiente para que el fenómeno estudiado se manifieste, pero acotada para que el análisis sea manejable—; disponibilidad de documentación —el caso debe estar suficientemente documentado para que la recolección de datos sea posible y verificable—; diversidad de características internas relevantes para el estudio —por ejemplo, múltiples componentes, flujos de datos, actores—; y acceso o familiaridad del investigador con el sistema, siempre que ese acceso no introduzca sesgos no declarados. Benbasat et al. (1987) identifican esta combinación de criterios como la condición para que el caso sea metodológicamente defensible, más allá de consideraciones de conveniencia.

Muestreo teórico El muestreo teórico, propuesto originalmente en el marco de la grounded theory por Glaser y Strauss (1967) y adaptado para el estudio de caso por Eisenhardt (1989), selecciona los casos en función de su capacidad para extender, refutar o matizar la teoría que se está construyendo o evaluando. A diferencia del muestreo intencional clásico, donde los criterios se definen antes de iniciar el trabajo de campo, el muestreo teórico puede ser iterativo: el análisis de un primer caso revela qué tipo de caso adicional sería más informativo para continuar el desarrollo teórico.

En ingeniería de software, el muestreo teórico es especialmente pertinente cuando el objetivo del trabajo es proponer o refinar una teoría sobre un fenómeno —por ejemplo, una teoría sobre las condiciones bajo las cuales ciertos patrones arquitectónicos generan deuda técnica acumulable.

En ese contexto, el investigador puede seleccionar deliberadamente casos que maximicen la variación en las variables de interés, con el propósito de encontrar los límites del fenómeno, en lugar de buscar la confirmación sistemática de lo ya conocido.

Muestreo de conveniencia y sus limitaciones El muestreo de conveniencia —seleccionar el caso que está disponible o que el investigador conoce de antemano— es el tipo más frecuente en la práctica real de la ingeniería de software, aunque rara vez se declara como tal. Un proyecto de grado o de maestría generalmente estudia el sistema en el que el investigador trabaja o ha trabajado; un artículo de conferencia evalúa una herramienta sobre el proyecto de código abierto más accesible.

Esta práctica no invalida el estudio, pero sí exige que el investigador argumente explícitamente por qué ese caso disponible cumple los criterios del dominio de interés. Baltes y Ralph (2022), en una revisión crítica de las prácticas de muestreo en investigación empírica de ingeniería de software, documentan que el muestreo aleatorio es raro, las estrategias sofisticadas son aún menos frecuentes, y que los conceptos de representatividad y aleatoriedad son frecuentemente mal comprendidos en los trabajos publicados —lo que los autores denominan una crisis de generalización (generalizability crisis) en el área—, siendo el muestreo de conveniencia no declarado uno de sus factores contribuyentes más recurrentes.

El problema del muestreo de conveniencia no declarado es que el lector no puede distinguir si el caso fue seleccionado por su valor metodológico o por su disponibilidad; esa ambigüedad introduce dudas sobre si los hallazgos serían replicables en un caso seleccionado con criterios más rigurosos.

En consecuencia, la recomendación práctica es que incluso cuando el caso fue seleccionado por conveniencia, el investigador documente los criterios de inclusión que ese caso satisface y los criterios

que, de no satisfacerse, habrían llevado a descartar el caso, convirtiendo retrospectivamente la selección de conveniencia en una selección intencional documentada

Criterios de selección del caso: guía operativa

Los cinco criterios fundamentales A partir de la revisión de los marcos metodológicos de Runeson y Höst (2009) y su desarrollo posterior en Runeson et al. (2012), Yin (2018) y Wohlin et al. (2024), es posible identificar cinco criterios cuya documentación es necesaria —aunque no siempre suficiente— para justificar la selección de un caso en ingeniería de software.

Criterio 1 — Representatividad estructural (Wohlin et al., 2024; Yin, 2018). El caso debe poseer las características estructurales del dominio de población declarado. Si la población es "sistemas de software empresarial multicapa", el caso debe tener al menos los estratos funcionales típicos de ese tipo de sistemas.

Este criterio garantiza que el mecanismo que se estudia pueda efectivamente manifestarse en el caso seleccionado; si el caso no tiene las características estructurales mínimas, el hallazgo —positivo o negativo— carece de relevancia para el dominio.

Criterio 2 — Complejidad suficiente y acotada (Runeson & Höst, 2009; Runeson et al., 2012). El caso debe ser suficientemente complejo para que el fenómeno de interés se produzca con la riqueza necesaria para el análisis, pero suficientemente acotado para que el investigador pueda completar el estudio en profundidad. Esta tensión entre complejidad y manejabilidad es uno de los equilibrios más difíciles del diseño de caso en ingeniería, y la decisión sobre dónde establecer el límite debe ser argumentada explícitamente en dos dimensiones.

El umbral mínimo de complejidad está determinado por la naturaleza del fenómeno: un análisis de amenazas de seguridad requiere múltiples componentes y flujos que crucen al menos un límite de confianza, de modo que un sistema con un único componente sin interacción exterior no satisfaría este criterio independientemente de su disponibilidad.

El umbral máximo de complejidad, en cambio, está dado por los recursos del estudio: un caso que requiere analizar centenares de componentes interrelacionados puede ser teóricamente representativo pero prácticamente inmanejable, lo que comprometería la profundidad del análisis. Ambos umbrales deben declararse en la sección de metodología, con indicación de cómo el caso seleccionado los satisface.

Criterio 3 — Documentación verificable (Yin, 2018; Runeson et al., 2012). El caso debe estar suficientemente documentado para que la recolección de datos sea posible y los hallazgos sean trazables.

En el contexto de la ingeniería de software, esto significa que la arquitectura, las decisiones de diseño relevantes y el comportamiento esperado del sistema deben estar especificados de forma accesible para el investigador. Cuando la documentación es parcial —como ocurre frecuentemente con sistemas heredados— debe declararse explícitamente qué aspectos están documentados y qué aspectos requieren inferencia del investigador, así como los procedimientos utilizados para validar esa inferencia.

Criterio 4 — Relevancia para el fenómeno estudiado (Yin, 2018; Eisenhardt, 1989). El caso debe exhibir el fenómeno de interés o, al menos, las condiciones que hacen posible que ese fenómeno se manifieste. Si el estudio versa sobre la efectividad de un proceso de análisis de seguridad, el caso seleccionado debe ser un sistema que presente vulnerabilidades potenciales susceptibles de ser identificadas mediante ese proceso; un sistema que ya ha sido completamente auditado y remediado no permitiría evaluar la eficacia del proceso de identificación. Este criterio parece obvio, pero su omisión en la documentación metodológica da lugar a estudios cuya pregunta de investigación y caso seleccionado están desalineados.

Criterio 5 — Condiciones controladas o declaradas (Yin, 2018; Wohlin et al., 2024). El caso debe tener condiciones de diseño o de operación suficientemente conocidas para que el investigador pueda distinguir los hallazgos del estudio de los artefactos introducidos por variaciones no declaradas del caso. En ingeniería de software, esto puede lograrse mediante el uso de sistemas sintéticos diseñados para el estudio —donde todas las decisiones de diseño son conocidas de antemano— o mediante la documentación exhaustiva de las decisiones de diseño del sistema real, incluyendo sus supuestos y restricciones.

Los sistemas sintéticos tienen la ventaja de que eliminan la incertidumbre sobre el estado inicial, aunque introducen la limitación de que sus características son construidas por el investigador y pueden no reflejar fielmente la complejidad de los sistemas reales del dominio.

La tabla de criterios como artefacto metodológico Una práctica recomendable adoptada en investigaciones de ingeniería de software publicadas en revistas de primer nivel como Empirical Software Engineering, Journal of Systems and Software o IEEE Transactions on Software Engineering— es presentar los criterios de selección del caso en forma de tabla, con columnas para el criterio, su definición operacional y la evidencia de que el caso lo cumple.

Este formato tiene un valor doble: organiza la argumentación de manera que el revisor puede verificar cada criterio independientemente, y obliga al investigador a operacionalizar los criterios; es decir, a especificar qué observable del caso satisface cada uno antes de considerarlos cumplidos.

La Tabla 1 muestra un ejemplo de esta estructura aplicada de forma genérica.

Tabla 1

Estructura genérica de criterios de selección para estudios de caso en ingeniería de software

Criterio de selección	Definición operacional	Satisfecho cuando...
Representatividad estructural	El caso exhibe los componentes mínimos del dominio de población	El sistema tiene los estratos/componentes propios de la clase de sistemas estudiada
Complejidad suficiente	El fenómeno puede manifestarse con la riqueza necesaria	El número de componentes y flujos supera el umbral mínimo para que el análisis sea significativo
Complejidad acotada	El caso es manejable dentro de los recursos del estudio	Un investigador puede completar el análisis en profundidad dentro del cronograma disponible
Documentación verificable	La arquitectura y las decisiones relevantes están especificadas	Existe documentación de diseño a la que el investigador tiene acceso y que cubre los aspectos del fenómeno
Relevancia para el fenómeno	El caso exhibe o puede exhibir el fenómeno estudiado	Las condiciones de diseño hacen posible que el fenómeno ocurra o sea observable
Condiciones declaradas	El estado inicial del caso es conocido	Las decisiones de diseño, supuestos y restricciones están documentados y son consistentes entre sí

Nota. Adaptado de Runeson & Höst (2009), Yin (2018) y Wohlin et al. (2024).

Validez y generalización en estudios de caso

Los cuatro tipos de validez relevantes La validez de un estudio de caso en ingeniería de software se evalúa a través de cuatro dimensiones que Yin (2018) describe como los criterios de calidad del diseño de investigación: validez de constructo, validez interna, validez externa y confiabilidad.

La validez de constructo se refiere a si las medidas o los procedimientos utilizados en el estudio capturan efectivamente los conceptos que el trabajo dice estudiar. En ingeniería de software, esto equivale a preguntar si el proceso seguido, la herramienta aplicada o el artefacto producido corresponden a las definiciones teóricas que el marco del trabajo establece. Si un estudio afirma evaluar la "efectividad del modelado de amenazas" pero mide únicamente el número de amenazas identificadas sin considerar la tasa de falsos positivos ni el tiempo empleado, la validez de constructo es cuestionable.La forma de mitigar este riesgo es definir operacionalmente cada concepto antes de iniciar el trabajo de campo y documentar la trazabilidad entre el concepto y la medida.

La validez interna concierne a la coherencia causal del argumento: si el estudio establece una relación entre X e Y –por ejemplo, entre el uso de prompts estructurados y la calidad del análisis de amenazas–, debe poder argumentarse que esa relación no es un artefacto de terceras variables no controladas. En el estudio de caso, la validez interna se fortalece mediante la triangulación de fuentes –confirmar el mismo hallazgo por vías independientes– y mediante el análisis de explicaciones alternativas, es decir, argumentar por qué los resultados no se explicarían mejor por otro mecanismo.

La validez externa es la dimensión más directamente relacionada con la cuestión de la población y la muestra. Se refiere a si los hallazgos del caso son aplicables más allá del caso específico estudiado, y la respuesta depende de cuán claramente se ha definido la población y cuán bien el caso la representa. Como se ha establecido en secciones anteriores, esta generalización es de tipo analítico: el investigador argumenta que el mecanismo identificado en el caso es el mismo mecanismo que actuaría en cualquier otro caso con las características de la población declarada. La validez externa no equivale a decir que todos los sistemas del dominio producirán exactamente los mismos resultados, sino que el proceso o mecanismo estudiado es aplicable a esa clase de sistemas.

La confiabilidad se refiere a la replicabilidad del estudio: si otro investigador siguiera los mismos procedimientos sobre el mismo caso, ¿llegaría a los mismos hallazgos? En ingeniería de software, la confiabilidad se asegura mediante la documentación detallada del protocolo de estudio –los pasos seguidos, las decisiones tomadas, los instrumentos utilizados– de forma que cualquier investigador pueda repetir el trabajo con los mismos datos. La publicación del material de trabajo completo – diagramas, prompts, respuestas, código– en repositorios públicos es una práctica que fortalece significativamente la confiabilidad.

La generalización analítica: mecanismo y no frecuencia La generalización analítica opera a través del reconocimiento de que un hallazgo en un caso particular es instancia de un mecanismo general, y que ese mecanismo se reproducirá en otros casos donde estén presentes las condiciones que lo hacen posible. Este tipo de generalización está implícito en cómo la ingeniería de software produce y acumula conocimiento: un patrón de diseño documentado a partir de unos pocos sistemas ejemplares se aplica posteriormente a cualquier sistema que presente el problema que el patrón resuelve; una vulnerabilidad identificada en un sistema particular se convierte en un aviso para todos los sistemas con la misma arquitectura.

Por tanto, cuando un investigador escribe la sección de discusión de su estudio de caso, la generalización válida no es "todos los sistemas del dominio tendrán

los mismos resultados numéricos", sino "el mecanismo identificado —en este caso, que los supuestos de diseño inseguros concentran la mayor parte de la superficie de ataque— actuará en sistemas que compartan las características de la población declarada". Esta formulación es metodológicamente honesta porque explicita la condición de aplicabilidad —compartir las características de la población— y no promete una frecuencia estadística que el diseño no puede respaldar. Conviene situar esta posición en relación con la tradición interpretativa de la investigación cualitativa: Lincoln y Guba (1985) proponen el concepto de transferibilidad como alternativa a la generalización, argumentando que la aplicabilidad de un hallazgo a otro contexto depende del grado de similitud entre ambos —denominado fittingness— y que es responsabilidad del lector, no del investigador, determinar si esa similitud existe. En el contexto de la ingeniería de software, la posición de Yin resulta operativamente más adecuada para la escritura académica porque exige al investigador explicitar las condiciones del dominio de población, orientando al lector sobre los sistemas a los que los hallazgos son aplicables; la transferibilidad de Lincoln y Guba, en cambio, delega esa carga interpretativa al receptor, lo que dificulta la evaluación por parte de revisores que esperan declaraciones de alcance verificables en la sección de metodología.

Guía práctica de cinco pasos

A partir del marco teórico revisado, se propone un procedimiento operativo de cinco pasos para declarar y justificar la población y la muestra en la sección de metodología de un trabajo de ingeniería de software basado en estudio de caso.

Paso 1 — Definir el dominio de interés. El investigador debe comenzar respondiendo: ¿a qué clase de sistemas, procesos u organizaciones pretende que sus hallazgos sean aplicables? La respuesta a esta pregunta es la definición de la población, y debe formularse en términos de características observables —tipo de arquitectura, escala, contexto de despliegue, dominio funcional— no en términos de instancias concretas. Una formulación adecuada es del tipo: "sistemas de software multicapa orientados a la web con múltiples roles de usuario y flujos de datos que cruzan al menos un límite de confianza"; una formulación inadecuada sería: "sistemas similares al sistema X que estudiamos". La primera define el dominio por sus características; la segunda lo define circularmente por el caso.

Paso 2 — Documentar los criterios de selección del caso. Una vez definida la población, el investigador debe especificar por qué el caso seleccionado pertenece a ese dominio y es adecuado para el propósito del estudio. Para ello debe documentar al menos los cinco criterios descritos en la sección anterior — representatividad estructural, complejidad suficiente y acotada, documentación verificable, relevancia para el fenómeno y condiciones declaradas—, indicando para cada uno la evidencia observable que lo satisface. Esta documentación es la que convierte la selección del caso —sea intencional, teórica o incluso de conveniencia— en una selección metodológicamente defensible.

Paso 3 — Justificar el tipo de diseño (único o múltiple). El investigador debe argumentar explícitamente si el trabajo utiliza un diseño de caso único o múltiple, y por qué ese diseño es coherente con el propósito del estudio. Si se utiliza un solo caso, la justificación debe indicar cuál de las cinco razones de Yin (2018) aplica: ¿es un caso representativo? ¿es un caso decisivo para una teoría? ¿es un caso revelador de un fenómeno antes inaccesible? Omitir esta justificación equivale a afirmar implícitamente que se estudiaron pocos casos por falta de recursos, lo que debilita la credibilidad del trabajo independientemente de la calidad del análisis. La distinción de Stake (1995) entre caso intrínseco e instrumental es complementaria en este paso: dado que la mayoría de los estudios en ingeniería de software son de naturaleza instrumental —el caso es el medio para iluminar un mecanismo o fenómeno que trasciende ese caso particular—, la justificación del diseño debe incluir no solo por qué se eligió un caso único o múltiple, sino también de qué manera el caso seleccionado es el vehículo adecuado para revelar el mecanismo de interés en el dominio de población declarado.

Paso 4 — Declarar el tipo de generalización. En la sección de metodología —o al inicio de la discusión— el investigador debe declarar explícitamente que la generalización que el trabajo produce es analítica y no estadística, y describir el mecanismo o proceso que se generaliza. Esta declaración tiene el efecto positivo de anticipar y desactivar la crítica más común al estudio de caso: que "un solo caso no es suficiente para generalizar". La respuesta correcta a esa crítica no es agregar más casos, sino aclarar que la generalización opera por un mecanismo diferente y que ese mecanismo está apropiadamente documentado en el trabajo. Una formulación concreta de esta declaración en la sección de metodología podría ser: *"Los hallazgos de este estudio son generalizables de forma analítica, no estadística: se argumenta que el mecanismo identificado—la concentración de superficie de ataque en los supuestos de diseño implícitos— actuará en cualquier sistema que comparta las características de la población declarada, con independencia de su dominio funcional."*

Paso 5 — Delimitar los límites de la generalización. Finalmente, el investigador debe indicar explícitamente las condiciones bajo las cuales los hallazgos no serían aplicables. Estos límites pueden ser de tipo estructural —"los hallazgos no se aplican a sistemas sin límite de confianza explícito"—, de escala —"no se ha estudiado el comportamiento del proceso en sistemas con más de cien componentes"—, o de contexto —"el proceso fue evaluado sobre un sistema sintético; su comportamiento en sistemas heredados con documentación incompleta puede diferir". Declarar los límites de la generalización no debilita el trabajo; al contrario, es un indicador de rigor metodológico que fortalece la credibilidad del investigador ante el revisor especializado. Un ejemplo de delimitación bien articulada sería: *"Los resultados son aplicables a sistemas web multicapa con autenticación de múltiples roles; no se ha evaluado el proceso en sistemas*

de arquitectura monolítica ni en sistemas embebidos con restricciones de tiempo real, donde la naturaleza de la superficie de ataque difiere estructuralmente."

Ejemplos de aplicación por subdisciplina

Seguridad de software y análisis de vulnerabilidades: En estudios que evalúan metodologías de análisis de seguridad —modelado de amenazas, revisión de código, pruebas de penetración—, la población se define habitualmente en términos de la clase de sistemas sobre los que la metodología es aplicable: sistemas web, sistemas embebidos, APIs REST, plataformas de gestión de contenido. El caso seleccionado debe ser representativo de esa clase en sus características de superficie de ataque —número de puntos de entrada, tipos de actores, flujos de datos entre componentes con distinto nivel de confianza. La generalización analítica afirma que el proceso de análisis producirá resultados de calidad comparable en sistemas con características análogas, con independencia del dominio funcional del sistema.

Arquitectura y patrones de diseño: En estudios sobre la aplicación y los efectos de patrones arquitectónicos, la población se define por el tipo de problema que el patrón resuelve —sistemas con alta demanda de escalabilidad, sistemas con acoplamiento entre módulos que deben desacoplarse— y el caso es un sistema que exhibe ese problema. La generalización analítica sostiene que el patrón produce el efecto documentado en cualquier sistema que exhiba el mismo problema en un contexto estructural comparable, lo que permite al investigador no reclamar que "todos los sistemas mejorarán con este patrón", sino que "los sistemas con este perfil de problema mejorarán de esta manera".

Metodologías ágiles y procesos de desarrollo: En estudios sobre la adopción o efectividad de metodologías de proceso —Scrum, Kanban, DevOps—, la población se define en términos de las características del equipo y del contexto organizacional: tamaño del equipo, tipo de producto, modelo de financiamiento, madurez técnica. Aquí el caso de estudio puede ser un equipo o una organización, y la generalización analítica afirma que el mecanismo identificado —por ejemplo, que la implantación de revisiones de código sistemáticas reduce la densidad de defectos— actuará en equipos con características comparables. En este dominio es especialmente frecuente el diseño de casos múltiples, precisamente porque la variabilidad entre equipos es parte central del hallazgo.

Herramientas y automatización: En estudios que evalúan una herramienta de software —un analizador estático, un generador de código, un asistente de inteligencia artificial—, la población es la clase de proyectos o sistemas sobre los que la herramienta es aplicable según su propósito declarado. El caso es un proyecto o sistema concreto, seleccionado porque exhibe los tipos de artefactos o problemas que la herramienta está diseñada para procesar. La generalización analítica afirma que la herramienta producirá resultados de calidad comparable en proyectos con perfil técnico análogo, y la limitación a declarar es el perfil de proyectos en los que la herramienta no se ha probado y donde su comportamiento podría ser distinto.

Errores frecuentes en la declaración de población y muestra

La observación sistemática acumulada por el autor a lo largo de más de quince años de actividad ininterrumpida como investigador universitario, docente de metodología de investigación y director de tesis en programas de maestría y doctorado en más de diez universidades —con más de un centenar de trabajos evaluados en ese período en los niveles de grado, maestría y doctorado, tanto en calidad de revisor como de oponente formal— permite identificar cinco errores recurrentes en la declaración de población y muestra en ingeniería de software, cuya corrección mejoraría significativamente la calidad metodológica percibida por los revisores. Esta preocupación por la calidad en el reporte de los diseños de investigación empírica tiene antecedentes en la propia disciplina: las directrices de Kitchenham et al. (2007) para la realización y reporte de revisiones sistemáticas de literatura en ingeniería de software establecen, precisamente, que la declaración explícita y verificable de los criterios de selección es una condición necesaria para que cualquier proceso de investigación sea metodológicamente válido —un principio que se traslada directamente al estudio de caso cuando el investigador debe justificar la selección de su objeto de estudio ante la comunidad académica. Baltes y Ralph (2022) aportan respaldo empírico sistemático a esta observación: su revisión crítica de las prácticas de muestreo en investigación de ingeniería de software documenta que la confusión entre representatividad estadística y analítica, y el muestreo de conveniencia no fundamentado, son problemas recurrentes en la literatura del área, frecuentemente asociados a debilidades en la declaración de validez externa.

El primer error es la omisión completa de la justificación del caso. El trabajo describe el caso en detalle pero no argumenta por qué ese caso fue seleccionado sobre otros posibles. El revisor no puede evaluar si la selección fue metodológicamente apropiada o simplemente conveniente. El segundo error es la definición circular de la población. El investigador define la población como "sistemas similares al que se estudia" o "casos como el caso X", lo que hace la declaración de población vacía de contenido: si no se puede saber si un sistema pertenece a la población sin compararlo con el caso, la definición no delimita nada. El tercer error es la confusión entre generalización estadística y analítica. El investigador defiende la representatividad del caso con argumentos del tipo "PImage es un sistema web típico" sin especificar típico en qué dimensiones ni respecto de qué universo medido. Esta formulación sugiere generalización estadística sin los datos que la respaldarían. El cuarto error es la omisión de los límites de la generalización. El trabajo concluye que la metodología o herramienta estudiada "es efectiva" sin declarar bajo qué condiciones ese juicio es válido y bajo cuáles no lo sería. Esta omisión da lugar a afirmaciones más amplias de lo que el diseño respalda y es frecuentemente la

base de las observaciones de rechazo por parte de revisores. El quinto error es la justificación del tamaño de la muestra con argumentos cuantitativos. El investigador se disculpa por haber estudiado un solo caso argumentando limitaciones de tiempo o acceso, en lugar de justificar metodológicamente por qué un caso único es suficiente para el propósito del estudio. Este error inadvertidamente señala al revisor que el investigador considera que más casos habrían sido metodológicamente superiores, debilitando la credibilidad del diseño elegido.

CONCLUSIONES

El estudio de caso es un diseño de investigación metodológicamente válido y productivo para la ingeniería de software, siempre que el investigador comprenda y aplique los principios que gobiernan la selección del caso, la declaración de la población y la naturaleza de la generalización que el diseño permite. La aplicación mecánica de los conceptos de población y muestra provenientes de la investigación cuantitativa —sin la reformulación que el contexto de ingeniería requiere— produce secciones de metodología imprecisas que debilitan la credibilidad del trabajo independientemente de la calidad del análisis realizado. El artículo cumplió los cuatro objetivos específicos propuestos.

En relación con OE1, se estableció que el estudio de caso es pertinente cuando el fenómeno no puede separarse de su contexto sin perder información esencial, y que su validez opera por generalización analítica —no estadística—, lo que tiene consecuencias directas sobre cómo se diseña, justifica y reporta el trabajo. En relación con OE2, se demostró que en el contexto de la ingeniería de software la población equivale al dominio de fenómenos al que son extrapolables los hallazgos, y la muestra es el caso seleccionado mediante criterios metodológicos explícitos, reformulación que reemplaza la lógica de representatividad estadística por la de representatividad estructural y teórica. En relación con OE3, se articularon cinco criterios cuya documentación es necesaria para justificar la selección del caso —representatividad estructural, complejidad suficiente y acotada, documentación verificable, relevancia para el fenómeno y condiciones declaradas—, derivados de los marcos canónicos de Runeson y Höst (2009), Yin (2018) y Wohlin et al. (2024). En relación con OE4, se propuso una guía operativa de cinco pasos —definir el dominio, documentar criterios, justificar el diseño, declarar la generalización y delimitar los límites— como procedimiento mínimo aplicable a cualquier investigación con diseño de caso en el área.

La adopción de estas prácticas, además de mejorar la tasa de aceptación en revistas especializadas, contribuye a elevar la calidad del conocimiento acumulado en el área, en la medida en que los hallazgos publicados tienen condiciones de aplicabilidad explícitas que permiten a otros investigadores situarlos correctamente en el mapa de lo que se sabe y lo que queda por saber. Como líneas de investigación futura, se identifican: la validación empírica de la guía de cinco pasos mediante su aplicación a una muestra de secciones de metodología publicadas en revistas del área; el desarrollo de instrumentos de evaluación cuantificables derivados de los cinco criterios de selección; y la extensión del marco a diseños de investigación mixtos en los que la lógica del estudio de caso se combina con componentes de encuesta o experimento controlado. La urgencia práctica de estas recomendaciones está respaldada por la evidencia empírica reciente: la crisis de generalización documentada por Baltes y Ralph (2022) y la prevalencia de estudios mal categorizados señalada por Wohlin y Rainer (2022) confirman que los problemas que esta guía busca corregir son sistémicos y activos en la literatura de ingeniería de software, lo que le confiere una relevancia operativa inmediata y no meramente didáctica.

BIBLIOGRAFÍA

Baltes, S., & Ralph, P. (2022). Sampling in software engineering research:A critical review and guidelines. Empirical Software Engineering, 27, Article 94. https://doi.org/10.1007/s10664-021-10072-8

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The case research strategy in studies of information systems. MIS Quarterly, 11(3), 369–386. https://doi.org/10.2307/248684

Eisenhardt, K. M. (1989). Building theories from case study research. Academy of Management Review, 14(4), 532–550. https://doi.org/10.2307/258557

Flyvbjerg, B. (2006). Five misunderstandings about case-study research. Qualitative Inquiry, 12(2), 219–245. https://doi.org/10.1177/1077800405284363

Glaser, B. G., & Strauss, A. L. (1967). The discovery of grounded theory: Strategies for qualitative research. Aldine.

Kitchenham, B., Charters, S., Dyba, T., Brereton, P., Turner, M., Linkman, S., Jorgensen, M., Mendes, E., & Visaggio, G. (2007). Guidelines for performing systematic literature reviews in software engineering (Technical Report EBSE-2007-01). Keele University and University of Durham.

Lincoln, Y. S., & Guba, E. G. (1985). Naturalistic inquiry. SAGE Publications. Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 14(2), 131–164. https://doi.org/10.1007/s10664-008-9102-8

Runeson, P., Höst, M., Rainer, A., & Regnell, B. (2012). Case study research in software engineering: Guidelines and examples. Wiley. Shaw, M. (2003). Writing good software engineering research papers. Proceedings of the 25th

International Conference on Software Engineering (ICSE 2003), 726–736.https://doi.org/10.1109/ICSE.2003.120

Stake, R. E. (1995). The art of case study research. SAGE Publications. Wohlin, C., & Rainer, A. (2022). Is it a case study? — A critical analysis and guidance. Journal of Systems and Software, 192, Article 111395. https://doi.org/10.1016/j.jss.2022.111395

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2024). Experimentation in software engineering (2nd ed.). Springer. https://doi.org/10.1007/978-3-662-69306-3

Yin, R. K. (2018). Case study research and applications: Design and methods (6th ed.). SAGE Publications.