Universidad Veracruzana

Facultad de Estadística e Informática

Región Xalapa

Licenciatura en ingeniería de software

Machine learning para la automatización de revisiones de código fuente

Protocolo de la revisión sistemática de la literatura para la experiencia educativa de Proyecto Guiado

Presenta:

Martin Emmanuel Cruz Carmona

Director:

Dr. Oscar Alonso Ramírez

Codirector:

Mtro. William Zarate Navarro

Diciembre de 2025

"Lis de Veracruz: Arte, Ciencia, Luz"

Universidad veracruzana

Facultad de estadística e informática
Región Xalapa

Licenciatura en ingeniería de software

Machine learning para la automatización de revisiones de código fuente

Protocolo de la revisión sistemática de la literatura para acreditar proyecto guiado

Presenta:

Martin Emmanuel Cruz Carmona

Director:

Dr. Oscar Alonso Ramírez

Codirector:

Mtro. William Zarate Navarro

índice

1. Introducción

En la última década, el rápido avance de tecnologías como la inteligencia artificial generativa, la realidad extendida (XR) y el Internet de las cosas ha transformado de forma disruptiva la producción de software y su impacto en la sociedad. En particular, la IA generativa ya ha redefinido este proceso al cambiar la forma en que el software se produce actualmente (Abrahão et al., 2025). En este contexto, En los últimos años el aprendizaje automático (ML) se ha convertido en una tecnología revolucionaria con un potencial transformador en diversos ámbitos. Los algoritmos de ML permiten a los sistemas aprender a partir de datos, realizar predicciones fundamentadas y adaptar su comportamiento (Nadimpalli et al., 2024).

Sin embargo, La creciente complejidad del desarrollo de software requiere procesos de revisión de código eficientes y fiables. A medida que las bases de código se amplían, mantener la calidad e identificar errores se convierte en un reto cada vez mayor. Las revisiones de código, aunque exhaustivas, suelen llevar mucho tiempo y son propensas a errores humanos, especialmente cuando se trata de grandes volúmenes de código (Phang & Lee, 2024). Si bien las revisiones de código aportan beneficios sustanciales en términos de calidad y transferencia de conocimiento, también implican una inversión de tiempo y dinero considerable, llegando a registrarse cientos o incluso miles de revisiones mensuales en proyectos de gran escala como Linux o Microsoft Bing (ICSE, 2022).

El objetivo de esta revisión sistemática de la literatura es analizar las técnicas de Machine Learning aplicadas en la actividad de revisión de código fuente. A diferencia de otras revisiones, esta RSL establecerá un panorama general del análisis de las técnicas de Machine Learning utilizadas y las limitaciones documentadas en la literatura, con el fin de ofrecer una visión integral del estado actual de la investigación.

La elección de una RSL cómo metodología se justifica por su capacidad para recopilar, evaluar y sintetizar evidencia de manera rigurosa, transparente y reproducible, lo cual resulta esencial para cumplir el objetivo planteado. Según Kitchenham et al. (2015), este enfoque es ampliamente utilizado en la ingeniería de software basada en evidencia, ya que permite mapear el estado del arte, identificar tendencias y vacíos en la literatura, así como consolidar hallazgos dispersos de manera estructurada.

En los últimos años, han surgido diferentes estudios, entre ellos revisiones sistemáticas, que abordan el uso de Machine Learning para la automatización de las revisiones de código fuente. Por ejemplo, Ben Sghaier (2025) considera las interconexiones

de la actividad de revisión de código y como se utilizan para compartir conocimiento, para esto propone una arquitectura de Deep Learning llamada DISCOREV, la cual incorpora dos modelos entrenados en conjunto en dos tareas sucesivas. Aplica esta arquitectura a pares de revisión de código, específicamente en la generación de comentarios, refinamiento de código y refinamiento de código/estimación de la calidad. El estudio muestra que DISCOREV fue evaluado en tareas de revisión de código utilizando un amplio conjunto de datos multilingüe. Para potenciar su desempeño, se llevó a cabo un preajuste que le permitió adquirir conocimientos previos y ofrecer retroalimentación útil durante el entrenamiento. Los resultados, medidos con métricas como BLEU y CodeBLEU, evidencian que DISCOREV superó a modelos de referencia como T5, CodeT5 y CodeReviewer, alcanzando un puntaje CodeBLEU notablemente superior. Esta ventaja pone de relieve la eficacia de incorporar retroalimentación basada en la estimación de calidad para lograr ediciones de código más precisas y relevantes.

2.Preguntas de investigación

Según Kitchenham et al (2015), las preguntas de investigación constituyen el núcleo de una revisión sistemática, ya que definen el alcance y orientan cada etapa del proceso. En este contexto, las preguntas de investigación han sido diseñadas siguiendo sus directrices propuestas para que la revisión sea enfocada y relevante. Dichas preguntas se muestran en la Tabla 1.

Tabla 1. Preguntas de investigación.

Pregunta de investigación	Tipo de pregunta	Motivación
PI 1.0 ¿Cuáles técnicas de Machine learning se utilizan para automatizar la actividad de revisión de código estático?	Descriptiva	Conocer las técnicas de ML aplicadas a la revisión de código permite identificar los enfoques más utilizados y las tendencias actuales de investigación, sirviendo como base para evaluar su

		efectividad y aplicabilidad en distintos contextos.
PI 2.0 ¿En qué etapas de la revisión de código se aplican las técnicas de machine learning?	Descriptiva	Determinar en qué fases del proceso de revisión se implementan los modelos de ML ayuda a comprender cómo se integran en el flujo de trabajo y qué aspectos del proceso pueden beneficiarse más de la automatización.
PI 3.0 ¿Qué métricas o criterios se usan para evaluar el rendimiento de estos modelos en revisiones de código?	Descriptiva	Identificar las métricas empleadas permite establecer estándares comparativos entre estudios, evaluar la calidad de los modelos y definir parámetros para futuras investigaciones reproducibles.
PI 4.0 ¿Qué beneficios y desafíos se han reportado al integrar ML en el proceso de revisión de código en entornos reales?	Descriptiva	Analizar los beneficios y desafíos documentados aporta una visión equilibrada sobre la viabilidad práctica de estas soluciones y ayuda a reconocer las condiciones necesarias para su implementación efectiva.
PI 5.0 ¿En qué medida la automatización mediante ML reduce el tiempo y esfuerzo humano en revisiones de código?	Descriptiva	Evaluar el impacto de la automatización en la eficiencia del proceso permite medir el valor agregado del uso de

		Machine Learning en esta actividad de la ingeniería de software.
PI 6.0 ¿Cuáles limitaciones técnicas o éticas han identificado los autores en el uso de ML para esta actividad?	Descriptiva	Comprender las limitaciones técnicas y éticas permite anticipar riesgos potenciales, orientar mejoras futuras y promover el uso responsable de la inteligencia artificial en la revisión de código.

3. Estrategia de búsqueda

La revisión sistemática se desarrolla conforme a los lineamientos planteados por Kitchenham et al. (2015), quienes describen diversas técnicas para identificar estudios relevantes en revisiones sistemáticas en ingeniería de software. Para este estudio, se ha optado por una combinación de búsqueda manual y automatizada.

Como paso preliminar, se lleva a cabo una búsqueda manual siguiendo la metodología propuesta por Zhang et al. (2011). Este enfoque tiene como objetivo obtener un primer acercamiento al tema de investigación y sentar las bases para la construcción de un quasi-gold standard (QGS), que consiste en un conjunto de estudios relevantes identificados en las fuentes de publicación específicas y dentro del periodo temporal seleccionado.

La búsqueda manual se realizó seleccionando las fuentes de publicación relevantes en el ámbito de la ingeniería de software, identificadas por su reconocimiento en la comunidad académica, y que tuvieran mayor probabilidad de publicar estudios relevantes que respondieran al menos una pregunta de investigación. En la Tabla 2 se muestran las fuentes de publicación seleccionadas.

Tabla 2. Estudios relevantes identificados

Evento	Nombre	Fuente	Tipo
ACM International Conference Proceeding Series	On the Automation of Code Review Tasks Through Cross-Task Knowledge Distillation	ACM Digital Library	Conferencia
International Conference on Software Engineering	Towards Automating Code Review Activities	ACM Digital Library	Conferencia
International Conference on Software Engineering	Towards Automating Code Review at Scale	ACM Digital Library	Conferencia
Information and Software Technology	Automating modern code review processes with code similarity measurement	ScienceDirect	Artículo de investigación

Posteriormente, se revisaron los estudios publicados entre 2020 y 2025 en cada una de las fuentes de publicación seleccionadas. Para cada estudio, se evaluó inicialmente su título y, si era necesario, su resumen para identificar indicios de relevancia con respecto a las preguntas de investigación; aquellos que mostraban tales indicios se agregaron a un grupo preliminar, conformando un conjunto inicial de 20 estudios. Finalmente, se leyó el texto completo de cada estudio para verificar que efectivamente respondiera al menos a una de las preguntas de investigación.

Tras este proceso, se identificaron 14 estudios relevantes que conformaron el QGS. Para facilitar la trazabilidad y análisis, los estudios del QGS se organizaron en tablas separadas según el motor de búsqueda de las fuentes de publicación de origen. Estas tablas señalan las preguntas de investigación que aborda cada estudio. A continuación, se presenta un resumen del número de estudios por motor de búsqueda:

ACM Digital Library: 4

IEEE Xplore: 5

ScienceDirect: 3

SpringerLink: 2

Tabla 3. Estudios relevantes en ACM Digital Library

ID	Referencia	Pregunta(s) de investigación
ER1	Hellendoorn, V. J., Tsay, J., Mukherjee, M., & Hirzel, M. (2021). Towards automating code review at scale. *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 1479–1482.	PI 1.0, PI 3.0, PI 4.0 y PI 6.0
ER2	Li, Z., Lu, S., Guo, D., Duan, N., Jannu, S., Jenks, G., Majumder, D., Green, J., Svyatkovskiy, A., Fu, S., & Sundaresan, N. (2022). Automating code review activities by large-scale pre-training. *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 1035–1047.	PI 1.0, PI 2.0, PI 3.0, PI 4.0 y PI 6.0
ER3	Yu, Y., Rong, G., Shen, H., Zhang, H., Shao, D., Wang, M., Wei, Z., Xu, Y., & Wang, J. (2024). Fine-Tuning Large Language Models to Improve Accuracy and Comprehensibility of Automated Code Review. *ACM Trans. Softw. Eng. Methodol.*, *34*(1).	PI 1.0, PI 3.0 Y PI 4.0
ER4	Tufano, R., Martin-Lopez, A., Tayeb, A., Dabić, O., Haiduc, S., & Bavota, G. (2025). Deep Learning-based Code Reviews: A Paradigm Shift or a Double-Edged Sword? *Proceedings - International Conference on Software Engineering*, 1640–1652.	PI 3.0 y PI 4.0

Tabla 4. Estudios relevantes de IEEE Xplore

ID	Referencia	Pregunta(s) de investigación
ER5	Moshkin, V., Kalachev, V., & Zarubin, A. (2022). Automation of Program Code Analysis Using Machine Learning Methods. *Proceedings - 2022 International*	PI 1.0, PI 2.0, PI 4.0 y PI 6.0

	Russian Automation Conference, RusAutoCon 2022, 404–408.	
ER6	Cihan, U., Haratian, V., İçöz, A., Gül, M. K., Devran, Ö., Bayendur, E. F., Uçar, B. M., & Tüzün, E. (2024). *Automated Code Review In Practice*.	PI 1.0, PI 3.0, PI 4.0, PI 5.0 y 6.0
ER7	Tufano, R., Masiero, S., Mastropaolo, A., Pascarella, L., Poshyvanyk, D., & Bavota, G. (2022). Using Pre-Trained Models to Boost Code Review Automation. *Proceedings - International Conference on Software Engineering*, *2022-May*, 2291–2302.	PI 1.0, PI 3.0, PI 4.0 Y PI 6.0
ER8	Saini, N., & Britto, R. (2021). Using machine intelligence to prioritise code review requests. *Proceedings of the 43rd International Conference on Software Engineering: Software Engineering in Practice*, 11–20.	PI 1.0, PI 2.0, PI 3.0, PI 4.0, PI 5.0 y PI 6.0
ER9	Tufano, R., Pascarella, L., Tufano, M., Poshyvanyk, D., & Bavota, G. (2021). Towards automating code review activities. *Proceedings - International Conference on Software Engineering*, 163–174.	PI 1.0, PI 3.0 y PI 6.0

Tabla 5. Estudios relevantes de ScienceDirect

ID	Referencia	Pregunta(s) de investigación
ER10	Kartal, Y., Akdeniz, E. K., & Özkan, K. (2024). Automating modern code review processes with code similarity measurement. *Information and Software Technology*, *173*, 107490.	PI 1.0, PI 3.0, PI 4.0, PI 5.0 y PI 6.0
ER11	Islam, K., Ahmed, T., Shahriyar, R., Iqbal, A., & Uddin, G. (2022). Early prediction for merged vs abandoned code	PI 1.0, PI 3.0, PI 4.0, PI 5.0 y PI 6.0

	changes in modern code reviews. *Information and Software Technology*, *142*, 106756.	
ER12	Liao, Z., Zhang, B., Huang, X., Yu, S., & Zhang, Y. (2023). Code Reviewer Intelligent Prediction in Open Source Industrial Software Project. *CMES - Computer Modeling in Engineering and Sciences*, *137*(1), 687–704.	PI 1.0, PI 2.0, PI 3.0, PI 4.0, PI 5.0 y PI 6.0

Tabla 7. Estudios relevantes de SpringerLink.

ID	Referencia	Pregunta(s) de investigación
ER13	Lu, J., Li, X., Hua, Z., Yu, L., Cheng, S., Yang, L., Zhang, F., & Zuo, C. (2025). DeepCRCEval: Revisiting the Evaluation of Code Review Comment Generation. *Lecture Notes in Computer Science*, *15693 LNCS*, 43–64.	PI 1.0, PI 2.0, PI 3.0, PI 4.0, PI 5.0 y PI 6.0
ER14	Lomio, F., Moreschini, S., & Lenarduzzi, V. (2022). A machine and deep learning analysis among SonarQube rules, product, and process metrics for fault prediction. *Empirical Software Engineering 2022 27:7*, *27*(7), 189-. https://doi.org/10.1007/S10664-022-10164-Z	PI 1.0, PI 2.0, PI 3.0, PI 4.0, PI 5.0 y PI 6.0

3.1 Términos de búsqueda.

Para la búsqueda automatizada en bibliotecas digitales y sistemas de indexación, las cadenas de búsqueda se construyeron de manera objetiva, siguiendo una de las propuestas de Zhang et al. (2011), que consiste en elicitar términos clave a partir del QGS. Para ello, se realizó un análisis de frecuencia de palabras en los títulos, resúmenes y palabras clave de los estudios del QGS, utilizando una herramienta en línea de conteo de palabras.

Los términos de búsqueda identificados se encuentran en la tabla 7.

Tabla 7. Términos de búsqueda

Palabra clave	Término en inglés	Homólogo
Revisión de código	Code review	code inspection
Machine learning	Machine learning	Deep learning, artificial intelligence
Análisis de código	Source code analysis	Static code analysis, program analysis
Identificación de errores	Bug detection	Code smells

3.2 Cadenas de búsqueda

Se construyó una cadena búsqueda general mediante la combinación de operadores lógicos y de los términos de búsqueda identificados.

("Code review" OR "Code Inspection" OR "Source code analisys" OR "Static code analisys") AND (("Machine learning" OR "Deep learning" OR Artificial intelligence") OR "Bug detection")

Esta cadena general se utilizó como base para desarrollar cadenas especificas adaptadas a los motores de búsqueda asociados a las fuentes de publicación de QGS. El rendimiento de cada cadena adaptada se evaluó utilizando las métricas de *recall* (sensibilidad) y *precisión* (precisión).

$$Recall = \frac{Estudios\ relevantes\ encontrados}{Total\ de\ estudios\ relevantes}\ x\ 100\%$$

$$Precision = \frac{Estudios\ relevantes\ encontrados}{Total\ de\ estudios\ encontrados}\ x\ 100\%$$

Se adoptó un umbral de *recall* del 80%, basado en las escalas propuestas por Dieste y Padua (2007), lo que permitió decidir si era necesario revisar y adaptar la cadena de búsqueda.

A continuación, se muestra cada una de las cadenas de búsqueda propuestas para cada motor de búsqueda, junto con el resultado de su evaluación.

Tabla 7. Cadenas de búsqueda para ACM Digital Library

ID	Cadena	Estudios encontrados	Estudios relevantes encontrados	Recall	Precisión
CB-1	("Code review" OR "Code inspection") AND ("Machine learning" OR "Deep learning" OR "Artificial intelligence" OR “Bug detection” OR “Code smells”)	1,289	3/4	75%	0.23%
CB-2	("Code review" OR “Code inspection”) AND ("Machine learning" OR "Deep learning") AND ("Code smells" OR "Bug detection")	237	2/4	50%	0.84%
CB-3	("Code review" OR "Code inspection") AND ("Machine learning" OR "Deep learning" OR	1,198	4/4	100%	0.33%

	"Artificial intelligence")				

La tabla 7 muestra la cantidad de estudios recuperados y la proporción de estudios relevantes obtenidos por cada una de las cadenas propuestas para ACM Digital Library. Se optó por la cadena CB-3, ya que registro un nivel de *recall* alto (100%) recuperando todos los estudios relevantes.

Tabla 8. Cadenas de búsqueda para IEEE Xplore

ID	Cadena	Estudios encontrados	Estudios relevantes encontrados	Recall	Precisión
CB-1	("Code review" OR "Code inspection") AND ("Machine learning" OR "Deep learning" OR "Artificial intelligence" OR “Bug detection” OR “Code smells”)	506	4/5	80%	0.79%
CB-2	("Code review" OR “Code inspection”) AND ("Machine learning" OR "Deep learning") AND ("Code smells" OR "Bug detection")	17	1/5	20%	5.88%
CB-3	("Code review" OR "Code inspection") AND ("Machine	306	3/5	60%	0.98%

	learning" OR "Deep learning" OR "Artificial intelligence")				

La tabla 8 muestra la cantidad de estudios recuperados y la proporción de estudios relevantes obtenidos por cada una de las cadenas propuestas para IEEE Xplore. Se optó por la cadena CB-1, ya que registro un nivel de *recall* alto (80%) recuperando todos los estudios relevantes.

Tabla 9. Cadenas de búsqueda para ScienceDirect

ID	Cadena	Estudios encontrados	Estudios relevantes encontrados	Recall	Precisión
CB-1	("Code review" OR "Code inspection") AND ("Machine learning" OR "Deep learning" OR "Artificial intelligence" OR “Bug detection” OR “Code smells”)	830	3/3	100%	0.36%
CB-2	("Code review" OR “Code inspection”) AND ("Machine learning" OR "Deep learning") AND ("Code smells" OR "Bug detection")	125	2/3	40%	1.6%
CB-3	("Code review" OR "Code inspection") AND ("Machine	780	3/3	100%	0.38%

	learning" OR "Deep learning" OR "Artificial intelligence")				

La tabla 9 muestra la cantidad de estudios recuperados y la proporción de estudios relevantes obtenidos por cada una de las cadenas propuestas para ScienceDirect. Se optó por la cadena CB-3, ya que registro un nivel de *recall* alto (100%) recuperando todos los estudios relevantes.

Tabla 10. Cadenas de búsqueda para SpringerLink

ID	Cadena	Estudios encontrados	Estudios relevantes encontrados	Recall	Precisión
CB-1	("Code review" OR "Code inspection") AND ("Machine learning" OR "Deep learning" OR "artificial intelligence" OR “Bug detection” OR “Code smells”)	282	1/2	50%	0.35%
CB-2	("Code review" OR “Code inspection”) AND ("Machine learning" OR "Deep learning") AND ("Code smells" OR "Bug detection")	212	0/2	0%	0%
CB-3	("Code review" OR "code inspection")	1,166	2/2	100%	0.17%

	AND ("Machine learning" OR "Deep learning" OR "Artificial intelligence")				

La tabla 9 muestra la cantidad de estudios recuperados y la proporción de estudios relevantes obtenidos por cada una de las cadenas propuestas para SpringerLink. Se optó por la cadena CB-3, ya que registro un nivel de *recall* alto (100%) recuperando todos los estudios relevantes.

3.3 Selección de fuentes

Siguiendo el ranking de motores de búsqueda más referenciados en la investigación de ingeniería de software propuesto por Zhang et al. (2011), se seleccionaron las bibliotecas digitales ACM Digital Library, SpringerLink, y IEEE Xplore puesto que estas plataformas concentran una gran cantidad de artículos del tema de investigación. Asimismo, se optó por incorporar ScienceDirect, reconocido como otro motor importante y al que se tiene acceso. Además, considerando la naturaleza multidisciplinaria de la investigación.

4. Selección de estudios primarios

4.1 Criterios de selección de estudios primarios.

Los criterios de selección se formulan para determinar si un estudio primario se incluye o se excluye para la revisión sistemática. Estos criterios de inclusión y exclusión se muestran en la Tabla 11 y Tabla 12 respectivamente.

Tabla 11. Criterios de inclusión

Criterios de inclusión	
Criterio	**Justificación**
CI-1: El trabajo fue publicado entre 2020 y 2025.	Dado que el machine learning está estrechamente relacionado a las tecnologías emergentes, es fundamental incluir estudios

	recientes que reflejen el contexto actual de estas tecnologías y su uso en las revisiones de código.
CI-2: El trabajo se encuentra escrito en idioma inglés.	El inglés es el idioma predominante en la investigación académica sobre ingeniería de software y machine learning.
CI-3: El título o resumen del trabajo menciona "Machine Learning" (o términos relacionados) y da indicios de responder al menos a una pregunta de investigación.	Filtra trabajos que aborden explícitamente machine learning y revisiones de código.
CI-4: El texto completo del estudio responde explícitamente al menos a una de las preguntas de investigación	Esencial para construir un análisis teórico sólido sobre el uso de machine learning para la automatización de revisiones de código fuente.

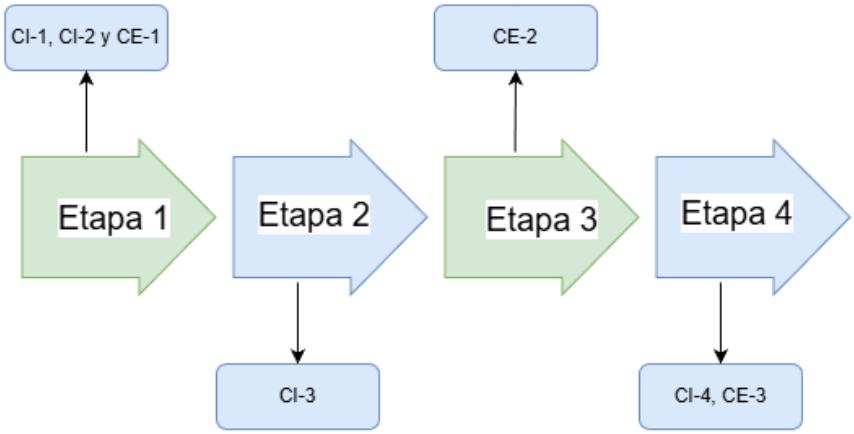
Tabla 12. Criterios de exclusión

Criterios de exclusión	
Criterio	**Justificación**
CE-1: El estudio no está publicado en un congreso o revista.	Los artículos publicados en congresos y revistas, al haber pasado por una revisión por pares, aseguran una metodología sólida, datos válidos y argumentos bien fundamentados. De este modo, se previene información que contenga sesgos.
CE-2: No se tiene acceso al texto completo.	Impide verificar si el artículo responde explícitamente a las preguntas de investigación y extraer datos para el análisis.
CE-3: El estudio se encuentra duplicado (se tomará la versión más reciente).	La versión más actualizada y completa es clave para capturar las últimas perspectivas

	sobre el uso de Machine Learning para la automatización de revisiones de código fuente.

4.2 Procedimientos para la selección de estudios primarios

El procedimiento para la selección de estudios primarios consta de las siguientes etapas, donde cada una está conformada por la aplicación de ciertos criterios de selección. **Ilustración 1.** Procedimiento de selección para estudios primarios.



5. Evaluación de calidad

La evaluación de calidad de los estudios primarios incluidos en la revisión sistemática se realizará para garantizar que los hallazgos sean rigurosos, creíbles y relevantes, siguiendo las recomendaciones de Kitchenham et al. (2015). Para ello, se adaptaron los criterios propuestos por Dybå y Dingsøyr (2008), que han sido ampliamente utilizados en revisiones sistemáticas de ingeniería de software. Los criterios se agrupan en cuatro categorías principales: Selección, Rigor, Credibilidad, y Relevancia, y están diseñados para evaluar la calidad metodológica y la utilidad de los estudios en el contexto de esta revisión sistemática.

Cada estudio será evaluado utilizando un conjunto de 9 criterios que se responden con "Sí" o "No". El primer criterio, relacionado con la naturaleza empírica del estudio, actuará como un umbral de inclusión/exclusión: los estudios que no sean empíricos serán excluidos. Los criterios restantes evaluarán el rigor metodológico, la credibilidad de los hallazgos, y la relevancia para la investigación y la práctica (por ejemplo, aporte al diseño ético de sistemas

persuasivos). Este proceso permitirá identificar estudios de alta calidad que contribuyan de manera confiable a la síntesis de datos.

Los criterios de evaluación se presentan en la tabla 13.

Tabla 13. Criterios de evaluación

Aspecto	Criterio
Selección	1. ¿El artículo se basa en investigación empírica o es simplemente un informe de "lecciones aprendidas" basado en opinión de expertos?
Rigor	2. ¿El artículo tiene una base teórica clara o está vinculado a literatura existente?
	3. ¿El diseño de la investigación es apropiado para abordar los objetivos del estudio?
	4. ¿Se describen adecuadamente los métodos de recolección y análisis de datos?
	5. ¿Se identifican limitaciones o amenazas a la validez del estudio?
Credibilidad	6. ¿El artículo tiene objetivos claros y bien definidos?
	7. ¿El artículo presenta hallazgos claros, creíbles y bien fundamentados?
Relevancia	8. ¿El artículo describe adecuadamente el contexto de la investigación?
	9. ¿El artículo aporta valor a la investigación o práctica en el diseño de sistemas de software persuasivos?

Siguiendo el enfoque de Meckenstock et al. (2024), se establecerá un umbral de calidad del 50% para los criterios 2 a 9: los estudios que obtengan al menos 4 respuestas "Sí" de los 8 criterios, excluyendo el criterio 1, serán considerados de alta calidad y priorizados para la síntesis.

6. Extracción de datos

A continuación, se presenta el formato para recabar la información necesaria de cada estudio primario seleccionado

Tabla 14. Formato de extracción de datos.

Datos generales	
ID	Identificador único para cada estudio primario siguiendo la nomenclatura EP- seguido de un número secuencial.
DOI	Identificador digital que corresponde al estudio publicado.
Titulo	Nombre oficial dado por los autores al trabajo.
Autores	Personas o equipo que elaboraron y publicaron el estudio.
Año	Fecha de publicación del estudio.
Motor de búsqueda	Motor de búsqueda en que encontró el estudio.
Fuente	Nombre del congreso o revista donde el estudio fue publicado o presentado.
Tipo de publicación	Tipo de publicación en el que se presentaron los hallazgos del estudio.
Palabras clave	Términos o frases seleccionados por los autores para describir los conceptos centrales del estudio.
Resumen	Resumen establecido del estudio.
Preguntas de investigación relacionadas	Preguntas de investigación a las cuales el contenido del estudio responde
Técnica de ML propuesta	El **algoritmo**, **arquitectura** o **enfoque** de Machine Learning que reporta el estudio primario. (PI 1.0)
Tarea automatizada/Etapa de aplicación	La **tarea específica** de la revisión de código que se automatiza (ej. generación de comentarios, detección de bugs) o la **etapa del proceso** donde se aplica la técnica. (PI 2.0)

Métrica de evaluación	Los **criterios** o **métricas** que utiliza el estudio para medir el rendimiento y la efectividad del modelo propuesto. (PI 3.0)
Beneficio reportado	Las **ventajas**, **mejoras** o **resultados positivos** que reporta el estudio al integrar la técnica de ML en el proceso. (PI 4.0)
Desafío reportado	Los **problemas**, **desventajas** o **retos** reportados durante la implementación o el uso del enfoque de ML. (PI 4.0)
Impacto en tiempo/esfuerzo	La evidencia **cuantitativa** o **cualitativa** sobre la reducción (o aumento) del tiempo o esfuerzo humano que reporta el estudio. (PI 5.0)
Limitación identificada	Las **limitaciones técnicas o éticas** del enfoque propuesto que son reconocidas por los autores del estudio. (PI 6.0)

Se llevo a cabo una prueba preliminar con 3 estudios del QSG elegidos al azar, cuyos resultados se presentan en el anexo I.

7. Estrategia para la síntesis de datos.

La síntesis de datos de esta revisión se realizó por medio de una síntesis narrativa. La síntesis narrativa constituye un enfoque metodológico en la revisión sistemática de la literatura, permitiéndonos el integrar y analizar de forma cualitativa los hallazgos de múltiples estudios primarios. De acuerdo con Popay et al (2006), esta metodología consiste fundamentalmente en el uso de un relato estructurado, utilizando el lenguaje textual para resumir y explicar los resultados obtenidos.

Popay et al. (2006) propone que la síntesis narrativa se articula en cuatro elementos interrelacionados, cada uno con un propósito específico en el proceso de integración de la evidencia.

El primer elemento es el desarrollo de la teoría del cambio, en esta etapa se formula un modelo conceptual que explica cómo y por qué opera la intervención y para quién resulta

eficaz, orientando la formulación de la pregunta de investigación y la selección de estudios, y facilitando la interpretación y generalización de los hallazgos.

El segundo elemento es la síntesis preliminar, donde se organiza y describe la información extraída de los estudios, identificando patrones en la dirección y magnitud de los efectos o en los facilitadores y barreras, lo que establece una base para el análisis de relaciones. Posteriormente la exploración de relaciones en los datos, donde se establecen vínculos entre los hallazgos agrupándolos en clústers temáticos, utilizando herramientas gráficas y una síntesis textual que resume de manera ordenada las interrelaciones.

El análisis de los hallazgos recopilados en esta revisión se realizó mediante una serie de tabulaciones, siguiendo el enfoque metodológico propuesto por Popay et al. (2006).

8. Limitaciones

LIM-01: El tiempo restringido para realizar la revisión sistemática, estimado en un plazo de dos meses, puede afectar la exhaustividad del análisis y la inclusión de estudios relevantes.

LIM-02: Dada la falta de experiencia previa en el campo de estudio, se aumenta el riesgo de omisiones de estudios importantes o interpretaciones incompletas de estos.

LIM-03: La dependencia de fuentes accesibles puede reducir la cantidad de información analizada y comprometer la profundidad de los resultados.

LIM-04: La exclusión de la literatura en idiomas distintos al inglés, puede omitir investigaciones relevantes publicadas en otras lenguas

9. Informe

La fase final de esta revisión sistemática consiste en documentar o reportar el estudio, siguiendo las pautas de la estructura PRISMA con adaptaciones menores sugeridas por Kitchenham et al. (2015) para orientarla hacia la ingeniería de software, dado que este formato fue diseñado inicialmente para el ámbito médico. La estructura del informe propuesta por Kitchenham et al. (2015) consta de las siguientes partes:

1. **Título:** Identificará el tema del estudio, que es una revisión sistemática de la literatura y que el tema es la ética en el diseño de sistemas de software persuasivos.
2. **Resumen:** Resumen estructurado, incluyendo encabezados para antecedentes, objetivos, métodos, resultados y conclusiones.

3. **Introducción:** Justificación de la revisión y de la(s) pregunta(s) de investigación.
4. Antecedentes: Cualquier información necesaria para comprender el tema de la revisión sistemática.
5. **Método:** Incluirá el acceso al protocolo, reportará el proceso de búsqueda y selección, incluyendo las bases de datos consultadas, los términos de búsqueda y los criterios de inclusión y exclusión.
6. **Resultados:** Reportará el proceso de selección de estudios, incluyendo el número de estudios excluidos en cada etapa principal, destacando las características principales de los estudios primarios identificados, de preferencia con métodos gráficos.
7. **Discusión:** Proporcionará un resumen de la evidencia para cada pregunta de investigación y cualquier análisis adicional. Discutirá las limitaciones de la revisión a nivel de los estudios primarios y a nivel de la revisión misma.
8. **Conclusiones:** Ofrecerá una interpretación general de los resultados en el contexto de cualquier otra evidencia. Proporcionará recomendaciones para investigadores y profesionales
9. **Apéndices:** Reportará las cadenas de búsqueda utilizadas para fuentes digitales individuales. Se incluirá el acceso a la lista de estudios primarios preliminares dividida por biblioteca, la lista estudios primarios eliminados por cada etapa junto al motivo de eliminación de la revisión, lista final de estudios primarios, formulario de extracción de datos y listas de cotejo de calidad.

10. Gestión de la revisión

Para la gestión de la revisión sistemática, se emplearán Mendeley y Excel como herramientas de soporte. Mendeley se utilizará para la gestión bibliográfica, pues es ideal para recopilar, organizar y citar referencias de manera eficiente. Como herramienta de análisis y organización de datos, Excel es útil para tabular y sintetizar la información extraída de los estudios incluidos en la revisión, esto ayuda a mantener un seguimiento ordenado del proceso de selección y análisis de los artículos.

Finalmente, se optó por trabajar con el siguiente cronograma para realizar las actividades con un orden y fechas establecidas.

Actividades	2025											
	Octubre				Noviembre				Diciembre			
	1	2	3	4	1	2	3	4	1	2	3	4
Búsqueda manual y automática												
Selección de estudios												
Evaluación de calidad												
Extracción												
Codificación y síntesis												
Redacción del informe												
Revisión y ajustes												

Tabla 15. Cronograma de actividades para la RSL

Referencias

Libros de consulta

Kitchenham, B. A., Budgen, D., & Brereton, P. (2015). Evidence-based software engineering and systematic reviews (1.ª ed.). Chapman and Hall/CRC. https://doi.org/10.1201/b19467

Publicaciones periódicas

Al-Msallam, S., Xi, N., & Hamari, J. (2023). Unethical Gamification: A Literature Review. Hawaii International Congreso on System Sciences. https://doi.org/10.24251/HICSS.2023.136

Benner, D., Schöbel, S., & Janson, A. (2021). It is only for your own good, or is it? Ethical considerations for designing ethically conscious persuasive information systems. 27th Annual Americas Congreso on Information Systems, AMCIS 2021. Scopus.

Benner, D., Schöbel, S. M., Janson, A., & Leimeister, J. (2022). How to Achieve Ethical Persuasive Design: A Review and Theoretical Propositions for Information Systems. AIS Transactions on HumanComputer Interaction, 14(4), 548-577. https://doi.org/10.17705/1thci.00179

Branch, C. C., Beaton, C. I., McQuaid, M., & Weeden, E. (2021). Perceptions of Ethics in Persuasive User Interfaces. En R. Ali, B. Lugrin, & F. Charles (Eds.), Persuasive Technology (Vol. 12684, pp. 275– 288). Springer International Publishing. https://doi.org/10.1007/978-3-030-79460-6_22

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Information and software technology, 50(9-10), 833-859.

Fogg, B. (2009). A behavior model for persuasive design. Proceedings of the 4th International Congreso on Persuasive Technology, 1–7. https://doi.org/10.1145/1541948.1541999

Gioia, D. A., Corley, K. G., & Hamilton, A. L. (2013). Seeking qualitative rigor in inductive research: Notes on the Gioia methodology. Organizational research methods, 16(1), 15-31.

Maedche, A., Legner, C., Benlian, A., Berger, B., Gimpel, H., Hess, T., Hinz, O., Morana, S., & Söllner, M. (2019). AI-Based Digital Assistants: Opportunities, Threats,
and Research Perspectives. Business & Information Systems Engineering, 61(4), 535–544. https://doi.org/10.1007/s12599-019-00600-8

Meckenstock, J. N. (2024). Shedding light on the dark side–A systematic literature review of the issues in agile software development methodology use. Journal of Systems and Software, 211, 111966.

Orji, R., Oyibo, K., Lomotey, R. K., & Orji, F. A. (2019). Socially-driven persuasive health intervention design: Competition, social comparison, and cooperation. Health Informatics Revista, 25(4), 1451–1484. https://doi.org/10.1177/1460458218766570

Rahman, P., & Adaji, I. (2024). Ethics in Persuasive Technologies: A Systematic Literature Review. Proceedings of the International Congreso on Mobile and Ubiquitous Multimedia (MUM '24), December 01–04, Stockholm, Sweden. ACM. https://doi.org/10.1145/3701571.3701572.

Williams, J. (2018). Stand out of our light: Freedom and resistance in the attention economy. Cambridge University Press.

Zhang, H., Babar, M. A., & Tell, P. (2011). Identifying relevant studies in software engineering. Information and Software Technology, 53(6), 625-637.

Anexo I

Tabla 16. Prueba piloto 1

Datos generales	
ID	EP-1
DOI	https://doi.org/10.1016/j.ins.2024.121753
Titulo	Move method refactoring recommendation based on deep learning and LLM-generated information
Autores	Zhang, Yang; Li, Yanlei; Meredith, Grant; Zheng, Kun; Li, Xiaobin
Año	2025
Motor de búsqueda	ScienceDirect
Fuente	Information Sciences
Tipo de publicación	Revista
Palabras clave	Move method, Refactoring recommendation,

	Deep learning, Large language models, Prompt engineering
Resumen	Move method refactoring is a prevalent technique typically applied when a method relies more on members of other classes than on its original class. Existing approaches for move method refactoring recommendations have improved accuracy based on deep learning. However, it is challenging to capture the deep semantics behind the code and the true intention of the developer. Furthermore, the accuracy of move method refactoring needs to be improved. To alleviate these problems, this paper proposes MoveRec, a move method refactoring recommendation based on deep learning and LLM-generated information. To generate the dataset, MoveRec selects 58 real-world projects from which it extracts metric, textual, and semantic features. Metric features are derived using static analysis tools. Textual features are generated with LLM to obtain code summaries, and the pre-trained model is used to produce word vectors. Semantic features are obtained by calculating the similarity between the original and target classes. Finally, we construct a dataset with 12,475 samples. A deep learning model CNN-LSTM-GRU is designed for refactoring recommendations. We evaluate MoveRec on this dataset and experimental

	results show that the average F1 is 74%. Compared to existing methods including PathMove, JDeodorant, JMove, and RMove, MoveRec improves F1 ranging from 9.4% to 53.4%, demonstrating its effectiveness.
Preguntas de investigación relacionadas	PI 1.0, PI 2.0 y PI 4.0
Técnica de ML propuesta	Specifically, we designed a three-branch CNN-LSTM-GRU model architecture where the CNN branch processed metric features, the LSTM branch handles textual features, and the GRU branch deals with semantic features.
Tarea automatizada/Etapa de aplicación	To alleviate these problems, this paper proposes MoveRec, a move method refactoring recommendation based on deep learning and LLM-generated information.
Métrica de evaluación	
Beneficio reportado	Compared to existing methods including PathMove, JDeodorant, JMove, and RMove, MoveRec improves F1 ranging from 9.4% to 53.4%, demonstrating its effectiveness.
Desafío reportado	
Impacto en tiempo/esfuerzo	
Limitación identificada	

Tabla 17. Prueba piloto 2

Datos generales	
ID	EP-2
DOI	10.1109/ISAI-NLP64410.2024.10799479

Titulo	Automated Code Review and Bug Detection
Autores	Xuan, Phang Justin Sheng; Lee, Yunli
Año	2024
Motor de búsqueda	IEEE Xplore
Fuente	19th International Joint Symposium on Artificial Intelligence and Natural Language Processing, iSAI-NLP 2024
Tipo de publicación	Congreso
Palabras clave	Machine Learning, Code Review, Bug Detection, Automated
Resumen	With the increasing demand for efficient code reviews, especially among beginner programmers, existing tools often lack user-friendliness and fail to streamline error detection and correction effectively. To address these gaps, this paper presents an Automated Code Review and Bug Detection Tool designed to simplify the identification and fixing of code errors. The tool features an intuitive graphical user interface (GUI) tailored for both novice and experienced programmers. Leveraging advanced machine learning models, it classifies errors and suggests code refinements, ensuring accurate and efficient error detection. The development process involved integrating machine learning algorithms for error classification within a streamlined GUI. Usability and effectiveness were assessed through feedback from 21 participants, highlighting

	the tool's capability to enhance programming accessibility and simplify the review process. The successful implementation demonstrates the tool's potential to provide a user-friendly solution, contributing to more efficient programming practices and making code review accessible to a broader audience.
Preguntas de investigación relacionadas	PI 1.0, PI 3.0 y PI 4.0
Técnica de ML propuesta	CodeBERT: A bimodal pre-trained model based on the ROBERTa architecture, fine-tuned for error classification using a hybrid objective function combining masked language modeling (MLM) and replaced token detection. Refined CodeT5: A model based on the T5 architecture, fine-tuned for code sequence-to-sequence tasks, particularly code generation and refinement.
Tarea automatizada/Etapa de aplicación	
Métrica de evaluación	The model's performance was evaluated using BLEU, ROUGE, and METEOR metrics on the validation set, and the best model is saved for later use." *Y también:* "The classification model was tested on a dataset of 30 good and 30 bad code snippets. Table 2 shows the confusion matrix for the proposed classification model.
Beneficio reportado	While effective for experienced programmers, the CLI's complexity proved challenging for new programmers, the primary target audience, who preferred a

	more user-friendly interface and concise error feedback.
Desafío reportado	
Impacto en tiempo/esfuerzo	
Limitación identificada	

Tabla 18. Prueba piloto 3

Datos generales	
ID	EP-3
DOI	10.1109/RUSAUTOCON54946.2022.9896360
Titulo	Automation of Program Code Analysis Using Machine Learning Methods
Autores	Moshkin, Vadim; Kalachev, Vladislav; Zarubin, Anton
Año	2022
Motor de búsqueda	IEEE Xplore
Fuente	Proceedings - 2022 International Russian Automation Conference, RusAutoCon 2022
Tipo de publicación	Congreso
Palabras clave	Machine Learning, Code Review, Bug Detection, Automated
Resumen	The paper presents a description of the developed approach and service for analyzing source code in Python. The service allows you to reduce the time spent on the code review process by partially automating this process. After publishing a changeset to a remote Git repository, the service starts checking the changeset and publishes reports of bugs and duplicates found in changeset comment format. A neural network with an original architecture is used for automatic code reviews. PyTorch,

	Scikit-learn and Transformers libraries are also used. The testing performed did not reveal any errors that affect the operation. All the main functions of the system are performed correctly.
Preguntas de investigación relacionadas	PI 1.0, PI 2.0 y PI 4.0
Técnica de ML propuesta	The FastText algorithm [9] was used to obtain vector representations of source code texts. A pre-trained neural network language model based on the transformer architecture was used to obtain a possible assignment of a function in natural language [10]. A classifier based on the gradient boosting algorithm was used to detect duplicate PRs [11]
Tarea automatizada/Etapa de aplicación	The service allows you to reduce the time spent on the code review process by partially automating this process. After publishing a changeset to a remote Git repository, the service starts checking the changeset and publishes reports of bugs and duplicates found in changeset comment format.
Métrica de evaluación	
Beneficio reportado	The main goal of the project is to reduce the time for reviewing and assessing the quality of the published set of changes and reduce the impact of the human factor (carelessness, bias, incompetence, etc.).
Desafío reportado	Currently, there are no approaches that simultaneously detect various types of errors and duplicates.
Impacto en tiempo/esfuerzo	
Limitación identificada	

“Lis de Veracruz: Arte, Ciencia, Luz”

www.uv.mx