PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE ESMERALDAS



CARRERA:
INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

TESIS DE GRADO
ESTUDIO DE MAPEO SISTEMÁTICO DE HERRAMIENTAS PARA LA MINERÍA DE REPOSITORIOS DE SOFTWARE.

LÍNEA DE INVESTIGACIÓN:
PROGRAMACIÓN Y DESARROLLO DE SOFTWARE

AUTOR:
CAICEDO ESTACIO CLAUDIA STEPHANIE

ASESOR:
JAIME SAYAGO HEREDIA

ESMERALDAS, 2023

TRIBUNAL DE GRADUACIÓN

Trabajo de tesis aprobado luego de haber dado cumplimientos a los requisitos exigidos por el Reglamento de Grado de la PUCESE, precio a la obtención del título de Ingeniería en Sistemas y Computación.

Presidente del Tribunal de Graduación **f.**_____

Mgt. Jaime Sayago Heredia **f.**_____

Asesor

Mgt. José Luis Carvajal **f.**_____

Lector #1

Mgt. Homero Velastegui **f.**_____

Lector #2

Mgt. Homero Velastegui **f.**_____

Coordinadora de carrera

AUTORÍA DE TESIS

Yo, **CAICEDO ESTACIO CLAUDIA STEPHANIE** con número de cédula de identidad 0802874685 manifiesto que mediante la presente investigación sobre el tema "**ESTUDIO DE MAPEO SISTEMÁTICO DE HERRAMIENTAS PARA LA MINERÍA DE REPOSITORIOS DE SOFTWARE**" los resultados obtenidos como tesis de grado, previo a la obtención del título de "**INGENIERO EN SISTEMAS Y COMPUTACIÓN**" son de total responsabilidad del autor, y que se ha respetado las fuentes de información consultadas, realizando las citas correspondientes y los resultados alcanzados son totalmente personales, únicos y legítimos. Al mismo tiempo declaro que todo el contenido incluyendo resultados, discusión, conclusiones, recomendaciones y otros efectos legales y académicos que se desglosan, son y serán exclusiva responsabilidad legal y académica del autor y de la PUCESE.

_____

Caicedo Estacio Claudia Stephanie

C.I. 0802874685

DEDICATORIA

Dedico este trabajo de investigación con mucho cariño a mi madre, ya que sin su amor y apoyo no hubiese logrado esta meta. A mi familia por siempre estar ahí para mí y alegrarme en los días más difíciles, muchas veces sin saberlo. Y sobre todo quisiera dedicar este trabajo a mi mamita Clarita, por siempre ver la luz en mí y apoyarme en este camino, espero que donde se encuentre este logro le haga sentir mucho orgullo.

AGRADECIMIENTO

Mi agradecimiento está dedicado a toda mi familia, ya que, sin su cariño y apoyo incondicional a lo largo de mi carrera, no hubiese conseguido ese gran objetivo.

De igual forma, quisiera agradecerle a mi pareja y amigos, por el ánimo, por las risas y sobre todo por la paciencia.

Y, por último, pero no menos importante quisiera agradecerles a mis profesores, compañeros y a todos quienes conforman la escuela de Sistemas de la PUCE-Esmeraldas. Ya que sus enseñanzas, consejos y motivaciones en el transcurso de mis estudios son los que permiten que hoy pueda llamarme profesional.

ÍNDICE DE CONTENIDO

TRIBUNAL DE GRADUACIÓN i
DEDICATORIA iii
AGRADECIMIENTO iv
ÍNDICE DE TABLAS vii
ÍNDICE DE FIGURAS vii
Resumen viii
ABSTRACT ix
Introducción 1
CAPÍTULO I 4
1. Marco teórico 4
1.1. Antecedentes 4
1.2. Bases teóricas-científicas 6
1.2.1. Software 6
1.2.2. Ingeniería de Software 6
1.2.3. Calidad del software 8
1.2.4. Sistema de control de versiones 8
1.2.5. Repositorios de software 11
1.2.6. Minería de repositorios de software 12
1.2.7. Estudios de mapeo sistemático 14
1.3. Marco Legal 17
CAPÍTULO II 18
2. Metodología 18
2.1. Población y muestra del estudio 18
2.2. Tipo de investigación 18
2.3. Métodos y técnicas 18
2.4. Población y muestra de estudio 19
2.5. Variables de investigación 19
2.6. Técnicas e instrumentos de recolección de datos 21
Descripción del proceso del estudio de mapeo sistemático 22
2.6.1. Definición de las preguntas de investigación 23
2.6.2. Fuentes de datos y estrategia de búsqueda 23

2.6.3. Selección de estudios 24
2.6.4. Proceso de clasificación 24
2.7. Técnicas de procesamiento y análisis de datos 25
2.8. Normas éticas 26
CAPÍTULO III 27
3. Resultados 27
3.1. PI2: Clasificación de las herramientas para MSR 35
3.2. PI1: Herramientas para la minería de repositorios de software que se han desarrollado durante los últimos años 42
3.3. PI3: Identificación de las herramientas para MSR más utilizadas en la industria del desarrollo de software 44
3.4. Medición del alcance de la minería de repositorios de software 45
CAPITULO IV 47
4. Discusión 47
CAPÍTULO V 49
5. Conclusiones y recomendaciones 49
Conclusiones 49
Recomendaciones 50
6. Referencias 51
6.1. Referencias bibliográficas 51
6.2. Anexos 56

ÍNDICE DE TABLAS

Tabla 1. Principales variables e indicadores sujetos a estudio 19
Tabla 2. Preguntas de investigación..... 23
Tabla 3. Estudios analizados para el mapeo sistemático de herramientas para la minería de repositorios de Software durante el periodo 2015-2021..... 27
Tabla 4. Estudios primarios clasificados según enfoque de la MSR**¡Error! Marcador no definido.**
Tabla 5. Clasificación de las herramientas según su enfoque 36
Tabla 6. Clasificación de las herramientas según sus datos de entrada..... 41
Tabla 7. Países con mayor número de estudios primarios en la investigación..... 45

ÍNDICE DE FIGURAS

Figura 1. Diagrama de Control de versiones local [19]..... 9
Figura 2. Diagrama de Control de versiones centralizado [19]..... 10
Figura 3. Diagrama de Control de versiones distribuido [19]..... 11
Figura 4. Pasos para un mapeo sistemático según la metodología de Petersen [29]..... 21
Figura 5. Proceso de selección de artículos..... 27
Figura 6. Distribución porcentual de los artículos según su enfoque de la MSR..**¡Error! Marcador no definido.**
Figura 7. Distribución porcentual de los artículos según sus datos de entrada. 39
Figura 8. Artículos publicados en los últimos seis años..... 45

Resumen

Año tras año, los investigadores de IS realizan estudios de nuevas técnicas y metodologías para el desarrollo y mantenimiento de software fiable y eficiente. Y hoy en día han encontrado en los repositorios de software un objeto de investigación debido a que la información que almacenan de los softwares los convierte en una fuente rica de información valiosa, la cual puede ser analizada utilizando diferentes técnicas.

Con base en lo anteriormente expuesto, nacen campos de estudio como La Minería de Repositorios de Software o MSR [*Mining Software Repositories* por sus siglas en inglés], que es un área dentro de la Ingeniería de Software cuyo objetivo es descubrir información interesante y útil acerca del desarrollo de software, extrayendo y analizando los datos disponibles de diferentes repositorios, con la finalidad de que los resultados obtenidos ayuden al mejoramiento de las técnicas utilizadas para crear nuevo software.

En esta investigación se describe la investigación de estudios recientes de las principales herramientas utilizadas en la minería de repositorios de software aplicando la técnica de mapeo sistemático para conocer la evolución de esta área en la ingeniería de software, sus enfoques y objetos de análisis. Los resultados del estudio permitieron conocer el panorama de la ingeniería de software, mostrando la gran cantidad que existen y los múltiples enfoques que ellas tienen, como la predicción de errores, la predicción de cambios, la búsqueda de código, entre otros.

ABSTRACT

Year after year, IS researchers conduct studies on new techniques and methodologies for the development and maintenance of reliable and efficient software. Today, they have found software repositories to be an object of research because the information they store about software makes them a rich source of valuable information, which can be analyzed using different techniques.

Based on the above, fields of study like Mining Software Repositories (MSR) have emerged. This is an area within Software Engineering whose aim is to discover interesting and useful information about software development by extracting and analyzing available data from different repositories. The objective is that the findings will aid in improving the techniques used for creating new software.

In this research, we describe the investigation of recent studies on the main tools used in software repository mining by applying the technique of systematic mapping to understand the evolution of this area in software engineering, its approaches, and objects of analysis. The results of the study allowed us to understand the landscape of software engineering, showing the large number of tools that exist and the multiple approaches they have, such as error prediction, change prediction, code searching, among others.

Introducción

Presentación de la investigacion

Hoy en día, el software es una de las tecnologías más importantes en la escena mundial. Hace algunas décadas, nadie hubiera podido predecir que el software se convertiría en una tecnología indispensable para los negocios, ciencias e ingeniería, ni que permitiría la creación de tecnologías nuevas, la ampliación de tecnologías ya existentes y el cambio radical de tecnologías antiguas.

Conforme la industria del desarrollo de software crecía, los programadores se vieron en la tarea de desarrollar nuevas técnicas y metodologías que faciliten la creación de software de forma más ágil, barato y de alta calidad [1], y es aquí donde nace la llamada Ingeniería de Software (IS), la cual es una disciplina que estudia el desarrollo, operación y mantenimiento del software [2], es decir que abarca cada una de las etapas en el ciclo de vida de un software.

Actualmente una de las prácticas más populares y estandarizadas en el desarrollo de software es el uso de repositorios de software, los cuales son servidores capaces de almacenar el código fuente de aplicaciones y conservar la historia de las modificaciones añadidas a cada archivo [3]. Gracias a su uso se ha logrado la creación de grandes proyectos ya que contribuyen al trabajo colaborativo, a la preservación de estos y a la recuperación de información.

Planteamiento del problema

Anualmente, los investigadores de IS realizan estudios de nuevas técnicas y metodologías para el desarrollo y mantenimiento de software fiable y eficiente. Y hoy en día han encontrado en los repositorios de software un objeto de investigación debido a que la información que almacenan de los softwares los convierte en una fuente rica de información valiosa, la cual puede ser analizada utilizando diferentes técnicas.

Con base en lo anteriormente expuesto, nacen campos de estudio como La Minería de Repositorios de Software o MSR [*Mining Software Repositories* por sus siglas en inglés], que es un área dentro de la Ingeniería de Software cuyo objetivo es descubrir información

interesante y útil acerca del desarrollo de software, extrayendo y analizando los datos disponibles de diferentes repositorios, con la finalidad de que los resultados obtenidos ayuden al mejoramiento de las técnicas utilizadas para crear nuevo software[4].

Entre los avances obtenidos en el MSR están el permitir analizar la calidad del software codificado, evitar el llamado *code smell,* predecir y detectar errores, mejorar las habilidades de los desarrolladores, optimizar aplicaciones y apoyar al mantenimiento del software [5].

Este campo de investigación inicialmente surgió al aplicar técnicas de minería de datos en repositorios de software, siendo en un principio bastante limitado, dado que no existían muchos proyectos disponibles en los cuales realizar los estudios[4], no fue hasta la década de los 80 con el nacimiento del software libre que la minería de repositorio de software pudo desarrollarse.

En el 2004, en vista de la gran cantidad de artículos y estudios de MSR publicados en las ediciones pasadas de diferentes conferencias de IS, se realizó por primera vez la Conferencia Internacional del Minería de Repositorios de Software [*MSRconf*][4], la cual se realizó por primera vez en la ciudad de Edimburgo, Escocia.

La afluencia de publicaciones realizadas durante los últimos años, son un indicativo de que pueden ser muchas las técnicas y herramientas para la minería de repositorios de software que se han desarrollado, pero así mismo, son pocos los estudios que muestren el estado actual de la evolución de esta área.

La problemática de esta investigación radica en lograr conocer los avances que ha tenido el campo de la MSR en los últimos años. Para ello se propone una revisión bibliográfica llamado estudio de mapeo sistemático, el cual se utiliza para conocer los panoramas de tecnologías o campos de la investigación, además se han planteado las siguientes interrogantes: (i) ¿Qué herramientas para la minería de repositorios de software se han desarrollado durante los últimos años? (ii) Qué categorías existen para clasificar a las herramientas de MSR existen? (iii) ¿Cuáles son las herramientas para MSR más utilizadas y con mayor aplicabilidad en la industria del desarrollo de software?

Justificación

Investigadores de IS, empresas de desarrollo de software y programadores en general, buscan día a día, soluciones que mejoren el desarrollo y mantenimiento del software, y a pesar de la existencia de varias técnicas y metodologías que faciliten esta labor, las exigencias del mercado varían constantemente por lo que se requiere estar a la vanguardia de los procesos actuales en el campo de la IS.

El presente proyecto de investigación, además de ofrecer una mirada integral sobre la evolución de la minería de repositorios de software, pretende investigar, comparar y evaluar las últimas técnicas, herramientas y métodos para MSR que han sido desarrollados durante los últimos siete años (2015 - 2022) y así dar a conocer cuáles son las soluciones que brinda la MSR para mejorar el desarrollo y mantenimiento de software.

Por otro lado, para muchos investigadores y desarrolladores que conocen la MSR, esta investigación les puede ayudar a conocer cuáles son las herramientas más actuales y en qué estado se encuentran los estudios en esta área. Y para aquellos que desean conocer o implementar por primera vez herramientas para la minería de repositorios de software este trabajo puede ser tomado como un punto de partida.

Objetivos

Objetivo General

Investigar estudios recientes de las principales herramientas utilizadas en la minería de repositorios de software aplicando la técnica de mapeo sistemático para conocer la evolución de esta área en la ingeniería de software.

Objetivos Específicos

a) Identificar las herramientas y tecnologías utilizadas en la minería de repositorios de software.

b) Clasificar las herramientas más utilizadas para la minería de repositorios de software.

c) Medir el alcance de la minería de repositorios de software tomando como base los avances tecnológicos en este campo.

CAPÍTULO I

1. Marco teórico

1.1. Antecedentes

La gran cantidad de estudios realizados y publicados anualmente acerca de la minería de repositorios de software ha desembocado en que varios autores deseen realizar análisis sobre los avances expuestos en estas publicaciones, tal es el caso de:

En el año 2015, Amann y otros publicaron "*Software mining studies: goals, approaches, artifacts, and replicability*"[6] , una revisión bibliográfica que tenía como objetivo analizar propósitos, alcance y replicabilidad de las investigaciones referentes a la minería de repositorios de software con la finalidad de poder observar cuales serían las tendencias futuras en este campo de investigación. En este estudio se evaluaron 297 papers publicados a lo largo de 10 años en las conferencias de MSR, y entre sus resultados encontraron que: solo el 60% de las investigaciones disponibles eran replicables, en algunos casos debido a la disponibilidad de los datos y en otros por la falta de información sobre su replicación; también se observó la curva de popularidad de diferentes tecnologías a lo largo de esa década se identificaron 3 temas que posiblemente serían tendencia en las investigaciones futuras: la minería verde, la ingeniería de software para dispositivos móviles y los aspectos humanos que influyen en el desarrollo de software. En esta investigación se concluye que a medida que los datos de los proyectos de software sigan creciendo rápidamente, la plétora de estudios de minería crecerá junto con el potencial de obtener más y mejores conocimientos sobre aspectos del desarrollo más productivo de software.

Ese mismo año, se publica "*A survey on the use of topic models when mining software repositories*" [7], una investigación que habla acerca del uso de técnicas de recuperación de información, específicamente los *topic models*, en la minería de repositorios de software para poder analizar los datos no estructurados, los cuales son difíciles de analizar con otras técnicas y representan a la mayoría de los datos creados a lo largo del desarrollo del software. Para ello se valieron de una revisión bibliográfica en la cual analizaron 167 artículos de la literatura de ingeniería de software que hacen uso de *topic models*. Gracias a este estudio los autores pudieron concluir que: la mayoría de los estudios se centran en

un número limitado de tareas de ingeniería de software, muchas de las investigaciones utilizan únicamente *topic models* básicos y pudieron notar que los investigadores no aprovechaban del todo el uso de esta técnica.

Mas adelante, en el 2016, De Farias y otros publicaron "*A systematic mapping study on mining software repositories*" [4], en donde presentaron un mapeo sistemática realizado en 107 trabajos de la MSRconf, divulgados entre el 2010 y el 2014. Este estudio tenía como finalidad clasificar las herramientas o técnicas encontradas para Minería de repositorios de software según su enfoque, objetivos, métodos de evaluación y tipos de datos utilizados. Los resultados de este estudio evidenciaron los distintos enfoques y las distintas problemáticas que se intentan resolver con el uso de MSR, siendo las más recurrentes: la comprensión de los defectos, el análisis de las contribuciones y el comportamiento de los desarrolladores. Concluyendo que a pesar de la gran variedad de estudios de MSR, todavía hay muchas oportunidades de investigación que deben explorarse.

Por otro lado en el año 2017, se publicó "*A systematic mapping study of software development with GitHub*" [8], el objetivo principal de este artículo es identificar la cantidad, el tema y los métodos empíricos de los trabajos de investigación, apuntando al análisis de cómo las prácticas de desarrollo de software se ven influenciadas por el uso de una plataforma de codificación social distribuida como GitHub. Se realizó un estudio de mapeo sistemático con cuatro preguntas de investigación y se evaluaron 80 publicaciones de 2009 a 2016, y en los resultados se mostró que la mayoría de los trabajos se centraron en la interacción en torno a tareas relacionadas con la codificación y comunidades de proyectos. También se identificaron algunas preocupaciones sobre cuán confiables eran estos resultados basados en el hecho de que, en general, los artículos usaban conjuntos de datos pequeños y técnicas de muestreo deficientes.

La investigación "*Data mining tools and techniques for mining software repositories: a systematic review*" [9], que fue divulgada en el 2018 muestra los problemas comunes que existe en el desarrollo de software, las consecuencias que pueden causar estos problemas y cómo la minería de repositorios de software puede ser una solución útil a estos problemas. Aquí se realizó una comparativa entre herramientas utilizando una revisión sistemática en la cual se demostró la importancia de la minería de repositorios de software en una era en donde todo está computarizado y el software es un producto importante.

Este documento buscaba también ser una guía para que las personas pudieran comprender el funcionamiento básico de MSR.

Los trabajos antes descritos muestran la evolución y la importancia que ha tenido la minería de repositorios de software en el área de la investigación. A través de mapeos sistemáticos y revisiones bibliográficas, los investigadores han buscado evidenciar los diferentes factores, técnicas y enfoques en los estudios de MSR, además de revelar los avances y tendencias de este campo de estudio. Esto demuestra la importancia de los estudios secundarios dentro de los campos de investigación y cómo pueden servir de guía a nuevas propuestas.

1.2. Bases teóricas-científicas

1.2.1. Software

El termino software se puede definir como el conjunto de programas, instrucciones y reglas informáticas que se ejecutan con la finalidad de realizar una tarea en un computador [10]. También se puede decir que se refiere a la parte intangible que hay en un computador o dispositivo, lo cual incluye tanto a las secuencias de instrucciones que se ejecutan como a los datos que este puede almacenar y manipular [2].

1.2.2. Ingeniería de Software

La ingeniería de software fue definida en 1993 por el Instituto de Ingeniería Eléctrica y Electrónica [conocido por sus siglas como *IEEE*], como la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software. La ingeniería del software ofrece métodos o técnicas para desarrollar y mantener software de calidad que resuelven problemas de todo tipo, y trata áreas muy diversas de la informática y de las ciencias computacionales [11].

En otras palabras también se la puede definir como una disciplina que comprende todos los aspectos de la producción de software, desde las etapas iniciales hasta el mantenimiento de este después de que se lo usa [12].

Historia de la Ingeniería de Software

El término “Software” fue popularizado en 1958 por John Turkey, quien lo usó en un artículo para el periódico matemático American Mathematical Monthly. En sus inicios, el software no tenía tanta relevancia como el hardware, ya que era un producto que se incluía de forma gratuita al comprar un equipo [12]. En las empresas no existía una planificación, ni una estimación de recursos precisa para el desarrollo del software, y no había muchas compañías que se dedicaran a desarrollar software a medida.

Existen varias hipótesis acerca del origen del término “Ingeniería de Software”, pero todo apunta que fue utilizado por primera vez en algún punto entre 1963 y 1964, por Margaret Hamilton, quien fue responsable del desarrollo del software de vuelo para el programa espacial Apolo. Ella lo utilizaba para poder hacer distinción entre su trabajo y el de la ingeniería de hardware [13]. Pero no fue hasta 1968, que este término se popularizó después de una reunión de la OTAN [Organización del Tratado del Atlántico Norte] donde se discutía la relevancia económica y social que estaba adquiriendo el software.

El software ya había dejado de ser sólo una pequeña parte que le daba vida a una computadora, para convertirse en un producto cada vez más costoso e importante. En esta conferencia se pudo detectar la denominada “Crisis del software”, la cual se debía a la introducción de nuevas computadoras de hardware basado en circuitos integrados, lo cual exigía que el software sea más complejo [12], [13]. El escribir software se había vuelto una tarea más difícil y continuamente se buscaba que este sea entendible y verificable. Todos estos cambios provocaron el encarecimiento en los costos de los proyectos, la baja calidad de estos y retrasos en los plazos de entrega, entre otras cosas. Desde ese entonces se empezaron a crear metodologías que permitieran crear software de forma ágil, costeable y de calidad, siendo el inicio de lo que hoy se conoce como ingeniería de software.

En 1987, Fred Brooks publicó su famoso artículo “*No silver bullet*” [14] que traducido literalmente al español significa “No hay balas de plata”. En él compara a los problemas que surgen en el desarrollo de software con monstruos de cuentos de hadas y explica que no existe una bala de plata que pueda acabar con ellos, es decir no hay una solución definitiva para los problemas en la Ingeniería de software. Hoy en día, este artículo sigue estando vigente, ya que los avances tecnológicos han permitido la creación de

mejores metodologías y herramientas para el desarrollo del software, pero de igual manera han traído nuevos retos para los programadores.

1.2.3. Calidad del software

En ingeniería de software, cuando se habla de calidad, se la puede definir como el grado de satisfacción con el que este producto cumple con los requerimientos de los usuarios [15]. Según Pressman la calidad del software es un proceso eficaz que se utiliza para crear un producto útil que brinde un valor medible tanto a quienes lo producen como a quienes lo utilicen [1], denotando que la calidad del software no solo depende de la satisfacción que obtengan los usuarios sino también la que obtengan sus desarrolladores.

Cuando se busca la calidad en un servicio o producto como lo es el software, se debe conocer cuáles son las necesidades del cliente, analizar a la competencia [16] y además implementar un modelo de calidad que permita la gestión de los diferentes puntos en el proceso de construcción del software [17]. Existen varios modelos de calidad para el desarrollo de software y en esta investigación se tomó como referencia a la norma internacional ISO/IEC 25010:2011, la cual define un modelo de calidad que consta de ocho características, las cuales se toman en cuenta la hora de evaluar las propiedades del software: Adecuación Funcional, Fiabilidad, Eficiencia de desempeño, Usabilidad, Mantenibilidad, Seguridad, Compatibilidad y Portabilidad [15]. La medición de estos parámetros ayuda a la categorización de la calidad del software.

1.2.4. Sistema de control de versiones

Un sistema de control de versiones [o VCS, por sus siglas en inglés] es un sistema que gestiona el desarrollo de un objeto en evolución [18]. En otras palabras, es una herramienta que permite guardar y administrar los cambios realizados en ficheros o conjuntos de ficheros a lo largo del tiempo [19], de modo que sea posible recuperar versiones específicas cuando sea requerido.

En el desarrollo de software, los Sistemas de control de versiones facilitan el trabajo colaborativo y permiten observar los cambios realizados en el código, incluyendo las modificaciones línea por línea e integrando información sobre quién realizó el cambio y en qué momento [20].

Hoy en día, existen muchos VCS, los cuales se encuentran clasificados en tres categorías: Locales, centralizados y distribuidos.

Sistemas de control de versiones local

Se trata de un sistema de control básico en el cual se copian los archivos en otro directorio del ordenador o dentro de una base de datos, indicando la fecha de modificación [20]. Este método es muy común por su facilidad de implementación, pero también es tremendamente propenso a errores y pérdidas de información [19].

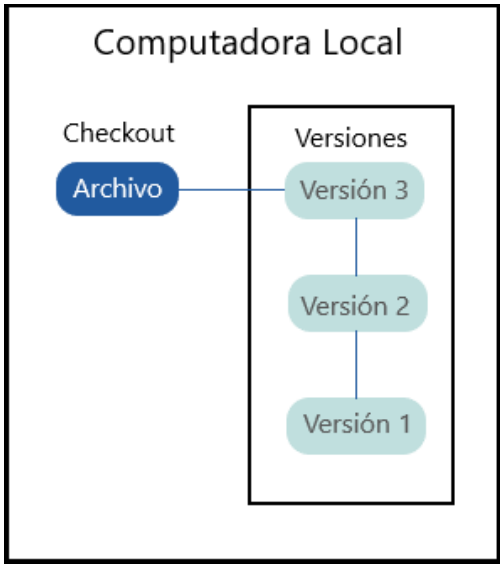


Figura 1. Diagrama de Control de versiones local [19]

Sistemas de control de versiones Centralizado

Este sistema de control de versiones se caracteriza por contar con un único directorio [repositorio] en un servidor central, donde se encuentran almacenados el código fuente del software y su historial de versiones [18].

Fueron desarrollados con la finalidad de posibilitar el trabajo colaborativo, ya que permiten que varios desarrolladores trabajen en un mismo proyecto accediendo al repositorio central mediante redes de área amplia conocidas como WAN [*Wide Area*

Network] o por redes de área local también llamadas LAN [*Local Area Network*] [18], [19].

Los sistemas de control de versiones centralizados más reconocidos son CVS y Subversion [SVN].

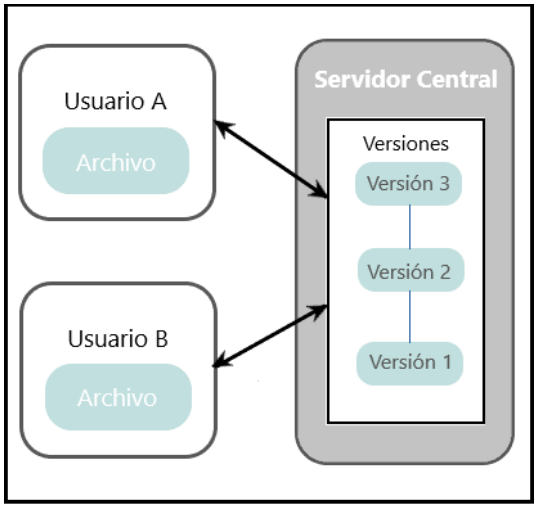


Figura 2. Diagrama de Control de versiones centralizado [19]

Sistemas de control de versiones Distribuido

En este tipo de sistemas cada usuario o desarrollador cuenta con su propia copia del repositorio y su historial de cambios en su computadora [18]. Esto permite que se establezcan varios flujos de trabajo y que en cualquier momento los usuarios puedan sincronizar sus repositorios entre sí [18], [19]. Las versiones modificadas son puestas como las más actuales y en caso de que se hubiera modificado un mismo documento, el sistema emite un alerta para que los usuarios implicados decidan qué cambios son más relevantes [21].

Es el tipo de VCS más popular en el área laboral y entre sus referentes se encuentran Git, Mercurial y Bazaar.

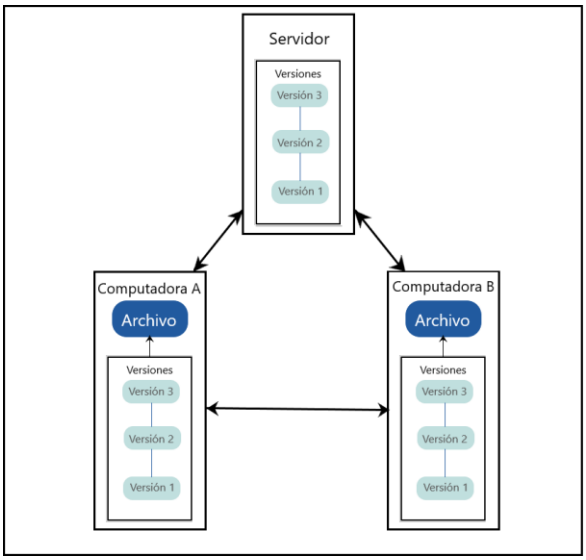


Figura 3. Diagrama de Control de versiones distribuido [19]

1.2.5. Repositorios de software

Los repositorios de software son un espacio de almacenamiento que puede estar en línea o no, es el lugar donde se mantienen los paquetes de software, el código fuente, logs, reporte de errores y mucha otra información relacionada con el software y su proceso de desarrollo [9]. Los repositorios de software contienen una gran cantidad de información valiosa sobre la evolución de un proyecto de software [22].

Según Siddiqui y Ahmad, los repositorios de software pueden ser de las siguientes formas:

Repositorios históricos

Estos repositorios contienen una cantidad enorme y heterogénea de datos de ingeniería de software generados entre largos períodos de tiempo [9]. Los repositorios históricos son esenciales para proporcionar a la industria de desarrollo de software información cuantitativa que puede derivarse exclusivamente de datos empíricos. Ellos proporcionan puntos de referencia externos para la productividad, la eficiencia, la calidad y otras características del proceso y producto de software [23].

Repositorios de tiempo de ejecución

Estos repositorios contienen información relacionada con la ejecución y el uso de una aplicación en uno o varios sitios de implementación diferentes, como registros de implementación [9].

Repositorio de código

Estos repositorios son espacios virtuales en donde se almacena el código fuente de proyectos, incluyendo sus historiales de cambios. Éstos pueden encontrarse dentro de un servidor o en computadoras de uso personal. Los repositorios de código se mantienen mediante la recopilación de código fuente de un gran número de proyectos heterogéneos. Sourceforge.net, GitHub y el Google code son ejemplos de repositorios de código [9].

1.2.6. Minería de repositorios de software

Es un campo de la Ingeniería de software en el cual se investigan técnicas para extraer y analizar la información valiosa que se encuentra en los repositorios de software [24]. Los desarrolladores almacenan en los repositorios discusiones de diseños, reportes de errores, historiales de cambios, commits, entre otros tipos de datos los cuales pueden ser extraídos para posteriormente ser analizados utilizando técnicas de MSR [25].

La MSR tiene como objetivo brindar herramientas que contribuyan al mejoramiento de las metodologías de desarrollo de software actuales.

1.2.6.1. Tipos de datos en la minería de repositorios de software

Durante el ciclo de vida de un software, los datos generados y almacenados en repositorios, presentan diferentes composiciones, lo cual puede facilitar o dificultar su análisis. Estos datos pueden ser clasificados en dos categorías, estructurados y no estructurados.

Datos no estructurados

Los datos no estructurados son aquellos que no están almacenados en un formato predefinido. Se suele referir a los textos de lenguaje natural como datos no estructurados,

ya que estos no cuentan con una estructura interna, y aunque para un humano estos tipos de datos sean fáciles de interpretar [26], resultan dificultosos para la mayoría de los softwares y algoritmos de minería ya que para ellos simplemente representan a un conjunto de caracteres sin sentido ni estructura, por lo que necesita ser organizada y preparada. Representan a la mayoría de datos en los repositorios de software y se pueden encontrar en: títulos y descripciones de informes de errores, datos lingüísticos en el código fuente (nombres de identificadores y comentarios), requerimientos, documentación, listas de correos, archivos multimedia, entre otros [7].

Datos estructurados

Los datos estructurados son aquellos que presentan un modelo y una forma conocida [7], representan menos de la mitad de los datos que pueden ser analizados pero resultan mucho más fáciles de analizar para cualquier software o algoritmo. Entre los datos estructurados presentes en los repositorios de software están el código fuente, scripts de bases de datos, metadatos de informes de errores, entre otros.

1.2.6.2. Alcance y limitaciones de la MSR

Las técnicas de MSR han permitido analizar la calidad del software codificado, evitar el llamado *code smell*, predecir y detectar errores, mejorar las habilidades de los desarrolladores, optimizar aplicaciones y apoyar al mantenimiento del software.

Estas herramientas, en algunas ocasiones, se encuentran limitadas al estar desarrolladas para repositorios basados en sistemas de control de versiones específicos, por lo que, esta práctica podría estar obstaculizada para repositorios con sistemas de control de versiones menos populares o antiguos. Pero el principal limitante de la Minería de repositorios de software son los datos, la cantidad de datos disponibles para MSR es abundante [27], sin embargo los resultados de MSR serán buenos dependiendo de la precisión de los datos en los que se basa [25].

1.2.6.3. Enfoques de la MSR

Según D'Ambros los estudios de minería de repositorios de software pueden ser categorizados según los siguientes enfoques:

Estudios empíricos: estudios en los que se pueden recopilar datos y realizar análisis empíricos para luego sacar conclusiones del tema.

Predicción de cambios: herramientas cuyo objetivo es crear modelos a partir del historial de cambios que existe en el repositorio, con la finalidad de identificar cómo una entidad [clases, métodos, etc.] puede afectar a otras, es decir si realizo un cambio en una entidad de mi programa cual será el impacto que tendrá sobre él y las demás entidades [25].

Predicción de problemas: se busca que mediante el uso de los reportes de errores del pasado se pueden crear modelos de predicción de errores futuros [25].

Recomendación de expertos y triaje de errores: gracias a que los repositorios guardan en su historial la información acerca de los desarrolladores que realizan los cambios, esto puede ser usado para determinar el conocimiento de programadores en determinados lenguajes de programación o tecnologías, lo cual permite tomar decisiones más acertadas al asignar tareas [27].

Búsquedas de código: acorta la resolución de problemas ya que, al existir una gran cantidad de software libre accesible en la web, es probable que alguien más ya tenga la solución.

Visualización de la evolución de un software: son herramientas visuales que ayudan a mostrar la evolución de un software a partir de datos obtenidos en los repositorios.

Aspectos humanos: el software está construido por humanos para humanos por lo que existe cierto interés por parte de los investigadores para explorar las herramientas de comunicación de los desarrolladores, tales como correos electrónicos, blogs, chats, etc. [27].

1.2.7. Estudios de mapeo sistemático

El estudio de mapeo sistemático es un tipo de revisión de la literatura, es decir una investigación secundaria [28]. Es un método empleado en la ingeniería de software y en otros campos de estudio para construir un esquema de clasificación y estructurar un campo de interés [29]. Los análisis de los resultados se realizan categorizando artículos relacionados con la temática y calculando la frecuencia de publicaciones dentro de cada categoría [30], los datos proporcionados en esta clasificación se utilizan para responder

las preguntas de investigación previamente planteadas, las cuales delimitan el alcance del estudio.

Conforme un campo de investigación va creciendo y toma interés, se ve un aumento en la realización de estudios y publicaciones relacionados con el tema, por lo que los estudios de mapeo sistemático permiten dar una visión general y cuál es la tendencia de esa área de conocimiento [31]. También esta metodología es utilizada con frecuencia como base para profundizar investigaciones futuras [32].

1.2.7.1. Herramientas utilizadas en estudios de mapeo sistemático.

Existen varias herramientas de apoyo, que facilitan la tarea de realizar un estudio de mapeo sistemático. Estas herramientas permiten recopilar, organizar y clasificar artículos de acuerdo con diferentes criterios, entre las más conocidas se encuentran las siguientes:

Mendeley

Es un gestor de referencias bibliográficas gratuito y una red social para académicos, que facilita la organización de investigaciones y colaborar con otros investigadores en línea [33]. Fue lanzado en el 2008 por Jan Reichelt, Victor Henning y Paul Foeckler, y posteriormente, adquirido por Elsevier en 2013 [34].

Es multiplataforma y cuenta con versiones web, de escritorio y móvil. Además Mendeley permite: generar bibliografías automáticamente, crear bases de datos de bibliográficas, almacenar y gestionar archivos y acceder a ellos desde cualquier lugar con acceso a internet [33].

EndNote

Es un paquete de gestión de referencias bibliográficas con licencia propietaria que es utilizado para manejar listados bibliográficos [35]. Lanzado en 1988 por Thomson Reuters y adquirido en el 2016 por Clarivate Analytics, quienes lo desarrollan actualmente.

La finalidad de EndNote es facilitar el trabajo de los investigadores, estudiantes y profesores en el proceso de realización de sus trabajos investigativos, favoreciendo el

trabajo colaborativo, la creación de bibliotecas bibliográficas, el intercambio de referencias con otros programas y mejorando la búsqueda de referencias y archivos de cientos de recursos en línea [36].

EndNote cuenta con una versión web y está diseñado para trabajar simultáneamente con el servicio de información científica *ISI Web of Knowledge* [37], lo cual simplifica el trabajo y proporciona acceso a un conjunto de bases de datos bibliográficas entre otros recursos de varios campos de conocimiento [38].

Zotero

Es un software de gestión de referencias bibliográficas de código abierto, que permite a los usuarios almacenar, administrar y organizar citas bibliográficas [39]. Es desarrollado por el *Roy Rosenzweig Center for History and New Media* de la Universidad George Mason [40] y fue creado por investigadores para investigadores.

Entre sus particularidades está el permitir guardar referencias desde bibliotecas, bases de datos bibliográficas, sitios web, y otras fuentes con la facilidad de un clic [41], gracias a su extensión para navegador.

RefWorks

Es una aplicación web multilingüe y comercial de gestión de referencias bibliográficas [42]. Al tener un entorno web, no requiere instalación y se puede acceder desde cualquier dispositivo conectado a internet. RefWorks LLC se fundó en el año 2001 y el software fue lanzado por *Cambridge Scientific Abstracts* a partir del 2002 hasta que fue adquirido por ProQuest en 2008.

RefWorks es una herramienta que permite: crear bases de datos bibliográficas en línea[43], exportar e importar referencias desde múltiples fuentes y compartir datos de investigación con otros usuarios sin importar si tienen o no Refworks [42].

QDA Miner

Es un poderoso software utilizado para el análisis de datos cualitativos y métodos mixtos que permite codificar, anotar, recuperar y analizar pequeñas y grandes colecciones de documentos e imágenes [44]. Es desarrollado por Provalis Research y se lanzó por

primera vez en el 2004. Es una herramienta de paga, pero cuenta con una versión gratuita llamada QDA Miner lite, la cual cuenta con las características básicas del programa principal.

QDA Miner es empleado para analizar transcripciones de entrevistas, documentos legales, artículos de revistas, discurso e incluso libros enteros y dibujos[44], lo que hace que sea utilizado en diferentes ramas como la política, la investigación, el periodismo, medicina, entre otros.

1.3. Marco Legal

Dentro de las bases legales que se tomaron en cuenta para esta investigación se encuentran: La Constitución de la República del Ecuador[45], el Código Orgánico de Economía Social de los Conocimientos, Creatividad e Innovación [COESCCI] [46] y La Ley de Propiedad Intelectual [47]:

En el COESCCI, en su artículo 142 que habla de Tecnologías libres, se menciona que: se entiende por software de código abierto al software en cuya licencia el titular garantiza al usuario el acceso al código fuente y lo faculta a usar dicho software con cualquier propósito. Este estudio se basa en la investigación de herramientas de minería de repositorios de software, con las cuales se pueda mejorar la calidad del desarrollo y mantenimiento del software. En el proceso, los datos obtenidos para este estudio fueron tomados de plataformas de alojamiento de repositorios en donde se encuentran millones de proyectos que cuentan con licencia de software libre. Se considera también necesario mencionar a La Ley de Propiedad Intelectual del Ecuador y al artículo 322 de la Constitución de la República del Ecuador que menciona que se reconoce la propiedad intelectual de acuerdo con las condiciones que señala dicha ley y se prohíbe toda forma de apropiación de conocimientos colectivos, en el ámbito de las ciencias, tecnologías y saberes ancestrales.

CAPÍTULO II

2. Metodología

2.1. Población y muestra del estudio

En esta investigación se realizó la búsqueda de artículos en conferencias y bases de datos bibliográficas, que pertenezcan a la temática de Minería de Repositorios de Software, de la misma forma se utilizaron como bases de datos referenciales a IEEE, ACM digital Library, Springer, Web of Science y Scopus, por su importancia en la comunidad científica a nivel internacional, sobre todo en el área de la Ingeniería de Software. Por otro lado, los datos que fueron considerados para la realización del mapeo sistemático se encuentran enmarcados dentro del periodo 2015 – 2022, ya que se consideró el espacio de tiempo necesario para recopilar los estudios más recientes en el área.

2.2. Tipo de investigación

Dada la naturaleza del estudio realizado, se consideró enfocarlo en los siguientes tipos de investigación: descriptivo porque a través de diferentes fuentes de información se obtuvieron datos que describían el comportamiento del objeto de estudio. Cuantitativo, ya que la información recopilada en ella fue medida y clasificada según las características que presentaron y que facilitaron la comprensión de la temática. Y por último documental debido a que toda la información que se obtuvo fue extraída de bases de datos bibliográficas.

2.3. Métodos y técnicas

El método de investigación que fue empleado en la primera etapa de este estudio fue el inductivo, dado que de una premisa se buscó información con la finalidad ampliar los conocimientos acerca de la temática. Y durante su faceta de desarrollo se utilizó el método deductivo, ya que, a través de un estudio de mapeo sistemático, se analizaron distintos artículos y publicaciones acerca de la Minería de repositorios de software y sus

herramientas, y se pudieron obtener varias conclusiones acerca del panorama de esta área dentro de la Ingeniería de Software.

2.4. Población y muestra de estudio

Arias en el 2006, define población o población objetivo a: "Un conjunto finito o infinito de elementos con características comunes para los cuales serán extensivas conclusiones de la investigación. Esta queda determinada por el problema y por los objetivos del estudio [48]."

Considerando las características y los objetivos de la investigación se tomó como población al conjunto conformado por todos los artículos publicados en el periodo 2015-2022, pertenecientes a la temática de minería de repositorios de software. Dichos artículos fueron extraídos de conferencias, revistas y distintas bases de datos bibliográficas referenciales como IEEE, ACM digital Library, Springer, Web of Science y Scopus.

2.5. Variables de investigación

Tabla 1. Principales variables e indicadores sujetos a estudio

Variables	Indicadores	Tipo de Variables	Entidades
Distribución de artículos.	▪ Fuente Bibliográfica ▪ Año ▪ País ▪ Tipo	Cuantitativa	Artículos
Herramienta	▪ Aplicabilidad ▪ Popularidad ▪ Tipo ▪ Enfoque	Cuantitativo	Herramientas para MSR.

	▪ Datos de Entrada		
Efectividad	▪ Exactitud (*Accuracy*) ▪ Precisión	Cualitativa	Herramientas para MSR.

Las descripciones de las variables listadas en la Tabla 1, así como sus correspondientes indicadores se describen como sigue:

Distribución: Es la división de los artículos estudiados tomando en cuenta diferentes parámetros.

- **Fuente Bibliográfica**. Clasificación de los artículos según la base de datos en donde fueron encontrados. Considerando que el siguiente estudio fue basado en cinco bases de datos, las cuales fueron: IEEE, ACM digital Library, Springer, Web of Science y Scopus.
- **Año**. Número de artículos clasificados según su año de publicación.
- **País**. Número de artículos clasificados según el país de origen del estudio.
- **Tipo**. Número de artículos clasificados según el tipo de investigación que se realiza en el artículo.

Herramienta: Características que presentan las herramientas de minería de software.

- **Aplicabilidad**. Herramientas que son más aplicables o recomendadas.
- **Popularidad**. Herramientas más utilizadas.
- **Tipo**. Clasificación de herramientas según la clase de aplicación que es y su entorno de ejecución.
- **Objetivo**. Finalidad que busca la herramienta al realizar el proceso de MSR.
- **Datos de Entrada**. Tipo de datos de entrada que utiliza la herramienta para procesar y en base a ella emitir un resultado.

Efectividad: Es la capacidad de conseguir el resultado deseado.

- **Exactitud (*Accuracy*).** Cuan exacta o correcta es la herramienta.

- **Precisión**. Se denomina precisión a la capacidad de un instrumento de dar el mismo resultado en mediciones diferentes realizadas en las mismas condiciones.

2.6. Técnicas e instrumentos de recolección de datos

La técnica de investigación que se utilizó para el estudio fue el análisis bibliográfico, el cual se realizó con la metodología de estudio de mapeo sistemático, que es ampliamente utilizado para observar la evolución en una determinada área de investigación.

El protocolo de búsqueda científica utilizado para el estudio fue principalmente la metodología propuesta por Petersen et al [29], quienes diseñaron una técnica para los estudios de mapeos sistemáticos enfocados al área de la Ingeniería de Software. Como técnica complementaria se usó la llamada *Snowballing* (Bola de nieve)[49], la cual beneficia a la búsqueda de artículos al identificar estudios citados o referenciados en otros artículos.

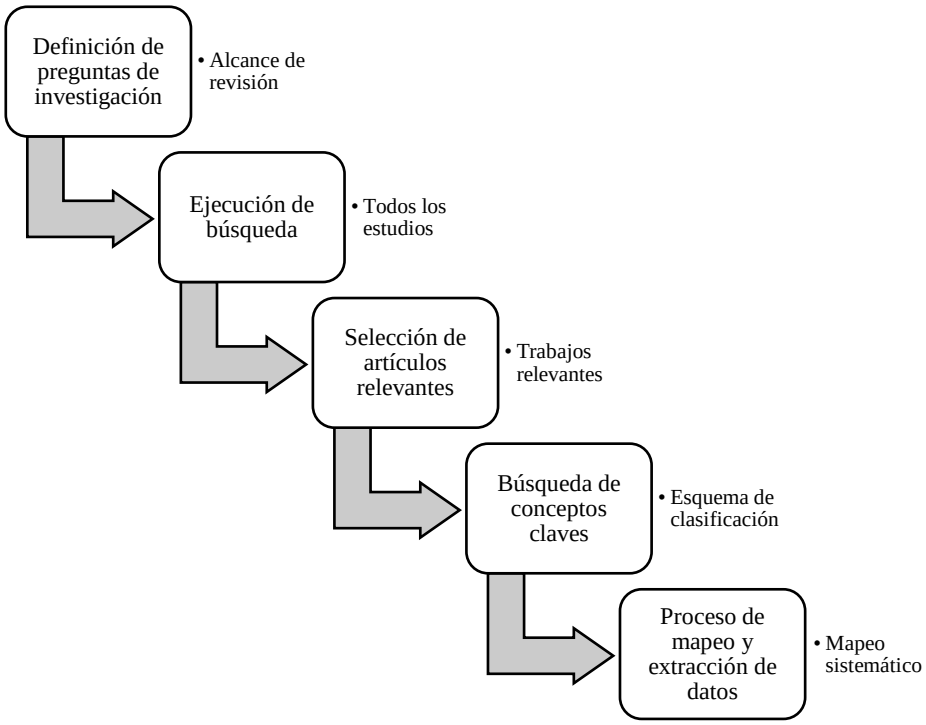


Figura 4. Pasos para un mapeo sistemático según la metodología de Petersen [29].

Descripción de los pasos definidos:

Definición de preguntas de investigación: El objetivo principal de los estudios de mapeo sistemático es proporcionar una descripción general de un área de investigación e identificar la cantidad, el tipo de investigación y los resultados disponibles dentro de ella. A menudo, se desea mapear las frecuencias de publicación a lo largo del tiempo para ver las tendencias. Estos objetivos se reflejan en las preguntas de investigación de los artículos, que son similares.

Ejecución de búsqueda: Los estudios primarios se identifican mediante el uso de cadenas de búsqueda en bases de datos científicas o navegando manualmente a través de actas de congresos o publicaciones de revistas relevantes.

Selección de artículos relevantes: Los criterios de inclusión y exclusión se utilizan para excluir estudios que no son relevantes para responder a las preguntas de investigación.

Búsqueda de conceptos claves: En este paso, los revisores leen los resúmenes y buscan palabras clave y conceptos que reflejen la contribución del artículo. Al hacerlo, el revisor también identifica el contexto de la investigación. Cuando se hace esto, el conjunto de palabras clave de diferentes artículos se combinan para desarrollar un conocimiento de alto nivel sobre la naturaleza y la contribución de la investigación. Esto ayuda a los revisores a crear un conjunto de categorías que sea representativo de la población subyacente.

Extracción de datos y mapeo: Este es el último paso del proceso, en el cual se realiza la extracción de los datos, ayudándose del esquema de clasificación, para a continuación realizar el mapeo y obtener los resultados del estudio.

Descripción del proceso del estudio de mapeo sistemático

Para llevar a cabo el presente trabajo de investigación se siguieron los pasos antes mencionados, pertenecientes a la metodología de Petersen [29].

2.6.1. Definición de las preguntas de investigación

Las preguntas de investigación fueron definidas de acuerdo con los objetivos principales que buscó alcanzar el estudio, los cuales fueron conocer la evolución que ha tenido la minería de repositorios de software en los últimos años y cuáles son las herramientas que se utilizan actualmente para este propósito en el desarrollo de software. Debido a esto se definieron las siguientes preguntas de investigación, con la finalidad de que fueran respondidas conforme se iban cumpliendo los objetivos de esta.

Tabla 2. Preguntas de investigación

Preguntas de Investigación	Motivación
PI1: ¿Qué herramientas para la minería de repositorios de software se han desarrollado durante los últimos años?	Conocer las herramientas que se han desarrollado para la minería de repositorios de software en los últimos 7 años.
PI2: ¿Qué categorías existen para clasificar a las herramientas de MSR existen?	Determinar las categorías en las que se pueden clasificar las herramientas para MSR según su enfoque.
PI3: ¿Cuáles son las herramientas para MSR más utilizadas en la industria del desarrollo de software?	Definir cuáles son las herramientas que más se utilizan y con mayor aplicabilidad dentro de la industria del desarrollo.

2.6.2. Fuentes de datos y estrategia de búsqueda

En el proceso de búsqueda se realizó en las bases de datos especificadas anteriormente en la sección de población y muestra. La cadena de búsqueda utilizada fue la siguiente: (Mining) AND (software OR code) AND (Repository). Cabe recalcar que junto a la cadena de búsqueda se realizó una filtración de artículos para que estos correspondieran al periodo 2015 – 2022.

Además de la cadena de búsqueda se hizo uso de la técnica de Snowballing, con la cual se podían encontrar artículos a través del uso de citas y referencias bibliográficas, estos

artículos eran seleccionados siempre y cuando pertenecieran a la temática y su publicación no sobrepasaba el periodo de este estudio.

2.6.3. Selección de estudios

Para la selección de los estudios primarios se definieron los criterios de inclusión y exclusión, que fueron empleados para la filtración de los artículos obtenidos durante el proceso de búsqueda.

Criterios de inclusión: Artículos completos referentes a la minería de repositorios de software, que cuenten con la utilización de una herramienta y detallados en el mismos los datos de entrada de la herramienta, el dataset utilizado y su aplicación.

Criterios de exclusión: Artículos cortos y artículos no referentes a la minería de repositorios de software o que no describan una herramienta o su implantación.

No se consideraron relevantes y fueron excluidos aquellos resultados de búsquedas que no cumplían con los criterios de inclusión mencionados.

2.6.4. Proceso de clasificación

Siguiendo la metodología de Petersen, quien sugiere un método de clasificación que es llamado *Keywording*, el cual consiste en la extracción de palabras claves y conceptos, a raíz de la lectura del resumen del artículo. Esto se hace con la finalidad de ahorrar tiempo en la definición de un criterio de clasificación y encontrar categorías que tengan concordancia con los resultados obtenidos.

Siguiendo esta metodología se obtuvieron los siguientes esquemas de clasificación:

- **Enfoque de la herramienta:** describe para que área de la minería de repositorios de código la herramienta está siento utilizada.
  - **Estudios empíricos:** herramientas en las cuales se realizan análisis de diferentes partes del software con la finalidad de buscar solución a problemas particulares.
  - **Predicción de cambios:** herramientas con las cuales partiendo de datos históricos se puede predecir los cambios y los efectos que tendrán dentro del código.

- **Predicción de errores:** herramientas que predicen errores basándose en reportes de errores anteriores.
- **Recomendación de expertos:** herramientas que recomiendan que desarrollador está más acto para una tarea, basándose en sus historiales de cambios.
- **Búsquedas de código:** herramientas que acortan la resolución de problemas al facilitar la búsqueda de código en diferentes fuentes.
- **Análisis de sentimientos:** herramientas que analizan la relación de los sentimientos con la productividad.

- **Datos de entrada:** describe que datos de entrada reciben las aplicaciones de Minería de repositorios de software.
  - **Reporte de bugs:** reporte de errores que se han dado durante la ejecución del software.
  - **Historial de versiones:** versiones almacenadas en los sistemas de control de versiones.
  - **Historial de cambios:** registro de los cambios que se han realizado en el software.
  - **Código fuente:** utilizando todo el código del software.
  - **Revisiones:** revisiones de código realizados por desarrolladores.
  - **Otros:** datos de entrada variados, que pueden ser desde post en StackOverFlow hasta podcast y videos.

2.7. Técnicas de procesamiento y análisis de datos

Para analizar los datos recopilados durante la investigación se emplearon diferentes técnicas de análisis de datos:

Análisis de patentes y literatura científica: que permitió extraer información importante a través de los metadatos que se encontraron en los artículos, tales como autor, país de origen, año, etc.

Estadística descriptiva: una vez extraídos los metadatos de los artículos a través de esta técnica se pudieron clasificar y tabular.

Visualización de datos: los datos tabulados se colocaron en gráficos, los cuales facilitaron su interpretación.

2.8. Normas éticas

La presente investigación se realizó de acuerdo con los lineamientos establecidos en el Reglamento de grados de la PUCESE, el respeto de los derechos de autor, Ley de Propiedad Intelectual y el respeto a las licencias de softwares empleados.

CAPÍTULO III

3. Resultados

La búsqueda ejecutada en las diferentes bases de datos bibliográficas y las conferencias de Ingeniería de software, dieron como resultado preliminar una recopilación de 611 estudios. Al aplicar el primer filtro basado en los criterios de inclusión y exclusión aplicados sobre el título y el resumen se obtuvieron 304 artículos. Y después de realizar el segundo filtro aplicado sobre el texto total de las investigaciones, se seleccionaron finalmente 99 artículos tal como se encuentra reflejado en la figura 5. de los que se extrajeron los siguientes de resultados:

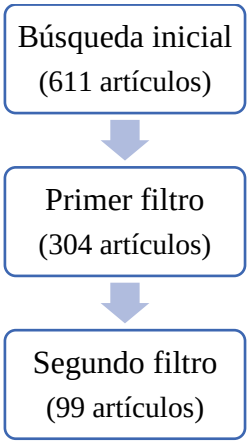


Figura 5. Proceso de selección de artículos.

Del total de producciones científicas seleccionadas y la gestión de su conocimiento, se reflejan en la Tabla 3, contando con información sobre títulos, autores, país y año de publicación.

Tabla 3. Estudios analizados para el mapeo sistemático de herramientas para la minería de repositorios de Software durante el periodo 2015-2022

#	Título	Autor	País
E1	A Method to Detect License Inconsistencies in Large-Scale Open-Source Projects	Yuhao Wu, Yuki Manabe, Tetsuya Kanda, Daniel M. German, Katsuro Inoue	Japón, Canadá
E2	Toward Deep Learning Software Repositories	Martin White, Christopher Vendome, Mario Linares-V´asquez, and Denys Poshyvanyk	Estados Unidos

E3	Are Bullies more Productive? Empirical Study of Affectiveness vs. Issue Fixing Time	Marco Ortu, Bram Adams, Giuseppe Destefanis, Parastou Tourani, Michele Marchesi, Roberto Tonelli	Italia, Canadá
E4	An Empirical Study of End-user Programmers in the Computer Music Community	Gregory Burlet, Abram Hindle	Canadá
E5	Do Bugs Foreshadow Vulnerabilities? A Study of the Chromium Project	Felivel Camilo, Andrew Meneely, Meiyappan Nagappan	Estados Unidos
E6	An Empirical Study of Architectural Change in Open-Source Software Systems	Duc Minh Le, Pooyan Behnamghader,Joshua Garcia, Daniel Link, Arman Shahbazian, Nenad Medvidovic	Estados Unidos
E7	Recommending Posts Concerning API Issues in Developer Q&A Sites	Wei Wang, Haroon Malik, Michael W. Godfrey	Canadá
E8	A Study on the Role of Software Architecture in the Evolution and Quality of Software	Ehsan Kouroshfar, Mehdi Mirakhorli, Hamid Bagheri, Lu Xiao, Sam Malek, Yuanfang Cai	Estados Unidos
E9	A Historical Analysis of Debian Package Incompatibilities	Maelick Claes, Tom Mens, Roberto Di Cosmo J´erˆome Vouillon	Bélgica, Francia
E10	Investigating Code Review Practices in Defective Files: An Empirical Study of the Qt System	Patanamon Thongtanunam, Shane McIntosh, Ahmed E. Hassan, Hajimu Iida	Canadá
E11	Characteristics of Useful Code Reviews: An Empirical Study at Microsoft	Amiangshu Bosu, Michaela Greiler, Christian Bird	Estados Unidos
E12	Unveiling Exception Handling Bug Hazards in Android based on GitHub and Google Code Issues	Roberta Coelho, Lucas Almeida, Georgios Gousios, Arie van Deursen	Brasil, Países Bajos
E13	Mining Android App Usages for Generating Actionable GUI-based Execution Scenarios	Mario Linares-Vásquez, Martin White, Carlos Bernal-Cárdenas, Kevin Moran, Denys Poshyvanyk	Estados Unidos
E14	An Empirical Study of the Copy and Paste Behavior during Development	Tarek M. Ahmed, Weiyi Shang, Ahmed E. Hassan	Canadá
E15	Mining Energy-Aware Commits	Irineu Moura, Gustavo Pinto, Felipe Ebert and Fernando Castor	Brasil
E16	Co-evolution of Infrastructure and Source Code - An Empirical Study	Yujuan Jiang, Bram Adams	Canadá
E17	Sameness: An Experiment in Code Search	Lee Martie and André van der Hoek	Estados Unidos

E18	Extracting Facts from Performance Tuning History of Scientific Applications for Predicting Effective Optimization Patterns	Masatomo Hashimoto, Masaaki Terai, Toshiyuki Maeda and Kazuo Minami	Japón
E19	Mining Component Repositories for Installability Issues	Pietro Abate, Roberto Di Cosmo, Louis Gesbert, Fabrice Le Fessant, Ralf Treinen, Stefano Zacchiroli	Francia
E20	How Android App Developers Manage Power Consumption?	Lingfeng Bao, David Lo, Xin Xia, Xinyu Wang, Cong Tian	China
E21	Mining Test Repositories for Automatic Detection of UI Performance Regressions in Android Apps	María Gómez, Romain Rouvoy, Bram Adams, Lionel Seinturier	Francia, Canadá
E22	GreenOracle: Estimating Software Energy Consumption with Energy Measurement Corpora	Shaiful Alam Chowdhury, Abram Hindle	Canadá
E23	Mining Performance Regression Inducing Code Changes in Evolving Software	Qi Luo, Denys Poshyvanyk, Mark Grechanik	Estados Unidos
E24	FEVER: Extracting Feature-oriented Changes from Commits	Nicolas Dintzner, Arie van Deursen, Martin Pinzger	Austria, Países Bajos
E25	Comparing Repositories Visually with RepoGrams	Daniel Rozenberg; Ivan Beschastnikh; Fabian Kosmale; Valerie Poser; Heiko Becker; Marc Palyart; Gail C. Murphy	Alemania, Canadá
E26	Cold-Start Software Analytics	Jin Guo, Mona Rahimi, Jane Cleland-Huang,Alexander Rasin, Jane Huffman Hayes, Michael Vierhauser	Estados Unidos, Austria
E27	Interactive Exploration of Developer Interaction Traces using a Hidden Markov Model	Kostadin Damevski, Hui Chen, David Shepherd, Lori Pollock	Estados Unidos
E28	Does Your Configuration Code Smell?	Tushar Sharma, Marios Fragkoulis, Diomidis Spinellis	Grecia
E29	Understanding the Exception Handling Strategies of Java Libraries: An Empirical Study	Demóstenes Sena, Roberta Coelho, Uirá Kulesza, Rodrigo Bonifácio	Brasil
E30	Externalization of Software Behavior by the Mining of Norms	Daniel Avery, Hoa Khanh Dam, Bastin Tony Roy, Aditya Ghose	Australia, Nueva Zelanda
E31	Sentiment analysis in tickets for IT support	Cássio Castaldi Araujo Blaz, Karin Becker	Brasil
E32	Mining Valence, Arousal, and Dominance – Possibilities for Detecting Burnout and Productivity?	Mika Mäntylä, Bram Adams, Giuseppe Destefanis, Daniel Graziotin, Marco Ortu	Canadá, Alemania, Reino Unido, Finlandia, Italia

E33	The Unreasonable Effectiveness of Traditional Information Retrieval in Crash Report Deduplication	Joshua CharlesCampbell, Eddie Antonio Santos, Abram Hindle	Canadá
E34	Using Dynamic and Contextual Features to Predict Issue Lifetime in GitHub Project	Riivo Kikas, Marlon Dumas, Dietmar Pfahl	Estonia
E35	Topic Modeling of NASA Space System Problem Reports	Lucas Layman, Allen P. Nikora, Joshua Meek, Tim Menzies	Estados Unidos
E36	A Look at the Dynamics of the JavaScript Package Ecosystem	Erik Wittern, Philippe Suter, Shriram Rajagopalan	Estados Unidos
E37	Grouping Android Tag Synonyms on Stack Overflow	Stefanie Beyer, Martin Pinzger	Austria
E38	Domain-Specific Cross-Language Relevant Question Retrieval	Bowen Xu, Zhenchang Xing, Xin Xia, David Lo, Qingye Wang, Shanping Li	Singapur, China
E39	Candoia: A platform for building and sharing mining software repositories tools as apps	Nitin M Tiwari, Ganesha Upadhyaya, Hoan Anh Nguyen, Hridesh Rajan	Estados Unidos
E40	Developer mistakes in writing android manifests: An empirical study of configuration errors	Ajay Kumar Jha, Sunghee Lee, Woo Jin Lee	Corea del sur
E41	Spencer: Interactive heap analysis for the masses	Stephan Brandauer; Tobias Wrigstad	Suecia
E42	Analyzing program dependencies in Java EE applications	Anas Shatnawi, Hafedh Mili, Ghizlane El Boussaidi, Anis Boubaker, Yann-Gaël Guéhéneuc,Naouel Moha, Jean Privat, Manel Abdellatif	Canadá
E43	Who you gonna call? Analyzing Web Requests in Android Applications	Marianna Rapoport, Philippe Suter, Erik Wittern, Ondrej Lhóták, Julian Dolby	Canadá, Estados Unidos
E44	Predicting likelihood of requirement implementation within the planned iteration: An empirical study at IBM	Ali Dehghan, Adam Neal, Kelly Blincoe, Johan Linaker, Daniela Damian	Nueva Zelanda, Canadá, Suecia
E45	Predicting Usefulness of Code Review Comments using Textual Features and Developer Experience	Mohammad Masudur Rahman, Chanchal K. Roy, Raula G. Kula	Canadá, Japón
E46	Leveraging Automated Sentiment Analysis in Software Engineering	Md Rakibul Islam; Minhaz F. Zibran	Estados Unidos
E47	RefDiff: Detecting Refactorings in Version Histories	Danilo Silva∗, Marco Tulio Valente	Brasil
E48	Mining Change Histories for Unknown Systematic Edits	Tim Molderez, Reinout Stevens, Coen De Roove	Bélgica
E49	Some From Here, Some From There: Cross-Project Code Reuse in GitHub	Mohammad Gharehyazie, Baishakhi Ray, Vladimir Filkov	Estados Unidos
E50	Extracting Build Changes with BUILDDIFF	Christian Macho, Shane McIntosh, Martin Pinzger	Canadá, Austria

E51	An Empirical Analysis of the Docker Container Ecosystem on GitHub	Jürgen Cito, Gerald Schermann, John Erik Wittern, Philipp Leitner, Sali Zumberi, Harald C. Gall	Estados Unidos, Suiza
E52	An Empirical Analysis of Build Failures in the Continuous Integration Workflows of Java-Based Open-Source Software	Thomas Rausch, Waldemar Hummer, Philipp Leitner, Stefan Schulte	Austria, Suiza
E53	Euphony: Harmonious Unification of Cacophonous Anti-Virus Vendor Labels for Android Malware	Médéric Hurier, Guillermo Suarez-Tangil, Santanu Kumar Dash,Tegawendé F. Bissyandé, Yves Le Traon, Jacques Klein, Lorenzo Cavallaro	Reino Unido, Luxemburgo
E54	To Mock or Not to Mock? An Empirical Study on Mocking Practices	Davide Spadini, Maurício Aniche, Magiel Bruntink, Alberto Bacchelli	Países Bajos
E55	CLEVER: Combining Code Metrics with Clone Detection for Just-In-Time Fault Prevention and Resolution in Large Industrial Projects	Mathieu Nayrolles, Abdelwahab Hamou-Lhadj	Canadá
E56	What did Really Change with the new Release of the App?	Paolo Calciati, Konstantin Kuznetsov, Xue Bai, Alessandra Gorla	España, China, Alemania
E57	On the impact of security vulnerabilities in the npm package dependency network	Alexandre Decan, Tom MensEleni Constantinou	Suecia, Suiza
E58	Profiling call changes via motif mining	Barbara Russo	Italia
E59	Characterising Deprecated Android APIs	Li Li, Jun Gao, Tegawendé F. Bissyandé, Lei Ma, Xin Xia, Jacques Klein	Australia, Luxemburgo, China
E60	The Android Update Problem: An Empirical Study	Mehran Mahmoudi, Sarah Nadi	Canadá
E61	A Benchmark Study on Sentiment Analysis for Software Engineering Research	Nicole Novielli, Daniela Girardi, Filippo Lanubile	Italia
E62	Prevalence of Confusing Code in Software Projects	Dan Gopstein, Hongwei Henry Zhou,	Estados Unidos
E63	How Swift Developers Handle Errors	Nathan Cassee, Gustavo Pinto, Fernando Castor, Alexander Serebrenik	Países Bajos, Brasil
E64	What Are Your Programming Language's Energy-Delay Implications?	Stefanos Georgiou, Maria Kechagia, Panos Louridas, Diomidis Spinellis	Grecia
E65	Which Contributions Predict Whether Developers Are Accepted Into GitHub Teams	Justin Middleton, Emerson Murphy-Hill, Demetrius Green,Adam Meade, Roger Mayer, David White, Steve McDonald	Estados Unidos

E66	Automatic Classification of Software Artifacts in Open-Source Applications	Yuzhan Ma, Sarah Fakhoury, Michael Christensen, Venera Arnaoudova, Waleed ZogaanMehdi, Mirakhorli	Estados Unidos
E67	Large-Scale Analysis of the Co-Commit Patterns of the Active Developers in GitHub's Top Repositories	Eldan Cohen, Mariano P. Consens	Canadá
E68	Analyzing Requirements and Traceability Information to Improve Bug Localization	Michael Rath, David Lo, Patrick Mäder	Alemania, Singapur
E69	Towards Extracting Web API Specifications from Documentation	Jinqiu Yang, Erik Wittern, Annie T.T. Ying, Julian Dolby, Lin Tan	Canadá, Estados Unidos
E70	Analyzing Conflict Predictors in Open-Source Java Projects	Paola Accioly, Paulo Borba, Léuson Silva, and Guilherme Cavalcanti	Brasil
E71	Imprecisions Diagnostic in Source Code Deltas	Guillermo de la Torre, Romain Robbes, Alexandre Bergel	Chile, Italia
E72	Was Self-Admitted Technical Debt Removal a real Removal? An In-Depth Perspective	Fiorella Zampetti, Alexander Serebrenik, Massimiliano Di Penta	Italia, Países Bajos
E73	Exploring the Use of Automated API Migrating Techniques in Practice: An Experience Report on Android	Maxime Lamothe, Weiyi Shang	Canadá
E74	DeepJIT: An End-To-End Deep Learning Framework for Just-In-Time Defect Prediction	Thong Hoang, Hoa Khanh Dam, Yasutaka Kamei, David Lo, Naoyasu Ubayashi	Singapur, Japón, Australia
E75	An Empirical Study of Method Chaining in Java	Tomoki Nakamaru, Tomomasa Matsunaga, Tetsuro Yamazaki, Soramichi Akiyama, Shigeru Chiba	Japón
E76	Semantic Source Code Models Using Identifier Embeddings	Vasiliki Efstathiou, Diomidis Spinellis	Grecia
E77	Import2vec: Learning Embeddings for Software Libraries	Bart Theeten, Frederik Vandeputte, Tom Van Cutsem	Bélgica
E78	STRAIT: A Tool for Automated Software Reliability Growth Analysis	Stanislav Chren, Radoslav Micko, Barbora Buhnova, Bruno Rossi	República Checa

E79	Exploring Word Embedding Techniques to Improve Sentiment Analysis of Software Engineering Texts	Eeshita Biswas, K. Vijay-Shanker, Lori Pollock	Estados Unidos
E80	An Empirical Study of Method Chaining in Java	Tomoki Nakamaru, Tomomasa Matsunaga, Tetsuro Yamazaki, Soramichi Akiyama, Shigeru Chiba	Japón
E81	Extracting API tips from developer question and answer websites	Wang, Shaohua; Phan, Nhathai; Wang, Yan; Zhao, Yong	Estados Unidos
E82	Automatically generating documentation for lambda expressions in Java	Alqaimi, Anwar; Thongtanunam, Patanamon; Treude, Christoph	Australia
E83	Investigating next steps in static API-Misuse detection	Sven, Amann; Nguyen, Hoan Anh; Nadi, Sarah; Nguyen, Tien N.; Mezini, Mira	Canadá, Estados Unidos, Alemania
E84	Automated software vulnerability assessment with concept drift	Le, Triet Huynh Minh; Sabir, Bushra; Babar, Muhammad Ali	Australia
E85	The rise of android code smells: Who is to blame?	Habchi, Sarra; Moha, Naouel; Rouvoy, Romain	Francia, Canadá
E86	Predicting co-changes between functionality specifications and source code in behavior driven development	Yang, Aidan Z. H.; Costa, Daniel Alencar Da; Zou, Ying	Canadá, Nueva Zelanda
E87	AIMMX: Artificial Intelligence Model Metadata Extractor	Tsay, Jason; Braz, Alan; Hirzel, Martin; Shinnar, Avraham; Mummert, Todd	Brasil, Estados Unidos
E88	On the Prevalence, Impact, and Evolution of SQL Code Smells in Data-Intensive Systems	Muse, Biruk Asmare; Rahman, Mohammad Masudur; Nagy, Csaba; Cleve, Anthony; Khomh, Foutse; Antoniol, Giuliano	Canadá, Italia

E89	Detecting Video Game-Specific Bad Smells in Unity Projects	Borrelli, Antonio; Nardone, Vittoria; Lucca, Giuseppe A. Di; Canfora, Gerardo; Penta, Massimiliano Di	Italia
E90	SoftMon: A Tool to Compare Similar Open-source Software from a Performance Perspective	Singh, Shubhankar Suman; Sarangi, Smruti R.	India
E91	Gambit - An open source name disambiguation tool for version control systems	Gote, Christoph; Zingg, Christian	Suiza
E92	Mining Scientific Workflows for Anomalous Data Transfers	Tu, Huy; Papadimitriou, George; Kiran, Mariam; Wang, Cong; Mandal, Anirban; Deelman, Ewa; Menzies, Tim	Estados Unidos
E93	On the Naturalness and Localness of Software Logs	Gholamian, Sina; Ward, Paul A. S.	Canadá
E94	Mining Energy-Related Practices in Robotics Software	Albonico, Michel; Malavolta, Ivano; Pinto, Gustavo; Guzman, Emitza; Chinnappan, Katerina; Lago, Patricia	Estados Unidos
E95	On the use of dependabot security pull requests	Alfadel, Mahmoud; Costa, Diego Elias; Shihab, Emad; Mkhallalati, Mouafak	Canadá
E96	Mining Code Review Data to Understand Waiting Times Between Acceptance and Merging: An Empirical Analysis	Kudrjavets, Gunnar; Kumar, Aditya; Nagappan, Nachiappan; Rastogi, Ayushi	Países Bajos, Estados Unidos
E97	Recommending Code Improvements Based on Stack Overflow Answer Edits	Ragkhitwetsagul, Chaiyong; Paixao, Matheus	Tailandia, Brasil
E98	Semantic Similarity Metrics for Evaluating Source Code Summarization	Haque, Sakib; Eberhart, Zachary; Bansal, Aakash; McMillan, Collin	Estados Unidos

E99	CLIP meets GamePhysics: Towards bug identification in gameplay videos using zero-shot transfer learning;	Taesiri, Mohammad Reza; Bezemer, Paul	Canadá

3.1. PI2: Clasificación de las herramientas para MSR

La basta cantidad de herramientas arrojadas durante la investigación y sobre todo la información obtenida acerca de ellas, permitió que se establecieran dos criterios para su clasificación: según el enfoque de su uso en el estudio y según los datos de entrada que recibe cada una.

La clasificación según el enfoque se encuentra plasmada en primer lugar en la figura 6, en donde se puede observar la clasificación de los artículos según ocho enfoques identificados durante la investigación, notando que de estos al que más apuntan los investigadores es al estudio empírico, destacando sobre los demás con un porcentaje del 46.25% de los 99 artículos seleccionado, seguido por la búsqueda de código con un 12.5%, evolución del software con un porcentaje de 11.25% y aspectos humanos con el 10% de la muestra. Con menores porcentajes se encuentran la detección de errores, la predicción de cambios y predicción de errores, empatados con un porcentaje del 5% y por ende determinando que el enfoque al que menos se estudia es la predicción de vulnerabilidades con un porcentaje del 2.5%. En la investigación también se encontraron artículos con enfoques múltiple tales como el E26 y el E39 representando de igual forma el 2.5% de la muestra cómo se indica en el gráfico.

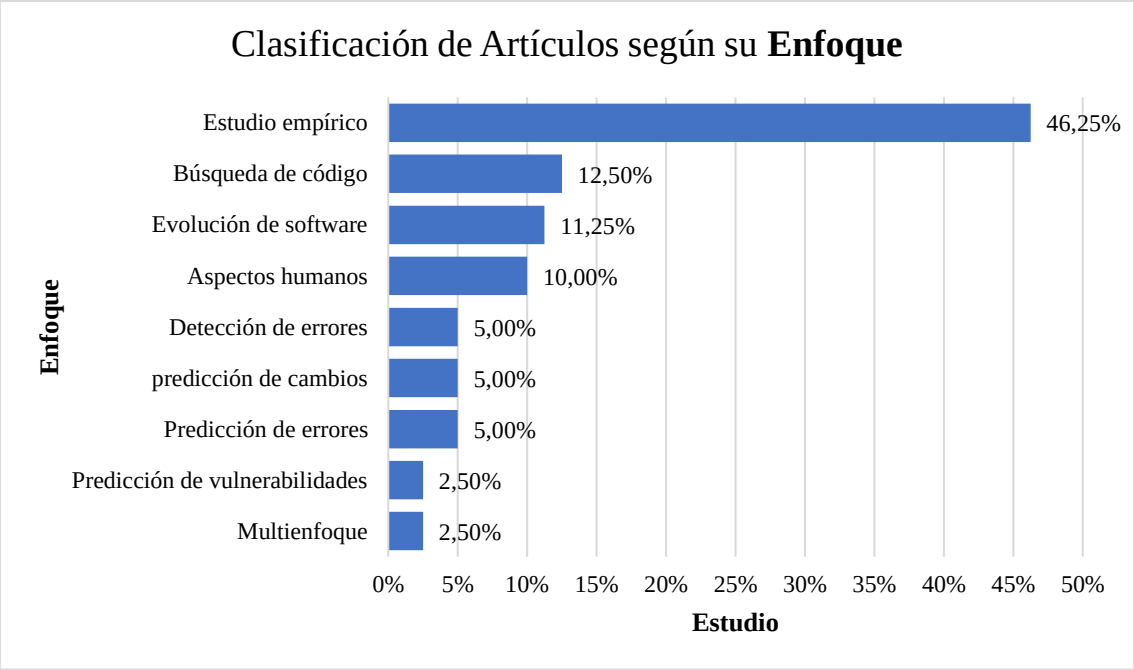


Figura 6. Clasificación de artículos según su enfoque

Tomando como referencia a las herramientas, la clasificación devuelve los siguientes y donde además se registró la frecuencia de uso de estas. Siendo notorio los casos de ciertas herramientas debido a su recurrencia en diferentes enfoques e investigaciones, entre los cuales están: git Mining Tools utilizado para enfoques como estudios empíricos, predicción de cambios y predicción de errores, con una frecuencia cercana al 6%, seguido por R tools con el 5% de toda la muestra. Por otro lado, en el enfoque de análisis de sentimientos destaca SentiStrength, la cual es una herramienta altamente utilizada con este fin.

Tabla 4. Clasificación de las herramientas según su enfoque

Enfoque de la MSR	Aplicación	Estudio	Cantidad	Porcentaje
Estudios empíricos	b-Bit Minwise Hashing	E52	1	1.15
	Candoia	E42	1	1.15
	Jhawk	E26	1	1.15
	Ckjm	E26	1	1.15
	Trace Lib	E26	1	1.15
	Weka	E26	1	1.15
	COMMEAN	E51	1	1.15
	DeJEE	E43	1	1.15
	diskcheck	E2	1	1.15
	Dune	E20	1	1.15
	Bugzilla	E16	1	1.15

	Eclipse tools	E16	1	1.15
	Git Mining Tools	E10, E45	2	2.30
	GreenOracle	E22	1	1.15
	JDK Api	E38	1	1.15
	Jira	E32	1	1.15
	LOESS	E54	1	1.15
	monkeylab	E7	1	1.15
	NDK Tools	E41	1	1.15
	Norms Miner	E30	1	1.15
	IBM	E33	1	1.15
	Watson,	E33	1	1.15
	MALLET	E33, E26	2	2.30
	OpenNLP	E33	1	1.15
	Orange	E1	1	1.15
	ORM framework	E27	1	1.15
	R tools	E14, E13, E47	3	3.45
	RefDiff	E53	1	1.15
	RepoGrams	E25	1	1.15
	RevHelper	E50	1	1.15
	RNNLM Toolkit - Deep Learning	E19	1	1.15
	Spencer	E46	1	1.15
	LAMBDADOC	E82	1	1.15
	SNIFFER	E85	1	1.15
	AIMMX	E87	1	1.15
	DECOR tool	E88	1	1.15
	SQLInspect	E88	1	1.15
	UnityLinter	E89	1	1.15
	SoftMon	E90	1	1.15
	Gambit	E91	1	1.15
	X-FLASH	E92	1	1.15
	ANALOG	E93	1	1.15
	ROS	E94	1	1.15
	Dependabot	E95	1	1.15
	Phabricator	E96	1	1.15
	Gerrit	E96	1	1.15
	SoCCMiner	E98	1	1.15
	CLIP	E99	1	1.15
Predicción de cambios	UDC de Eclipse	E6	1	1.15
	FEVER	E24	1	1.15
	Git Mining Tools	E3, E35	2	2.30
	LASE	E23	1	1.15
	MALLET	E15	1	1.15
	WEKA	E47	1	1.15
	Rapid Miner	E47	1	1.15
	UDC de Eclipse	E6	1	1.15
	Watson Libraries for Analysis (WALA)	E12	1	1.15

	CoreNlp	E86	1	1.15
Predicción de errores	Bicho	E34	1	1.15
	CI bot	E34	1	1.15
	Early Warning System	E11	1	1.15
	ExceptionMiner	E8	1	1.15
	Git Mining Tools	E35	1	1.15
	Google code labels	E17	1	1.15
	Jira, Google Issue tracker	E36	1	1.15
	ManifestInspector	E40	1	1.15
	ORM framework	E27	1	1.15
	PLEA Tools	E29	1	1.15
	MUDETECT	E83	1	1.15
	nltk	E84	1	1.15
	scikit-learn	E84	1	1.15
Recomendación de expertos	Mylyn de Eclipse	E5	1	1.15
	PerfImpact	E21	1	1.15
Búsqueda de código	Algoritmos: ST, MoreWithLess, MWL-ST Hybrid, KL	E4	1	1.15
	R tools	E39	1	1.15
	CoDesNPub Miner	E44	1	1.15
	SpreadCluster	E48	1	1.15
	DeepTip	E81	1	1.15
	MATCHA	E97	1	1.15
Análisis de sentimientos	SentiStrength	E18, E31, E49	3	3.45

En referencia a los datos de entrada usado en los estudios analizados acorde a la Figura 7, se determinó que el de mayor uso fue el de código de cuente con el 30%, seguido de estudios que usan otros datos de entrada con el 22% en opciones de desarrolladores como pots, archivos multimedia u otros datos de entrada en la minería de información, con historial de contenidos se encontró el 20% de la muestra y con el 17% el de tipo de reporte de bugs. En los de menos uso se presentaron revisiones con el 6%, seguido de historial d de versiones con el 4%, y, el 2% que usaron código fuente y reporte de bugs en un mismo estudio para optimizar el objetivo de búsqueda con diferentes enfoques.

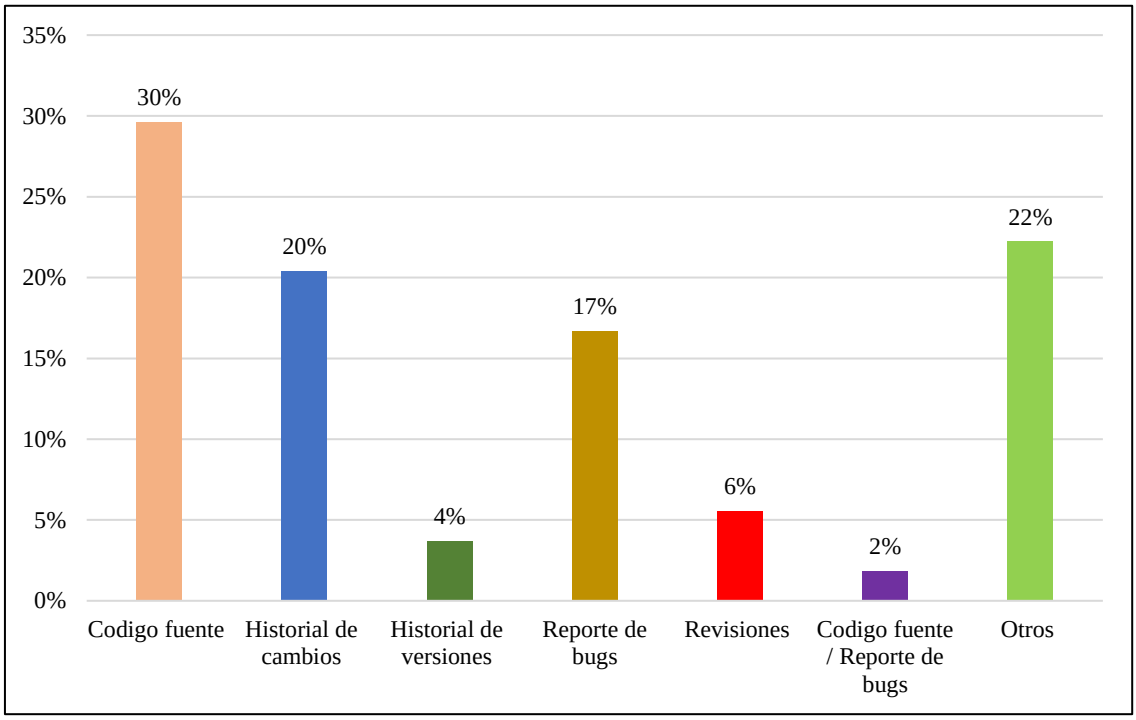


Figura 7. Distribución porcentual de los artículos según sus datos de entrada.

En la clasificación de los estudios según sus datos de entradas y herramientas utilizadas como se refleja en la Tabla 6, se encontró diversificación de la implementación de los datos de entrada para la información de búsqueda en los repositorios de minería de software.

Tabla 5. *Clasificación de las herramientas según sus datos de entrada*

Tipo de datos	Herramientas	Estudios	Cantidad	Porcentaje
Código fuente	Algoritmos: ST, MoreWithLess, MWL-ST Hybrid, KL	E4	1	1.52
	Ckjm, Jhawk, Trace Lib, Weka, Mallet	E26	1	1.52
	DeJEE	E43	1	1.52
	diskcheck	E2	1	1.52
	Git Mining Tools	E45	1	1.52
	JDK Api	E38	1	1.52
	LOESS	E54	1	1.52
	Orange	E1	1	1.52
	PLEA Tools	E29	1	1.52

	Puppet-Lint	E28	1	1.52
	R tools	E13	1	1.52
	Rapid Miner, WEKA	E47	1	1.52
	RepoGrams	E25	1	1.52
	RNNLM Toolkit - Deep Learning	E19	1	1.52
	Spencer	E46	1	1.52
	SpreadCluster	E48	1	1.52
	Gambit	E91	1	1.52
	SoftMon	E90	1	1.52
	UnityLinter	E89	1	1.52
	DECOR tool	E88	1	1.52
	SQLInspect tool	E92	1	1.52
	AIMMX	E87	1	1.52
	CoreNLP	E86	1	1.52
	MUDETECT	E83	1	1.52
	LAMBDADOC	E82	1	1.52
	MATCHA	E97	1	1.52
Historial de cambios	b-Bit Minwise Hashing	E52	1	1.52
	Early Warning System, CI bot	E11	1	1.52
	FEVER	E24	1	1.52
	Git Mining Tools	E3, E10	2	3.03
	GreenOracle	E22	1	1.52
	LASE	E23	1	1.52
	MALLET MAchine Learning for LanguagE Toolkit	E15	1	1.52
	ORM framework	E27	1	1.52
	PerfImpact	E21	1	1.52
	SNIFFER		1	1.52
	Nltk		1	1.52
	Scikit-learn		1	1.52
	Watson Libraries for Analysis (WALA)	E12	1	1.52
Historial de versiones	UDC de Eclipse	E6	1	1.52
	RefDiff	E53	1	1.52
Reporte de bugs	Bicho	E34	1	1.52
	Eclipse tools, Bugzilla	E16	1	1.52
	Git Mining Tools	E35	1	1.52
	Google code labels	E17	1	1.52
	Jira	E32	1	1.52
	Jira, Google Issue tracker	E36	1	1.52
	ManifestInspector	E40	1	1.52
	Mylyn de Eclipse	E5	1	1.52
	Norms Miner	E30	1	1.52
Revisiones	CodeFlow	E9	1	1.52

	RevHelper	E50	1	1.52
	COMMEAN	E51	1	1.52
Código fuente / Reporte de bugs	Candoia	E42	1	1.52
Otros	CoDesNPub Miner	E44	1	1.52
	Dune	E20	1	1.52
	ExceptionMiner	E8	1	1.52
	monkeylab	E7	1	1.52
	NDK Tools	E41	1	1.52
	OpenNLP, IBM Watson, MALLET	E33	1	1.52
	R tools	E14, E37, E39	3	4.55
	DeepTip	E81	1	1.52
	SentiStrength	E18, E31, E41	3	4.55
TOTAL			99	100%

El dato de entrada con mayor porcentaje es el código fuente con un 30%: esto se puede deber por el hecho que del código fuente se puede extraer mucha información, tal como: soluciones a problemas que se presenten (es decir si tengo un problema programando al que no encuentro solución, puede que alguien ya se haya enfrentado a esa situación y se encuentre la respuesta en el código de alguien más), buenas y malas prácticas a la hora de programar, code smells, lenguajes más utilizados en determinadas áreas, entre otros.

El siguiente dato es otros por la gran cantidad de estudios empíricos en los que los datos no se limitan a los tradicionales, sino que también se expanden a posts de Stack Overflow, tickets, archivos multimedia como videos o podcasts, artículos acerca de desarrollo de software, entre otros, en los que la información extraída tiene diferente fines desde analizar los sentimientos del programador, encontrar soluciones a problemas en posts o artículos, a minar archivos de audio para extraer tutoriales para diferentes lenguajes de programación.

Le sigue el historial de cambios por el hecho de que a través de él se puede observar la evolución del programa y al igual que el reporte de bugs, es útil para la predicción de errores/bugs. En menor porcentaje se encuentran las revisiones y el historial de cambios. Las revisiones ayudan a analizar aspectos humanos y observar la recepción de la app o programa frente a los usuarios finales. Y el historial de cambios es útil en la predicción de cambios.

3.2. PI1: Herramientas para la minería de repositorios de software que se han desarrollado durante los últimos años

De los estudios seleccionados, se encontró a través del mapeo que durante el periodo 2015-2022 se han desarrollado 32 nuevas herramientas para la minería de repositorios de software las cuales se muestran en la siguiente tabla junto con su enfoque:

Estudio	Title	Herramienta/Metodología
E12	Unveiling Exception Handling Bug Hazards in Android based on GitHub and Google Code Issues	ExceptionMiner*
E23	Mining Performance Regression Inducing Code Changesin Evolving Software	PerfImpact*
E24	FEVER: Extracting Feature-oriented Changes fromCommits	FEVER*
E29	Understanding the Exception Handling Strategies of Java Libraries: An Empirical Study	PLEA tool*
E30	Externalization of Software Behavior by the Mining of Norms	Norms Miner*
E38	Domain-Specific Cross-LanguageRelevant Question Retrieval	CLRQR*
E39	Candoia: A platform for building and sharing mining software repositories tools as apps	Plataforma Candoia*
E40	Developer mistakes in writing android manifests: An empirical study of configuration errors	ManifestInspector *
E41	Spencer: Interactive heap analysis for the masses	Spencer*
E42	Analyzing program dependencies in Java EE applications	DeJEE*
E43	Who you gonna call?Analyzing Web Requests in Android Applications	Stringoid*
E45	Predicting Usefulness of Code Review Commentsusing Textual Features and Developer Experience	RevHelper*
E47	RefDiff: Detecting Refactorings in Version Histories	RefDiff*
E48	Mining Change Histories forUnknown Systematic Edits	SysEdMiner*
E50	Extracting Build Changes with BUILDDIFF	BUILDDIFF*

E53	Euphony: Harmonious Unification of CacophonousAnti-Virus Vendor Labels for Android Malware	EUPHONY*
E55	CLEVER: Combining Code Metrics with Clone Detection forJust-In-Time Fault Prevention and Resolution in Large IndustrialProjects	CLEVER*
E56	What did Really Change with the new Release of the App?	CALAPPA*
E57	On the impact of security vulnerabilities inthe npm package dependency network	semver-tool
E59	Characterising Deprecated Android APIs	CDA*
E68	Analyzing Requirements and Traceability Information toImprove Bug Localization	TraceScore*
E69	Towards Extracting Web API Specifications from Documentation	D2Spec*
E77	Import2vec: Learning Embeddings for Software Libraries	Import2vec*
E78	STRAIT: A Tool for Automated Software Reliability Growth Analysis	STRAIT*
E81	Extracting API tips from developer question and answer websites	DeepTip
E82	Automatically generating documentation for lambda expressions in Java	LAMBDADOC
E83	Investigating next steps in static API-Misuse detection	MUDETECT
E90	SoftMon: A Tool to Compare Similar Open-source Software from a Performance Perspective	SoftMon
E91	Gambit - An open source name disambiguation tool for version control systems	Gambit
E92	Mining Scientific Workflows for Anomalous Data Transfers	X-FLASH
E93	On the Naturalness and Localness of Software Logs	ANALOG
E97	Recommending Code Improvements Based on Stack Overflow Answer Edits	MATCHA*
E9	Semantic Similarity Metrics for Evaluating Source Code Summarization	SoCCMiner

3.3. PI3: Identificación de las herramientas para MSR más utilizadas en la industria del desarrollo de software

Se consideran las herramientas más utilizadas para la minería de reposito de software, a aquellas que se prestan para múltiples enfoques, incluyendo los estudios empíricos. Herramientas como WEKA, MALLET y R tools, diseñadas para el machine learning, han resultado de gran ayuda en la predicción de cambios, predicción de errores y en estudios experimentales. Además, al ser herramientas altamente conocidas y utilizadas, son las que cuentan con mayor aplicabilidad. Haciendo referencia a la pregunta de investigación número 4.

Como se refleja en la Figura 8, donde se encontró que de los 99 estudios citados el 8% usaron como herramientas de datos el de tipo Eclipse, seguido del 7% R tolos, 5% Senti Strength y MALLET en ambos casos y 4% WEKA. Identificando para la mayor parte son dirigidos a la búsqueda de información e intereses de los consumidores y establecer estrategias de diseños y ventas a partir de ello.

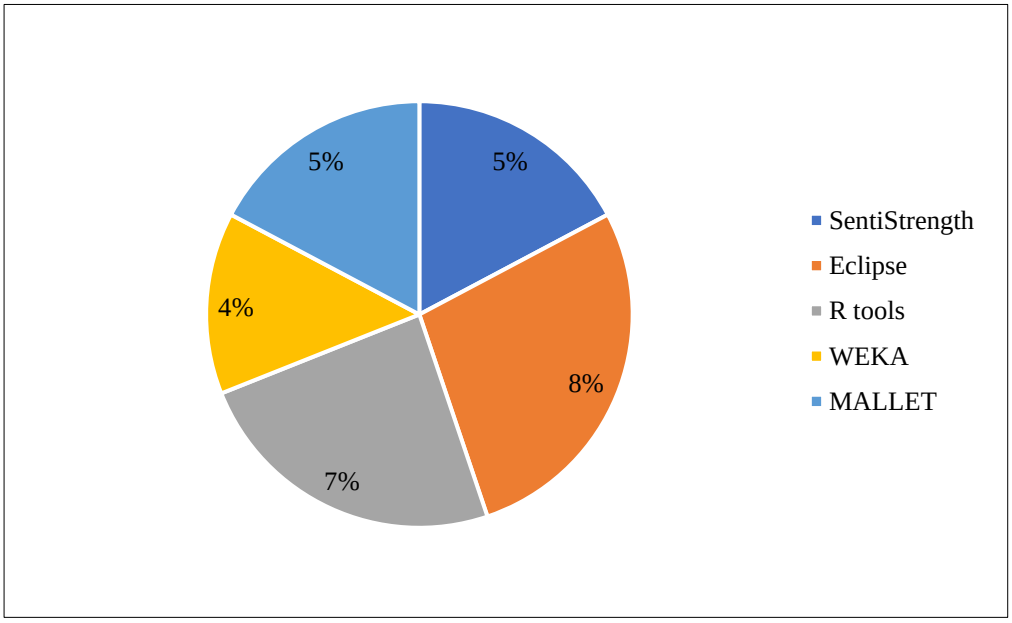


Figura 7. Herramientas con mayor frecuencia de uso en los MRS

3.4. Medición del alcance de la minería de repositorios de software

Las búsquedas iniciales revelaron el panorama en el que se encuentra la investigación en la minería de repositorios de software, como se puede observar en la figura 8, se reciben gran cantidad de artículos relacionados a la temática año a año, siendo el 2016 el año con menos artículos publicados, observando que en los siguientes años las cifras han ido en aumento, solo disminuyendo ligeramente en 2020.

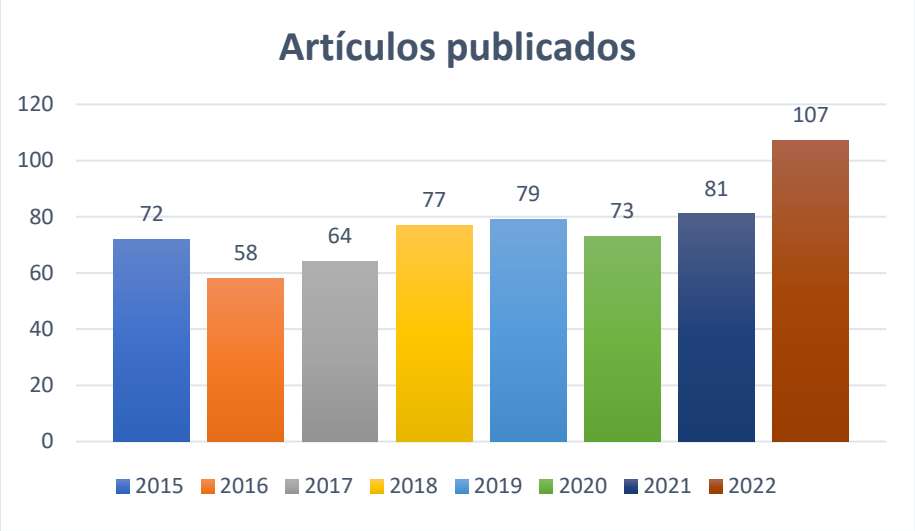


Figura 8. Artículos publicados en los últimos seis años.

También durante este periodo se pudo observar, cuáles eran los países en donde más se dedicaban a la investigación de la minería de repositorio de software, siendo los países más influyentes Estados Unidos y Canadá, participando en 24 y 22 artículos respectivamente dentro de los estudios primarios, otros países destacados son Brasil, Australia, Japón e Italia.

Tabla 6. Países con mayor número de estudios primarios en la investigación

País	Cantidad
Alemania	2
Australia	3
Austria	1
Brasil	5
Canadá	22
China	1
Corea	1
Estonia	2
Francia	2

Grecia	1
Holanda	2
Hong Kong	1
Japón	2
Suecia	1
Suiza	1
Italia	3
Tailandia	1
USA	24

Como se pudo demostrar en el estudio, existen gran cantidad de herramientas utilizadas para MSR, en múltiples enfoques y con distintas características. La minería de repositorios es un campo que está en crecimiento, ya que cada año se están indicando nuevas formas de extraer datos de los repositorios y conjuntos de datos, para darles un valor. Durante el mapeo se pudo observar que la MSR, además de los usos clásicos ya se está utilizando en aplicaciones como: el control de consumo de energía que provoca un software, realizar búsquedas en podcast y vídeos para extraer información necesaria, analizar como la personalidad y los pensamientos afectan el rendimiento de un programador, etc. Esto es solo un indicativo que este campo de IS seguirá creciendo con los años y cada vez se verán usos más impresionantes y se hará más relevantes en el mundo del desarrollo de software.

CAPITULO IV

4. Discusión

La investigación en la minería de repositorios de software es un campo en el cual aún falta mucho por explorar, a pesar de ya no ser tan joven, se pudo evidenciar a través del proceso de mapeo sistemático y por medio de las preguntas de investigación la existencia del desarrollo constante de nuevas herramientas para esta tarea, muchas de las cuales tienen enfoques únicos y totalmente nuevos.

Las herramientas y técnicas aplicadas para la minería de repositorios de software tienen la finalidad de mejorar el proceso de desarrollo, por lo que los investigadores cada año desarrollan nuevas formas de hacerlo, analizando datos como el código fuente, los commits, reportes de bugs, revisiones de código e incluso publicaciones en Stack OverFlow. El mapeo también mostró que existe una tendencia hacia el análisis de sentimientos y como la personalidad o las emociones buenas o malas pueden influir sobre la productividad de una persona. Otra rama que está creciendo es la minería de aplicaciones móviles, en donde incluso se pueden minar la forma en que las personas interactúan con la aplicación, y con los resultados crear interfaces de usuarios más intuitivas y eficientes. La razón de este estudio era conocer las herramientas de MSR más relevantes en la industria y que este resultado sea un indicativo de la situación en la que se encuentra este campo de investigación. En el 2016, De Farias et al [4] concluían que a pesar de la gran variedad de estudios de MSR, todavía hay muchas oportunidades de investigación que deben explorarse, lo cual describe a la situación actual de esta área.

A través de la recolección de datos del presente trabajo, se encontró el uso de diversas herramientas que han considerado los desarrolladores. Siendo los de git Mining Tools con cerca al 6% los de mayor uso el de la muestra, seguido por R tools con el 4% de toda la muestra. Se destacan herramientas como Weka. Mallet, R tools y git mining tools por ser herramientas utilizadas principalmente en la minería de datos y que cuentan con gran cantidad de algoritmos que pueden ser utilizados en diferentes enfoques en la MSR, sobre todo en el empírico y predicción de cambios y errores.

Lo que se relaciona con la investigación desarrollada por Amann et al [6], al coincidir que de la muestra seleccionada el 60% de las investigaciones disponibles eran replicables, en algunos casos debido a la disponibilidad de los datos y en otros por la falta de

información sobre su replicación, además de considerar la curva de popularidad de diferentes tecnologías a lo largo de esa década se identificaron 3 temas que posiblemente serían tendencia en las investigaciones futuras: la minería verde, la ingeniería de software para dispositivos móviles y los aspectos humanos que influyen en el desarrollo de software. Reflejando la importancia de los repositorios de software, que en ocasiones a partir de experiencias previas de desarrolladores permiten el fortalecimiento de las herramientas a través de los predictores de cambios o errores. En relación a ello, la investigación "*Data mining tools and techniques for mining software repositories: a systematic review*" [9], muestra los problemas comunes que existe en el desarrollo de software, las consecuencias que pueden causar estos problemas y cómo la minería de repositorios de software puede ser una solución útil a estos problemas.

Se identificaron como las herramientas de mayor uso para la minería de reposito de software, a aquellas que se prestan para múltiples enfoques, entre ellas la WEKA, MALLET y R tools, diseñadas para el machine learning, han resultado de gran ayuda en la predicción de cambios, predicción de errores y en estudios experimentales. Donde el 8% usaron como herramientas de datos el de tipo Eclipse, seguido del 7% R tolos, 5% Senti Strength y MALLET en ambos casos y 4% WEKA. Lo que se relaciona con el trabajo sobre "*A systematic mapping study of software development with GitHub*" [8], donde se mostró que la mayoría de los trabajos se centraron en la interacción en torno a tareas relacionadas con la codificación y comunidades de proyectos. Identificando para la mayor parte son dirigidos a la búsqueda de información e intereses de los consumidores y establecer estrategias de diseños y ventas a partir de ello.

Por su parte, desde la investigación "*A survey on the use of topic models when mining software repositories*" [7], proponen nuevos desafíos en el desarrollo de software y su relación con los repositorios al identificar que la mayoría de los estudios se centran en un número limitado de tareas de ingeniería de software, muchas de las investigaciones utilizan únicamente *topic models* básicos y pudieron notar que los investigadores no aprovechaban del todo el uso de esta técnica. Presentando oportunidades de crecimiento para el área en donde los datos de minería que se recopilen sean usados en diversas fuentes y estén al uso acorde a diferentes necesidades.

CAPÍTULO V

5. Conclusiones y recomendaciones

Conclusiones

Por medio de este trabajo de investigación se conocieron las herramientas que se utilizan en la minería de repositorios de código actualmente, además, se pudo conocer cuáles son los datos de entradas que son objeto de análisis y los enfoques más comunes en las investigaciones acerca de MSR.

Concluyendo, dado los resultados, que existe una fuerte inclinación de los investigadores hacia proyectos con un enfoque empírico, resaltando que el 51% de las aplicaciones encontradas durante el mapeo, fueron utilizadas en investigaciones con ese propósito. Lo que manifiesta la versatilidad de esta técnica para hallar soluciones dentro del desarrollo de software. Los siguientes enfoques más populares fueron la predicción de errores y la predicción de cambios, siendo el enfoque menos común el análisis de sentimientos.

De igual forma, se pudo observar que los datos de entradas más utilizados son aquellos de tipo estructurado, siendo los más recurrentes el código fuente y el historial de cambios. A pesar de eso, se puede destacar que el mapeo mostró que existe una gran variedad de tipos de datos que pueden ser usados dentro de la MSR, tales como artículos de investigación, archivos de voz y video, entre otras.

Entre las herramientas que se destacaron, estuvieron WEKA, R Tools y MALLET, lo cual se justifica por ser herramientas bastante utilizadas en el campo de análisis de datos, ya que les proporciona la versatilidad de adaptarse a la visión del investigador y a los algoritmos que este desea utilizar.

Por medio de este estudio se demostró que el campo de la MSR cuenta con una gran comunidad que contribuye anualmente con la creación de herramientas e investigaciones que buscan mejorar el proceso de desarrollo de software, pero a pesar de la variedad de estudios, aún hay mucho terreno por explorar, ya que los avances tecnológicos siempre traerán consigo nuevos desafíos en este campo y para quienes lo estudian.

Recomendaciones

Las herramientas de minería de repositorios código son de mucha ayuda para mejorar la calidad del desarrollo del software, por lo que su implementación en proyectos y equipos de trabajo puede ayudar a que estos mejoren y sean mmás agiles. Existe una gran cantidad de herramientas para este propósito, por lo que es imprescindible saber identificar la que se adapte mejor a la tarea a realizar. También es recomendable antes de implementar la MSR en un proyecto, establecer que enfoque se debería seguir y determinar cómo este le puede resultar útil al proyecto.

Como se pudo observar en el presente estudio, WEKA Y R Tools, son de las herramientas más utilizadas en este campo, esto se debe no solo a su versatilidad, sino también a la gran cantidad de documentación que se puede encontrar acerca de estas herramientas, además de poseer una inmensa comunidad de usuarios en ambos casos. Motivos por los cuales se recomienda su uso a la hora de empezar en la minería de repositorios de código e incluso cuando se tenga ya algo de experiencia. El enfoque por implementar depende sobre todo de las problemáticas encontradas en cada proyecto o equipo.

La MSR es un campo de investigación muy interesante ya que muestra cómo se puede extraer información valiosa del código de nuestras aplicaciones, usarlo y estudiarlo puede ayudar a mejorar las habilidades de programación de un individuo o equipo. La implementación correcta de la MSR puede ser sinónimo de mejora en el proceso de desarrollo de software dentro de un equipo.

6. Referencias

6.1. Referencias bibliográficas

[1] R. S. Pressman, *Ingeniería del software: un enfoque práctico*. 2013.

[2] P. Guillermo y L. RINAUDO, *Ingeniería de Software*. Alfaomega Grupo Editor, 2015.

[3] B.-A. Guérin, *Gestión de proyectos informáticos: desarrollo, análisis y control*. Ediciones ENI, 2015.

[4] M. A. de F. Farias, R. Novais, M. C. Júnior, L. P. da Silva Carvalho, M. Mendonça, y R. O. Spínola, «A systematic mapping study on mining software repositories», en *Proceedings of the 31st Annual ACM Symposium on Applied Computing - SAC '16*, Pisa, Italy, 2016, pp. 1472-1479, doi: 10.1145/2851613.2851786.

[5] D. Spadini, M. Aniche, y A. Bacchelli, «PyDriller: Python framework for mining software repositories», en *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2018*, Lake Buena Vista, FL, USA, 2018, pp. 908-911, doi: 10.1145/3236024.3264598.

[6] S. Amann, S. Beyer, K. Kevic, y H. Gall, «Software Mining Studies: Goals, Approaches, Artifacts, and Replicability», en *Software Engineering*, vol. 8987, B. Meyer y M. Nordio, Eds. Cham: Springer International Publishing, 2015, pp. 121-158.

[7] T.-H. Chen, S. W. Thomas, y A. E. Hassan, «A survey on the use of topic models when mining software repositories», *Empirical Software Engineering*, vol. 21, n.º 5, pp. 1843-1919, oct. 2016, doi: 10.1007/s10664-015-9402-8.

[8] V. Cosentino, J. L. Canovas Izquierdo, y J. Cabot, «A Systematic Mapping Study of Software Development with GitHub», *IEEE Access*, vol. 5, pp. 7173-7192, 2017, doi: 10.1109/ACCESS.2017.2682323.

[9] T. Siddiqui y A. Ahmad, «Data Mining Tools and Techniques for Mining Software Repositories: A Systematic Review», en *Big Data Analytics*, vol. 654, V. B.

Aggarwal, V. Bhatnagar, y D. K. Mishra, Eds. Singapore: Springer Singapore, 2018, pp. 717-726.

[10] R.- ASALE y RAE, «software | Diccionario de la lengua española», *«Diccionario de la lengua española» - Edición del Tricentenario*. https://dle.rae.es/software (accedido feb. 14, 2021).

[11] «IEEE Standard Glosary of Software Engineering Terminology». IEEE Standards Board, 1993.

[12] Sommerville Ian, *Ingenieria del Software 7ma. Ed. - Ian Sommerville.pdf*, 7ma ed. Pearson, 2005.

[13] G. Booch, «The History of Software Engineering», *IEEE Software*, vol. 35, n.º 5, pp. 108-114, sep. 2018, doi: 10.1109/MS.2018.3571234.

[14] Brooks, «No Silver Bullet Essence and Accidents of Software Engineering», *Computer*, vol. 20, n.º 4, pp. 10-19, abr. 1987, doi: 10.1109/MC.1987.1663532.

[15] «ISO 25010». https://iso25000.com/index.php/normas-iso-25000/iso-25010?limit=3&limitstart=0 (accedido feb. 15, 2021).

[16] D. Carrizo y A. Alfaro Avalos Loki, «Método de aseguramiento de la calidad en una metodología de desarrollo de software: un enfoque práctico», *Ingeniare. Revista chilena de ingeniería*, vol. 26, pp. 114-129, mar. 2018, doi: 10.4067/S0718-33052018000100114.

[17] M. Callejas-Cuervo, A. C. Alarcón-Aldana, y A. M. Álvarez-Carreño, «Modelos de calidad del software, un estado del arte», *Entramado*, vol. 13, n.º 1, pp. 236-250.

[18] N. N. Zolkifli, A. Ngah, y A. Deraman, «Version Control System: A Review», *Procedia Computer Science*, vol. 135, pp. 408-415, 2018, doi: 10.1016/j.procs.2018.08.191.

[19] «Git - Book». https://git-scm.com/book/es/v2/ (accedido ago. 25, 2020).

[20] «Resumen de Bitbucket | Atlassian». https://www.atlassian.com/es/software/bitbucket/guides/getting-started/overview (accedido ago. 25, 2020).

[21] H. Borges y M. Tulio Valente, «What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform», *Journal of Systems and Software*, vol. 146, pp. 112-129, dic. 2018, doi: 10.1016/j.jss.2018.09.016.

[22] A. Hassan, «Mining Software Repositories to Assist Developers and Support Managers», en *2006 22nd IEEE International Conference on Software Maintenance*, Philadelphia, PA, USA, sep. 2006, pp. 339-342, doi: 10.1109/ICSM.2006.38.

[23] L. Lavazza y L. Santillo, «Historical Data Repositories in Software Engineering: Status and Possible Improvements», en *2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement*, Assisi, Italy, oct. 2012, pp. 221-225, doi: 10.1109/IWSM-MENSURA.2012.39.

[24] K. K. Chaturvedi, V. B. Sing, y P. Singh, «Tools in Mining Software Repositories», en *2013 13th International Conference on Computational Science and Its Applications*, Ho Chi Minh City, Vietnam, jun. 2013, pp. 89-98, doi: 10.1109/ICCSA.2013.22.

[25] S. F. Ochoa *et al.*, «PANORAMA DE LA CIENCIA DE LA COMPUTACIÓN EN CHILE», p. 92.

[26] V. Rainardi, *Building a Data Warehouse: With Examples in SQL Server*. Apress, 2007.

[27] R. R. D' Ambros Marco, «Effective Mining of Software Repositories», 2011, doi: 10.1109/icsm.2011.6080839.

[28] Barbara Ann Kitchenham y Stuart Charters, «Guidelines for performing Systematic Literature Reviews in Software Engineering», *EBSE 2007-001*, pp. 5-8, 2007.

[29] K. Petersen, R. Feldt, S. Mujtaba, y M. Mattsson, «Systematic Mapping Studies in Software Engineering», presentado en 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), jun. 2008, doi: 10.14236/ewic/EASE2008.8.

[30] M. A. A. Guamán, «MAPEO SISTEMATICO DE LITERATURA DE UN DATA», p. 17.

[31] J. P. R. Ibañez, «Estudio de Mapeo Sistemático sobre Arquitecturas de Software para Big Data», p. 14.

[32] B. A. Kitchenham, D. Budgen, y O. Pearl Brereton, «Using mapping studies as the basis for further research – A participant-observer case study», *Information and Software Technology*, vol. 53, n.º 6, pp. 638-651, jun. 2011, doi: 10.1016/j.infsof.2010.12.011.

[33] «Reference Manager and Academic Social Network - Mendeley Database | Elsevier Solutions». https://www.elsevier.com/solutions/mendeley (accedido oct. 15, 2020).

[34] «¿Qué es Mendeley?» http://www.cobdc.net/gics/?p=1057 (accedido oct. 15, 2020).

[35] «EndNote», *Wikipedia, la enciclopedia libre*. ene. 20, 2020, Accedido: oct. 15, 2020. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=EndNote&oldid=122921589.

[36] «Product Details», *EndNote*. https://endnote.com/product-details/ (accedido oct. 16, 2020).

[37] P. Martínez, «LibGuides: Gestores Bibliográficos: EndNote Basic». https://uandes.libguides.com/gestoresbibliograficos/endnote (accedido oct. 16, 2020).

[38] «ISI Web of Knowledge», *Wikipedia, la enciclopedia libre*. jul. 02, 2020, Accedido: oct. 16, 2020. [En línea]. Disponible en:

https://es.wikipedia.org/w/index.php?title=ISI_Web_of_Knowledge&oldid=127430574.

[39] «Zotero | Bibliotecas Universidad de Salamanca». https://bibliotecas.usal.es/zotero (accedido oct. 16, 2020).

[40] «Zotero», *Wikipedia, la enciclopedia libre*. ago. 06, 2020, Accedido: oct. 16, 2020. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Zotero&oldid=128296521.

[41] J. Puckett, *Zotero: A Guide for Librarians, Researchers, and Educators*. Assoc of Cllge & Rsrch Libr, 2011.

[42] «RefWorks». https://www.refworks.com/es/faqs.asp?l=spanish#q1 (accedido oct. 16, 2020).

[43] G. mantenidas por C. UC3M, «Índice: Guía temática sobre citas bibliográficas UC3M: RefWorks». https://uc3m.libguides.com/guias_tematicas/citas_bibliograficas/refworks (accedido oct. 16, 2020).

[44] «Software de análisis de datos cualitativos - QDA Miner», *Provalis Research*. https://provalisresearch.com/es/products/software-de-analisis-cualitativo/ (accedido oct. 16, 2020).

[45] «Constitución de la Republica del Ecuador», p. 218.

[46] «Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación», p. 113.

[47] «Ley de Propiedad Intelectual Registro Oficial 320», n.º 320, p. 92.

[48] Arias G. Fideas, *EL PROYECTO DE INVESTIGACIÓN.pdf*, 5ta ed. 2006.

[49] C. Wohlin, «Guidelines for snowballing in systematic literature studies and a replication in software engineering», en *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, London, England, United Kingdom, 2014, pp. 1-10, doi: 10.1145/2601248.2601268.

6.2. Anexos

Anexo 1. Tabla de extracción de estudios primarios

Autor	Título	Base de datos/conferencia	Palabras claves	Resumen	Año	URL	DOI

Anexo 2. Tabla de extracción de estudios seleccionados

Información
Autor
Título
Base de datos/Conferencia
Palabras claves
Resumen
Año
URL

DOI

Citas

Puntuación

Tipo de información de entrada

Tipo de información de salida

Método/Técnica

Aplicación

Tipo de publicación

Tipo de investigación

País

Información característica del proyecto

Defectos

Comentarios