Empirical Studies conducted with software collections: a mapping study

Juan Andrés Carruthers1, Jorge Andrés Díaz-Pace2 y Emanuel Agustín Irrazábal1
1 Universidad Nacional del Nordeste, Departamento de Informática, Corrientes, Argentina
{jacarruthers, eairrazabal}@exa.unne.edu.ar
2 Universidad Nacional del Centro de la Provincia de Buenos Aires, Instituto Superior De Ingeniería Del Software, Buenos Aires, Argentina
adiaz@exa.unicen.edu.ar

Abstract. Software projects are inputs in Evidence-Based Software Engineering, although they are selected without following a specific strategy, which decreases the generalization and replication of the results. One option is to use collections of existing projects, but these must have explicit construction and maintenance rules. The objective of this work was to perform a secondary study on software project selection strategies in empirical studies, and learn about: the rules considered, project characteristics, code metrics, extraction tools and statistical analyses employed. A systematic mapping was used to identify articles from January 2013 to December 2021. We selected 150 studies in which 67% used their own rules for project selection and 31% worked with existing collections, and the majority (80%) used Java projects. Furthermore, there was no evidence of a standardized framework for project selection for empirical studies in Software Engineering.

Estudios empíricos realizados con colecciones de proyectos software: un mapeo sistemático

Resumen. Los proyectos software son insumos en la Ingeniería del Software Basada en Evidencias, aunque estos sean seleccionados sin seguir una estrategia específica, lo cual disminuye la generalización y replicación de los resultados. Una opción es usar colecciones de proyectos existentes, pero estas deben contar con reglas explícitas de construcción y mantenimiento. El objetivo de este trabajo fue realizar un estudio secundario sistematizado sobre las estrategias de selección de los proyectos software en estudios empíricos, y conocer: las reglas consideradas, las características de los proyectos, las métricas de código, las herramientas de extracción y los análisis estadísticos practicados. Se utilizó un mapeo sistemático para identificar artículos desde enero de 2013 a diciembre de 2021. Se seleccionaron 150 estudios de los cuales el 67% utilizó reglas propias para la selección de los proyectos y el 31% trabajó con colecciones existentes, y la mayoría (80%) empleó proyectos Java. Asimismo, no se encontraron evidencias de un marco estandarizado para la selección de proyectos para estudios empíricos en Ingeniería de Software.

Received January 2023 Accepted April 2023; Published May 2023
https://doi.org/10.24215/15146774e006

Palabras Clave: Colecciones, Proyectos software, Ingeniería del Software Basada en Evidencia.

1 Introducción

El desarrollo software actual trabaja con la construcción de aplicaciones multi versión 1 que crecen, tanto en complejidad como en funcionalidad, siendo necesario conservar su calidad actual 2. Por ello, es necesario obtener métodos empíricos para demostrar la calidad del producto software 3, utilizando evidencia directamente relacionada por medio de mediciones que se vinculen con los atributos de calidad del código fuente 4.

En este sentido, la Ingeniería del Software Basada en Evidencia (ISBE) proporciona los medios para que la evidencia actual de la investigación pueda integrarse con la experiencia práctica y los valores humanos en el proceso de toma de decisiones para el desarrollo y mantenimiento de software 5. En el caso de los estudios empíricos cuyo objeto de estudio es el código fuente, una colección ad hoc de proyectos software a veces no es suficiente para lograr representatividad, ni replicación de los resultados. Con la intención de mejorar los resultados surgen los "datasets" o colecciones de proyectos software junto con los datos extraídos de estos proyectos, utilizados para reducir el costo de recopilar los proyectos software y para que los estudios sean replicables y comparables 6.

Estas colecciones son un insumo para los grupos de trabajo y sirven como mecanismo de comparación para los estudios en ISBE. Por ejemplo, las colecciones construidas por Barone y Sennrich 7, Allamanis y Sutton 8 o Keivanloo 9; se diferencian por la cantidad y calidad de los proyectos, como también los criterios y métodos de agrupamiento. Otras colecciones seleccionan los proyectos considerando el tipo de software; como bibliotecas y frameworks de pruebas 10, o frameworks de bases de datos 11.

Sin embargo, en muchos casos las reglas para la selección de los proyectos no son explícitas. La definición de un modelo de procedimientos a partir de reglas estándar y herramientas es fundamental para garantizar la capacidad de preservar sistemáticamente la colección a lo largo del tiempo por integrantes del mismo equipo de trabajo o externos, es decir, que la conservación de la misma sea independiente del grupo que la construyó. Un ejemplo de esta tendencia es la colección Qualitas corpus 6, cuya última versión es del año 2013 lo cual disminuye su vigencia. De los 112 proyectos de esta colección 51 ya no son mantenidos y el resto tienen un promedio de 58 cambios de versión.

Por lo antes indicado, el objetivo de este trabajo es determinar los criterios de selección y las características de los proyectos software que son insumos de estudios en ISBE, las métricas extraídas del código fuente, las herramientas de extracción y los estudios estadísticos utilizados. Se eligió el enfoque de mapeo sistemático para obtener una visión general del desarrollo técnico actual o el nivel de práctica de un área de investigación 12. Este estudio pretende brindar una visión general sobre las prácticas seguidas por los grupos de investigación para la ejecución de estudios en ISBE con colecciones de proyectos, exponiendo los problemas encontrados que comprometen a la representatividad de las muestras, la replicación y la generación de los resultados.

Además de esta introducción, el trabajo se encuentra organizado de la siguiente manera. La Sección 2 describe antecedentes de la temática. En la Sección 3 se detalla la metodología empleada para el estudio y se presentan las preguntas de investigación. En la Sección 4 se reportan los resultados obtenidos. En la Sección 5 se discuten y se responden las preguntas de investigación. En la Sección 6 se listan las amenazas a la validez. Finalmente, en la Sección 7 se incluye la conclusión del mapeo sistemático.

2 Antecedentes

El uso de proyectos software para realización de estudios empíricos no resulta una práctica novedosa en ISBE. En 1971, Knuth 13 recolectó aleatoriamente programas en FORTRAN de la Universidad de Stanford y de la compañía Lockheed Missiles and Space. Chidamber y Kemerer 14 desarrollaron un trabajo para validar las métricas orientadas a objetos creadas por los mismos autores tres años antes 15, una parte del mismo buscaba demostrar la factibilidad de estas métricas en dos sistemas comerciales: uno codificado en el lenguaje C++ y el otro en Smalltalk. Harrison et al. 16 condujeron un estudio con cinco proyectos software para evaluar y comparar la métrica "acoplamiento entre objetos" y la métrica "número de asociaciones entre una clase y sus pares".

El advenimiento de plataformas para compartir código como SourceForge, aumentó la accesibilidad a proyectos de fuente abierta 6. Con estos nuevos repositorios públicos fue posible el acceso al código fuente versionado de proyectos software de distintos tamaños, tecnologías o perfiles de trabajo de los equipos. Esto hizo más necesaria la construcción de colecciones de proyectos software que aumente la replicabilidad de los experimentos, se agreguen los resultados, se reduzcan los costos y se mejore la representatividad de las muestras.

En la literatura existe una gran variedad de estas colecciones que también reciben el nombre de datasets. Barone y Sennrich 7 crearon una colección de más de 100.000 funciones en Python con sus respectivos cuerpos y descripciones. Allamanis y Sutton 8 construyeron el "Github Java Corpus" con todos los proyectos Java en Github Archive que no fuesen repositorios duplicados. Similar al anterior, Keivanloo et al. 9 incluyeron 24.824 proyectos Java, alojados en SourceForge y Google Code.

Zerouali y Mens 10 actualizaron el Github Java Corpus y analizaron el uso de bibliotecas y frameworks de pruebas como JUnit, Spring o TestNG. En esta actualización agregaron los repositorios creados en Github no presentes en la colección original y descartaron aquellos que no estuvieran disponibles y los que no utilizaron la herramienta Maven para gestionar las dependencias, teniendo como resultado una colección de 4532 proyectos. De igual manera, Goeminne y Mens 11 realizaron cambios al Github Java Corpus y estudiaron cinco frameworks de base de datos.

Tempero et al 6 en el año 2013 construyeron la colección Qualitas Corpus donde incluyeron sistemas desarrollados en Java con sus archivos fuente y binarios disponibles públicamente en formato ".jar". El objetivo del Qualitas fue reducir sustancialmente el costo de los equipos de investigación para desarrollar grandes estudios empíri-

cos del código fuente. Partiendo del Qualitas Corpus, Qualitas.class 17 contiene medidas relativas a los proyectos tales como métricas de tamaño (cantidad de líneas de código, número de paquetes, clases e interfaces) y métricas de diseño.

En algunas colecciones, además de la documentación, código fuente, métricas, y código binario del proyecto, también se proporcionan meta-datos relacionados. Por ejemplo, FLOSSmole 18 brinda los nombres de los proyectos, lenguajes de programación, plataformas, tipo de licencia, sistema operativo y datos de los desarrolladores involucrados. FLOSSMetrics 19 calcula, extrae y almacena información de sistemas de control de versiones, sistemas de rastreo de defectos, archivos de listas de correos y métricas de código. Ejemplos más recientes, como las colecciones de Gyimesi et al. 20 y Chavez et al. 21 reunieron proyectos Java con sus métricas orientadas a objetos (falta de cohesión entre métodos, acoplamiento entre objetos, peso de métodos por clase, entre otras).

Es evidente la diversidad de estrategias existentes para el muestreo de proyectos, demostrando la ausencia de lineamientos metodológicos uniformes. En este estudio buscamos determinar los criterios y características de proyectos software considerados para realizar estudios en ISBE. A su vez también reportar las métricas de código fuente extraídas de los proyectos, las herramientas utilizadas y los estudios estadísticos practicados. De esta manera, evaluaremos los factores que consideramos perjudiciales en la replicación y representatividad de los estudios en ISBE, y realizaremos recomendaciones para la selección de proyectos.

3 Metodología

Se llevó a cabo un estudio de mapeo sistemático siguiendo las pautas identificadas en 22 para obtener una visión general del uso de colecciones de proyectos en la ISBE. Se seleccionó esta técnica por centrarse en la “clasificación y análisis temático de un tema de la Ingeniería del Software” 23. En este caso, aunque la recolección de proyectos sea una práctica estándar para estudios en ISBE, los criterios de selección de los proyectos software, sus características, como también las métricas del código fuente extraídas, las herramientas utilizadas para ello y los análisis estadísticos ejecutados no han sido abordados ampliamente por otros estudios secundarios. Por lo tanto, es necesario identificar, categorizar y analizar la investigación disponible sobre el tema para describir las prácticas y obtener una visión general de su estado de arte.

Las preguntas de investigación que guiaron el desarrollo del estudio se presentan en la **¡Error! No se encuentra el origen de la referencia.**.

Table 1. Preguntas de investigación.

Pregunta de investigación	Motivación
RQ1: ¿Cuáles son los criterios de selección de los proyectos software objeto de estudios empíricos?	Identificar cuáles son los criterios tenidos en cuenta por los investigadores para la selección de proyectos en la realización de estudios empíricos.
RQ2: ¿Con qué tipo de proyectos trabajan los grupos de investigación para realizar estudios empíricos?	Conocer qué características tienen los proyectos seleccionados en términos del tipo de software y lenguaje de programación.
RQ3: ¿Cuáles son las métricas del código que se extraen de los proyectos?	Determinar las métricas obtenidas del análisis estático del código de cada uno de los proyectos seleccionados.
RQ4: ¿Qué herramientas se utilizan para obtener los datos del código?	Determinar cuáles son las herramientas utilizadas para recolectar las métricas y otros artefactos del código.
RQ5: ¿Qué análisis estadísticos se realizan con los datos recopilados?	Definir los análisis estadísticos practicados sobre los proyectos y datos extraídos para interpretar los resultados obtenidos.

3.1 Estrategia de búsqueda

Para reunir los estudios primarios relevantes se llevó a cabo una búsqueda y selección iterativa en tres fases tal y como se indica en la **¡Error! No se encuentra el origen de la referencia.**.

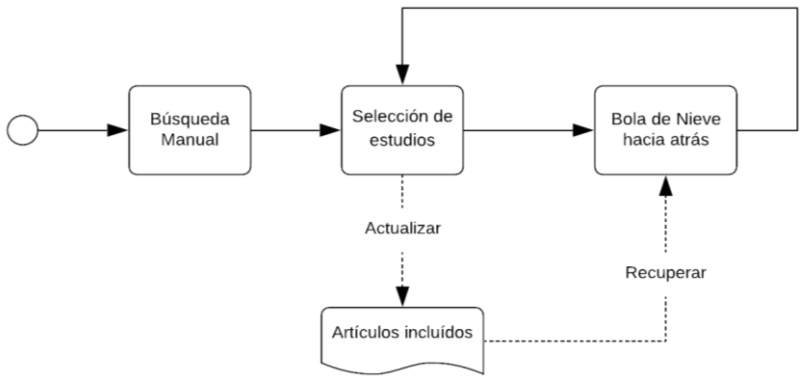

Fig. 1. Proceso de búsqueda y selección de artículos.

- Búsqueda manual: se realizó una búsqueda manual en las principales conferencias y revistas enfocadas en ISBE. Se seleccionó la revista Empirical Software Engineering (EMSE) y las conferencias Empirical Software Engineering and Measurement (ESEM) e International Conference on Evaluation and Assessment in Software Engineering (EASE) por ser representativas del área de investigación y haber sido utilizadas en otros estudios 2425. De acuerdo a estrategias planteadas como las de Zhang et al. 24 y Storey et al. 26, conjuntamente con las buenas prácticas de búsquedas manuales 22, en primera instancia se seleccionó el período de tiempo comprendido entre el 1 de enero del 2013 hasta el 30 de noviembre del 2020 de EMSE y ESEM. Posteriormente se complementó el rango de búsqueda hasta el 31 de diciembre del 2021 conforme se fueron publicando nuevos números de la revista

y el congreso. Después se incorporaron las ediciones del congreso EASE como Molléri et al. 25 considerando el mismo período de tiempo.
En particular, el instrumento de recolección fue una hoja de cálculo, donde se escribieron los nombres y autores de los artículos de la revista y las conferencias.

- Selección de estudios: esta fase consistió en una revisión dual que se realizó de forma iterativa. Los artículos candidatos, después de ser recopilados se incluyeron o excluyeron de acuerdo con los criterios presentes en la **¡Error! No se encuentra el origen de la referencia.**. Los trabajos fueron analizados considerando resumen, introducción, metodología, resultados y conclusión. En cada iteración, se seleccionaron 15 estudios al azar que fueron revisados por dos investigadores. Los mismos anotaron sus decisiones de incluir o excluir cada estudio, junto con el CI o CE en que se basó la decisión. Para medir el grado de acuerdo entre los investigadores se utilizó el estadístico Kappa de Cohen 2728. El valor registrado de Kappa de Cohen fue 0.77, lo que demuestra un nivel de acuerdo alto.
- Muestreo "bola de nieve": se complementó la búsqueda manual con el método de bola de nieve hacia atrás con un muestreo de los artículos incluidos. Las referencias proporcionaron artículos relacionados o similares. Aquellos artículos recopilados durante esta etapa también se agregaron a la lista de candidatos y se seleccionaron de acuerdo con los criterios descritos anteriormente. Este proceso se realizó en dos iteraciones.

Table 2. Criterios de inclusión y exclusión de estudios.

ID	Descripción
CI1	El artículo es un estudio primario
CI2	Es un artículo completo.
CI3	En el artículo se realizan estudios empíricos.
CI4	Los estudios empíricos se realizan en base a la selección de un conjunto de proyectos software.
CE1	Artículos no técnicos (guías, artículos de introducción, editoriales, etc.)
CE2	Artículos duplicados.

3.2 Proceso de selección de artículos

De acuerdo a la estrategia de busque definida en la Subsección 3.1 se realizó una búsqueda manual en los foros EMSE, ESEM y EASE donde fueron recolectados 1709 artículos; de los cuales se incluyeron 1610 y excluyeron 99 según los criterios de exclusión. De los 1610 incluidos en la fase 2, 1480 fueron rechazados por no cumplir alguno de los criterios de inclusión. A los 130 artículos aceptados se le sumaron 20 en la fase 4, quedando finalmente la suma de 150. Los resultados se observan en la **¡Error! No se encuentra el origen de la referencia.**.

Table 3. Detalle de cada fase de selección de estudios primarios.

Fase	Descripción	Incluidos	Excluidos
1	Búsqueda manual	1709	-
2	Selección por criterios de exclusión	1610	99
3	Selección por criterios de inclusión	130	1480
4	Bola de Nieve hacia atrás	20	-

3.3 Extracción de datos

Se utilizó un cuestionario de extracción de datos basado en las preguntas de investigación para recopilar la información relevante de los estudios primarios. La extracción fue realizada por dos investigadores y revisada por un tercero, comprobando la información presente en el formulario con cada uno de los artículos para verificar la consistencia. Los datos recolectados incluyeron información general (título, autores, año de publicación y fuente) e información relativa a las preguntas de investigación (RQ1 – RQ5), como se ilustra en la **¡Error! No se encuentra el origen de la referencia.**.

La información se extrajo exactamente como los autores la mencionan en los artículos y los conflictos se discutieron y resolvieron internamente por los investigadores. Se adoptó este enfoque para evitar la subjetividad y facilitar la replicación del estudio.

Para dar soporte a este proceso se utilizó la herramienta FRAMEndeley 32, una extensión que permite codificar con tablas el texto resaltado en el gestor de referencias Mendeley valiéndose de los colores de subrayado para distinguir y extraer fragmentos de texto relevantes. En este caso se le asignó un color identificativo a cada RQ.

Previo a la extracción, se realizó una prueba piloto con 15 artículos para calibrar el instrumento de extracción, refinar la estrategia y evitar diferencias entre los investigadores.

Table 4. Formulario de extracción de datos.

	ID	Item	Descripción
General	D1	Título	Título del estudio primario
	D2	Autores	Autores del estudio primario
	D3	Año de Publicación	Año de publicación del estudio primario
	D4	Fuente	Fuente donde se publicó el estudio primario
RQ1	D5	Criterio	Criterio de selección de los proyectos utilizados en el estudio primario
RQ2	D6	Lenguaje	Lenguaje de programación utilizado en los proyectos seleccionados
	D7	Tipo de proyecto	Tipo de proyecto de acuerdo a su funcionalidad
RQ3	D8	Métricas del código	Métricas extraídas del código de los proyectos seleccionados en los estudios primarios
RQ4	D9	Herramientas	Herramientas para la recolección de métricas y artefactos del código.
RQ5	D10	Análisis	Análisis estadísticos conducidos en los proyectos y datos extraídos

3.4 Análisis de los datos

Las respuestas a las RQ fueron analizadas de forma cualitativa mediante la codificación abierta de los textos encontrados en los estudios primarios recolectados 33. La selección de esta estrategia se basó en la realizada por Janssen y Van der Voort 34. El primer y segundo autor realizaron el proceso de forma separada e identificaron los desacuerdos, que se resolvieron sumando un tercer investigador que visitaba nuevamente la fuente de información y tenía en cuenta las justificaciones dadas por los dos integrantes originales.

Como siguiente paso se utilizó la codificación cerrada 35 para identificar y resignificar las respuestas a las taxonomías encontradas. Nuevamente los dos primeros autores del estudio realizaron la codificación inicial y los desacuerdos fueron resueltos mediante la presentación a un tercer investigador. En la sección 4 de análisis y resultados se pueden encontrar las descripciones de las clasificaciones empleadas.

4 Análisis y resultados

En esta sección, se presentan los datos recopilados para cada pregunta de investigación. Los mismos se organizaron y resumieron en tablas y gráficos para una mejor visualización. Es necesario mencionar que existen preguntas con categorías no mutuamente excluyentes, es decir, los artículos pueden figurar en más de una; por ello al sumar los porcentajes puede dar un valor superior a 100.

4.1 RQ1: Criterios de selección

Los estudios recolectados fueron clasificados de acuerdo a la estrategia de selección escogida teniendo en cuenta dos enfoques, por un lado la estrategia general y por otro los criterios de selección específicos. En la primera clasificación (ver **¡Error! No se encuentra el origen de la referencia.**), que abarca 3 categorías, el 67% de los artículos optaron por un método de selección propio, el 31% se basaron en una colección existente o subgrupo de esta y el 2% utilizaron un mecanismo de aleatorización para seleccionar los proyectos.

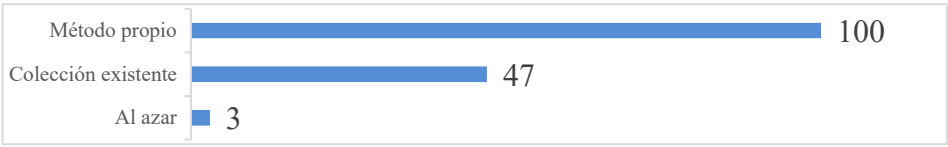

Fig. 2. Método utilizado para elegir los proyectos.

Para la segunda clasificación (ver **¡Error! No se encuentra el origen de la referencia.**) se definieron 13 categorías. La clasificación fue elaborada en función de los datos recopilados tal y como se indica en la Sección 3.4. En total, se extrajeron datos de 113 estudios primarios.

Table 5. Criterios tenidos en cuenta para seleccionar los proyectos.

#	Criterios	Artículos
46	Específicas del caso de estudio del articulo	P8 P12 P13 P15 P23 P32 P34 P45 P48 P50 P51 P58 P61 P62 P64 P65 P66 P68 P69 P71 P73 P75 P76 P79 P82 P83 P89 P96 P98 P99 P102 P109 P110 P111 P114 P128 P131 P132 P133 P134 P136 P139 P141 P142 P144 P148 P149
40	Actividad en el proyecto a lo largo del tiempo	P1 P7 P9 P20 P27 P32 P34 P39 P42 P47 P49 P50 P54 P56 P59 P61 P68 P71 P77 P82 P86 P88 P90 P97 P111 P114 P115 P118 P125 P128 P129 P131 P132 P133 P139 P141 P142 P148 P149 P150
37	Proyectos y meta-datos disponibles públicamente	P1 P6 P8 P10 P14 P21 P26 P31 P35 P36 P39 P43 P46 P47 P50 P56 P64 P65 P68 P69 P72 P73 P74 P75 P76 P77 P78 P83 P94 P95 P98 P114 P123 P125 P139 P142 P144
32	Tamaño	P6 P7 P11 P14 P15 P17 P34 P35 P42 P54 P58 P61 P65 P70 P72 P75 P88 P95 P99 P100 P102 P104 P114 P118 P125 P128 P129 P130 P131 P141 P146 P148
31	Popularidad	P6 P7 P8 P20 P36 P37 P39 P43 P45 P56 P61 P65 P66 P68 P72 P73 P74 P77 P78 P79 P89 P91 P92 P100 P101 P115 P133 P134 P141 P148 P149
26	Dominio o tipo de software	P3 P7 P8 P9 P13 P14 P18 P22 P30 P34 P38 P42 P43 P44 P45 P55 P60 P70 P72 P92 P93 P100 P104 P115 P129 P130
23	Usan herramientas que dan soporte a procesos	P27 P51 P59 P63 P65 P67 P68 P71 P75 P88 P90 P92 P94 P97 P101 P102 P132 P133 P134 P139 P141 P142 P144
15	Actividad reciente en el momento de recolección	P1 P26 P27 P39 P54 P69 P74 P77 P78 P97 P98 P102 P133 P134 P139
15	Disponibilidad Información de defectos	P1 P46 P47 P50 P54 P73 P79 P94 P95 P111 P125 P128 P132 P141 P144
11	Calidad	P6 P15 P19 P48 P58 P73 P76 P109 P142 P144 P147
10	Disponibilidad datos históricos	P1 P3 P7 P32 P56 P90 P115 P129 P130 P142
9	Equipo de Desarrollo	P6 P32 P35 P39 P56 P73 P114 P118 P148
6	Documentación	P6 P7 P61 P73 P98 P128

De esta manera, el 41% de los estudios describieron criterios específicos del caso de estudio por el cual se realizó estudio empírico. Por ejemplo, repositorios cuyo primer commit no pareciera una migración [P8], proyectos que incluyan una matriz que indique las fallas que cubren los tests [P12], sistemas de fuente abierta que sean similares a soluciones industriales [P13], entre otros.

El 35% consideraron la actividad del proyecto a lo largo del tiempo como factor de selección. La actividad puede ser medida a través de la cantidad de años de desarrollo o cantidad de commits. Así se encuentran casos en que se selecciona proyectos con 3 o más años [P1], proyectos con más de 10K commits [P71], o sistemas que tengan más de 32 commits y un año de desarrollo [P50].

El 33% de los estudios construyeron la colección de proyectos seleccionando aquellos que estuvieran disponibles públicamente con sus meta-datos asociados. Se mencionaron sitios de alojamiento de proyectos software como: SourceForge [P1, P26, P94], ohloh.net (ahora Black Duck Open Hub) [P8], Squeaksource [P10], Google Play [P31, P43], Github [P36, P68, P114] o Apple Store [P43], entre otros.

El 28% tuvieron en cuenta el tamaño de los proyectos cuantificado en términos de clases [P6, P14, P42], módulos [P17]; líneas de código [P75, P99, P118], puntos de función IFPUG [P15] o puntos de función FiSMA [P58]. El 27% eligieron la popularidad en repositorios públicos como criterio de recolección. Dependiendo del trabajo se

tomaron enfoques diferentes para cuantificarla. Por ejemplo, en términos de: el número de descargas [P6]; la cantidad de usuarios [P7, P45, P56], el número de visitas [P36], el número de duplicaciones del repositorio [P39] o la cantidad de reseñas de usuarios [P43], entre otras.

En el 23% de los estudios primarios consideraron como criterio el dominio o tipo de software, es decir, la funcionalidad o ámbito de aplicación del mismo. En su mayoría (23 artículos) establecen que los sistemas seleccionados deben provenir de múltiples dominios de aplicación. En los tres casos restantes recolectaron proyectos con dominios específicos, como bases de datos [P9], aplicaciones de propósito general [P45], y sistemas de un banco comercial en China [P18].

En el 20% de los casos la condición fue el uso de herramientas para dar soporte a procesos dentro del desarrollo del proyecto. Esto incluyó procesos tales como: gestión de dependencias [P27, P68, P88], gestión de versiones [P63, P71, P97] o revisión de código [P51, P59].

El 13% buscaron proyectos en los que se haya registrado actividad reciente al momento de recolección, es decir, que los mismos sigan siendo mantenidos o actualizados por sus colaboradores. La actividad suele ser medida en base a la cantidad de commits realizados en un periodo de tiempo [P98]. En la misma proporción, consideraron la disponibilidad de información de defectos en sistemas de rastreo de defectos.

La calidad del proyecto fue considerada en el 10% de los casos. Se utilizaron distintas formas para establecerla, a partir de: el diseño y metodología de desarrollo del proyecto [P48], herramientas como Reaper [P73, P76, P109], y escalas de calidad como ISBSG [P15]. El 9% buscaron aquellos proyectos en los que existiera datos históricos del proceso de desarrollo, en algunos casos para recuperar información evolutiva del sistema.

En el 8% de los artículos recolectaron proyectos según la cantidad de participantes en el equipo de desarrollo o si existiera una comunidad que ofreciera soporte. Finalmente, la documentación del proyecto fue la condición en el 5% de los casos, en medios como los comentarios en el código fuente [P6, P61, P98], documentación de desarrollo [P7]

4.2 RQ2: Características de los proyectos

Cada uno de los proyectos software en las colecciones utilizados en los estudios seleccionados fueron clasificados según el lenguaje de programación principal (**¡Error! No se encuentra el origen de la referencia.**) y el tipo de software (**¡Error! No se encuentra el origen de la referencia.**). En total se registraron 138 artículos que describieron los lenguajes de programación principales de los proyectos. En un 80% fueron construidos en Java, seguido por C con un 25% y C++ con un 20%.

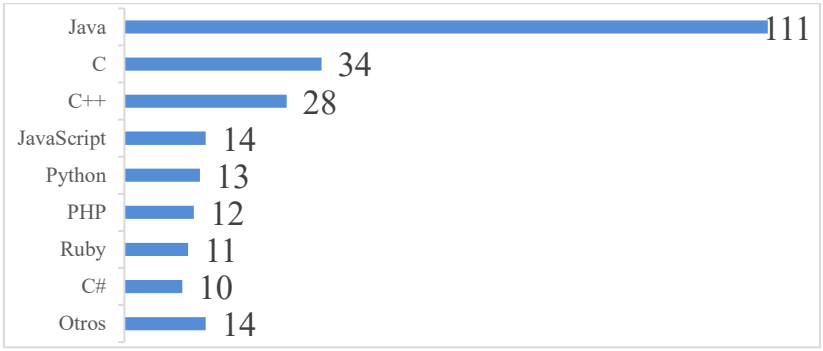

Fig. 3. Lenguajes de programación de los proyectos.

Para identificar los tipos de software se buscaron los nombres y descripciones de proyectos mencionadas en los estudios. En los casos donde los datos extraídos no eran suficientes para determinar el tipo de software utilizado, se realizó una búsqueda y consulta manuales. Cada proyecto fue clasificado según la taxonomía publicada por Forward y Lethbridge 37, de donde se emplearon cinco categorías de segundo nivel: diseño y construcción de software, servidor, redes y comunicaciones, sistema operativo y sistema embebido (ver **¡Error! No se encuentra el origen de la referencia.**). Para comprender las demás categorías dentro de la taxonomía original que abarcan al software guiado por datos, orientado al consumidor y de propósito general se usó el término "aplicación de uso general". De los estudios primarios seleccionados 117 aportaron esta información.

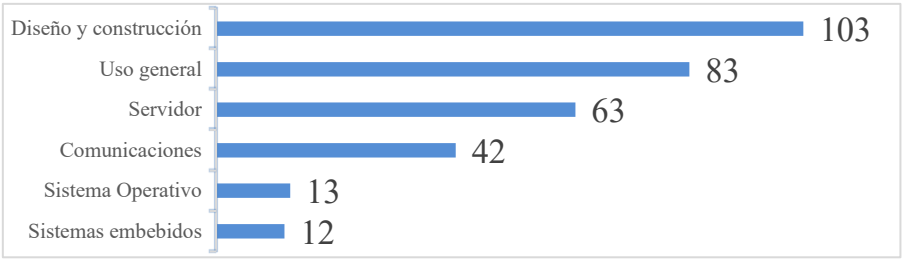

Fig. 4. Tipos de software.

El 88% de los artículos trabajaron con proyectos software para el diseño y construcción de sistemas. Por ejemplo, compiladores [P41], frameworks de desarrollo [P8], entornos de desarrollo integrado [P2], interfaces de programación de aplicaciones [P4], herramientas de construcción [P8], herramientas para el modelado del sistema [P4] o librerías para la generación de casos de prueba [P5], entre otros.

El 71% emplearon aplicaciones de uso general. En esta clasificación entran: procesadores de texto [P1], videojuegos [P44], navegadores [P20], aplicaciones cliente de mensajería [P1], sitios web [P1], o reproductores de música [P28], entre otras.

El 54% trabajaron con servidores, es decir, sistemas preparados para la recepción de solicitudes de clientes. Por ejemplo, servidores web [P4], servidores de base de datos

[P9], servidores de transferencia de archivos [P1], servidores para computación distribuida [P56] o servidores de instancias de virtualización [P71]. En el 36% utilizaron software de sistema para tareas de redes y comunicaciones entre aplicaciones y dispositivos, por ejemplo: Apache Synapse [P11], dnsjava [P21], Quagga [P41], ActiveMQ [P71].

El 11% trabajaron con sistemas operativos, como Linux [P20] y Android [P27]. Finalmente, el 10% utilizaron sistemas embebidos, es decir, software embebido en un dispositivo mecánico o eléctrico diseñado para realizar una función específica. Por ejemplo, software para sistemas armamentísticos [P23], sistemas de navegación aéreo [P29], controladores de motores de combustión [P34] y soluciones de procesamiento de audio [P34], entre otros.

4.3 RQ3: Métricas del código

Dependiendo del objeto de estudio del artículo se recolectaron diferentes datos del código de cada uno de los proyectos, en la **¡Error! No se encuentra el origen de la referencia.** fueron agruparon siguiendo la taxonomía elaborada por Elmidaoui et al. 38. De los 68 estudios que extrajeron métricas del código, el 66% son métricas de tamaño del código fuente, por ejemplo, líneas de código [P14], o líneas de comentarios [P16].

Table 6. Categorías de meta-datos.

#	Categorías de Meta-datos	Artículos
45	Tamaño Código Fuente	P8 P11 P14 P16 P23 P24 P26 P34 P36 P44 P50 P51 P53 P56 P67 P69 P71 P73 P74 P75 P76 P80 P82 P87 P92 P93 P96 P100 P107 P109 P112 P113 P114 P119 P120 P122 P124 P128 P132 P135 P138 P141 P143 P146 P148
44	Diseño	P5 P6 P11 P14 P16 P19 P26 P34 P35 P36 P44 P50 P51 P53 P54 P55 P65 P67 P74 P75 P76 P80 P93 P96 P97 P106 P107 P109 P112 P113 P116 P119 P121 P122 P124 P127 P132 P135 P138 P141 P143 P145 P146 P148
23	Code Smells	P11 P31 P36 P37 P44 P54 P61 P66 P74 P75 P88 P90 P92 P100 P102 P109 P116 P129 P132 P135 P139 P145 P150

En el 65% de artículos se utilizan métricas de diseño. Estas miden propiedades inherentes del software y se puede disgregar en otras 7 subcategorías: complejidad, acoplamiento, diagrama de clases, herencia, cohesión, encapsulamiento y polimorfismo (ver **¡Error! No se encuentra el origen de la referencia.**).

Las métricas de complejidad calculan la cantidad de caminos existentes en el flujo de control del programa; como la complejidad ciclomática de McCabe [P26], o el peso de métodos por clase [P11]. Las métricas de acoplamiento cuantifican la asociación o interdependencia entre los módulos, como el acoplamiento entre objetos [P11] o la respuesta por clase [P14].

Las medidas de diagrama de clases representan la estructura estática del sistema y pueden clasificarse en términos de clases, métodos y atributos, por ejemplo: el número de métodos [P14], el número de clases [P44] o el número de atributos [P14].

Las métricas de herencia miden los atributos y métodos compartidos entre clases en una relación jerárquica, dentro de las cuales están el número de hijos [P11], la profundidad de árbol de herencia [P26]. Las métricas de cohesión establecen el grado en que métodos y atributos de una misma clase están conectados, como la falta de cohesión en los métodos, la cohesión de clases ajustada o la cohesión de clases suelta [P14].

Las medidas de encapsulamiento calculan los datos y funciones empaquetadas en una sola unidad, por ejemplo, el número de métodos accessors [P44], la métrica de acceso a datos [P54]. Y las medidas de polimorfismo se usan para cuantificar la habilidad de un componente de tomar múltiples formas, como el número de métodos polimórficos [P54].

Table 7. Subcategorías de meta-datos de diseño.

#	Subcategorías de Diseño	Artículos
32	Complejidad	P11 P16 P26 P34 P36 P44 P50 P51 P67 P74 P75 P76 P80 P93 P97 P106 P107 P109 P112 P113 P116 P119 P122 P124 P127 P132 P135 P138 P143 P145 P146 P148
27	Acoplamiento	P5 P11 P14 P19 P26 P36 P44 P50 P53 P54 P67 P76 P80 P96 P106 P107 P116 P119 P122 P124 P127 P132 P138 P141 P143 P145 P146
23	Diagrama de Clases	P14 P26 P35 P36 P44 P50 P53 P54 P65 P67 P76 P80 P93 P96 P106 P107 P119 P122 P127 P135 P138 P141 P146
20	Herencia	P6 P11 P26 P44 P50 P54 P67 P76 P80 P106 P107 P109 P116 P121 P122 P124 P127 P138 P143 P146
20	Cohesión	P11 P14 P26 P50 P54 P55 P67 P76 P80 P96 P106 P107 P116 P119 P122 P124 P127 P138 P143 P146
10	Encapsulamiento	P44 P53 P54 P67 P80 P96 P106 P107 P122 P138
1	Polimorfismo	P54

La tercera categoría es code smells. Los code smells son estructuras en el código fuente que indican la existencia de un problema de calidad o estructural, y plantean la necesidad de realizar una refactorización. 39. Si bien esta categoría no se encuentra presente en la taxonomía original 38, es razonable incluirla porque se ha establecido como un método efectivo para descubrir problemas en el código fuente [P44]. Dicho esto, el 34% de los estudios recolectan code smells.

4.4 RQ4: Herramientas

Analizamos las herramientas empleadas para generar métricas de código, así como otros artefactos relacionados con el código. En total se reportaron 59 herramientas, las cuales fueron clasificadas de acuerdo a los datos o artefactos extraídos del código en seis categorías: métricas, code smells y vulnerabilidades, estructura y dependencias, código refactorizado, conjunto de pruebas y patrones de diseño. De los estudios seleccionados, 50 informaron herramientas para generar métricas y artefactos del código.

En la **¡Error! No se encuentra el origen de la referencia.** se presentan las herramientas agrupadas según la clasificación antes mencionada. El 58% calcularon métricas de código con Understand [P26, P50, P74], CKJM [P26, P67, P132], PMD [P11, P22, P44]. El 42% extrajeron code smells o vulnerabilidades con iPlasma [P44], InFu-

sion [P88, P102], SonarQube [P75, P124]. En un 40% obtuvieron la estructura y dependencias del sistema con PomWalker [P68], Analizo [P93], SDMetrics [P124]. En menor medida, refactorizaron código (8%), generaron conjuntos de prueba (8%) y detectaron patrones de diseño (4%).

Table 8. Tipo de herramientas.

#	Datos extraídos	Artículos
29	Métricas	P11 P16 P22 P23 P26 P35 P44 P50 P51 P54 P56 P63 P67 P69 P74-P76 P88 P89 P93 P94 P102 P104 P107 P109 P124 P139 P145 P146
21	Code smells y vulnerabilidades	P11 P16 P22 P36 P37 P42 P44 P54 P66 P75 P88 P90 P100 P102 P107 P109 P116 P124 P139 P145 P150
20	Estructura y dependencias	P6 P16 P17 P26 P34 P42 P45 P50 P51 P68 P74 P76 P89 P93 P102 P104 P111 P124 P134 P146
4	Código refactorizado	P55 P74 P76 P77
4	Conjunto de pruebas	P4 P25 P60 P62
2	Patrones de diseño	P42 P54

4.5 RQ5: Análisis Estadísticos

Es habitual que los investigadores realicen un análisis previo a los datos recabados para exponer los hallazgos encontrados. Este proceso permite identificar las características que posee la muestra y a su vez seleccionar los estudios apropiados para generar los resultados. De los artículos recolectados se extrajeron los análisis estadísticos utilizados y se clasificaron en las dos áreas generales de la estadística: estadística inferencial y estadística descriptiva (ver **¡Error! No se encuentra el origen de la referencia.**), tomando como referencia las clasificaciones de Sheskin 40. En total 109 estudios informaron los procedimientos estadísticos realizados.

Table 9. Clasificación de análisis estadísticos.

#	Estadística	Artículos
84	Inferencial	P1-P3 P5 P6 P8 P10-P13 P15 P17 P19 P22 P25-P30 P32 P34 P35 P38 P42 P43 P46 P49-P51 P53 P54 P56-P62 P66 P67 P69 P70 P72 P73 P76 P77 P79 P80 P84 P85 P88 P89 P93 P95 P96 P100 P102-P107 P111 P113 P115 P116 P119 P122 P126 P128 P130 P132 P133 P136 P138 P140-P142 P145-P149
81	Descriptiva	P1-P4 P7 P8 P10 P12-P15 P17 P21 P25 P26 P28-P30 P35 P36 P38 P42 P46 P47 P49 P50 P51 P53 P55-P61 P66-P71 P73 P77 P78 P86 P88 P89 P92 P95 P98 P100 P102 P104 P106-P109 P111 P112 P114 P116-P118 P122 P124 P128-P133 P139-P145 P147-P149

La estadística inferencial emplea los datos para sacar conclusiones o hacer predicciones. En el 77% de los artículos seleccionaron procedimientos inferenciales que se clasificaron en 2 categorías, test no paramétrico y test paramétrico (ver **¡Error! No se encuentra el origen de la referencia.**).

Una de las diferencias entre estos grupos es que los test paramétricos hacen suposiciones específicas con respecto a uno o más parámetros de la población que caracterizan la distribución subyacente para la cual el test está siendo empleado. Ejemplos de test paramétricos son: test T [P10], test chi cuadrado de Wald [P51], y test de análisis de varianza [P1]. Dentro de los no paramétricos hay ejemplos como test Mann-Whitney

[P2], test de rangos con signo de Wilcoxon [P3], y test de normalidad Shapiro-Wilk [P28].

Table 10. Procedimientos estadísticos inferenciales.

#	Procedimientos Inferenciales	Artículos
76	Test No Paramétrico	P1 P2 P3 P5 P6 P8 P10 P12 P13 P15 P17 P19 P22 P25 P26 P27 P28 P29 P30 P32 P34 P35 P38 P42 P43 P46 P49 P50 P51 P53 P54 P56 P57 P58 P59 P60 P61 P62 P66 P67 P69 P70 P72 P73 P76 P77 P79 P80 P85 P89 P93 P95 P100 P102 P104 P106 P107 P111 P115 P116 P122 P126 P128 P130 P132 P133 P136 P138 P140 P141 P142 P145 P146 P147 P148 P149
24	Test Paramétrico	P1 P10 P11 P26 P27 P28 P46 P49 P51 P59 P60 P70 P76 P84 P88 P96 P103 P111 P113 P115 P119 P122 P141 P149

Por otra parte, el 74% de los artículos trabaja con procedimientos descriptivos utilizados para presentar y resumir los datos. Estos procedimientos fueron clasificados en cinco categorías: tamaño del efecto, medida de variabilidad, medida de tendencia central, medida de asimetría y medida de curtosis (ver **¡Error! No se encuentra el origen de la referencia.**).

Table 11. Procedimientos estadísticos descriptivos.

#	Procedimientos Descriptivos	Artículos
65	Tamaño del efecto	P7 P8 P10 P12 P14 P17 P21 P25 P28 P29 P30 P35 P36 P38 P42 P47 P49 P50 P51 P53 P55 P56 P59 P60 P61 P66 P68 P69 P70 P71 P73 P77 P78 P86 P88 P89 P92 P95 P98 P100 P102 P104 P106 P107 P109 P111 P112 P116 P117 P118 P122 P124 P128 P130 P132 P133 P139 P140 P141 P142 P143 P144 P147 P148 P149
31	Medida de Variabilidad	P1 P2 P3 P4 P10 P13 P14 P17 P26 P28 P30 P38 P46 P49 P51 P57 P58 P59 P67 P68 P88 P102 P108 P114 P116 P122 P129 P139 P141 P143 P147
33	Medida de Tendencia Central	P1 P2 P3 P4 P10 P13 P14 P17 P26 P28 P30 P38 P46 P49 P51 P58 P59 P67 P68 P88 P102 P108 P114 P116 P122 P131 P139 P141 P143 P145 P147 P148 P149
3	Medida de Asimetría	P4 P15 P46
2	Medida de Curtosis	P4 P46

El tamaño de efecto mide la magnitud de fuerza de un fenómeno o efecto en un experimento. Por ejemplo, podemos mencionar: el coeficiente de correlación Spearman [P8], el tamaño de efecto de Cliff [P28] o el coeficiente de correlación de Pearson [P36].

Las medidas de variabilidad y las de tendencia central son los estadísticos más básicos empleados tanto en la estadística inferencial como en la descriptiva. En esta clasificación fueron incluidos solamente dentro de los procedimientos descriptivos porque en cada caso los usaron con el fin de describir la muestra recolectada. Como ejemplos de medidas de variabilidad se encuentran la desviación estándar [P46], los cuartiles

[P14] y la varianza [P108]. En el caso de medidas de tendencia central están la media [P51], la mediana [P4] y la moda [P108].

5 Discusión

En esta sección se discuten los resultados obtenidos y como se relacionan con las preguntas de investigación identificadas en la Sección 2. Sin embargo, es posible que no se hayan localizado todos los estudios relevantes, por esa razón el proceso fue desarrollado metodológicamente siguiendo un protocolo bien definido y múltiples investigadores revisaron la calidad de la información extraída. El mapeo comprendió 150 artículos publicados en la revista EMSE y las conferencias ESEM y EASE durante el período 2013 – 2021.

5.1 RQ1: Criterios de selección

Para abordar esta pregunta se buscaron evidencias de la selección de proyectos software con dos enfoques, la estrategia general y los criterios de selección específicos. A nivel general, en la mayoría de los casos (67%) los investigadores siguen lineamientos propios para seleccionar sus proyectos y en menor medida (31%) utilizan una colección de proyectos existentes. Con respecto al último, en total se emplearon 44 colecciones creadas en otro estudio. En la **¡Error! No se encuentra el origen de la referencia.** se pueden visualizar las más populares. Las colecciones más utilizadas, Jureczko 41, Bug Prediction Dataset 42 y NASA 43 contienen valores de métricas obtenidas del código de los proyectos.

Table 12. Colecciones de proyectos.

#	Nombre de la colección	Artículos
12	Jureczko 41	P11 P80 P96 P99 P106 P107 P112 P116 P119 P122 P138 P143
4	Bug Prediction Dataset (BPD) 42	P112 P127 P138 P143
3	NASA 43	P29 P113 P120
3	Maxwell 44	P3 P15 P57
3	Miyazaki 45	P3 P57 P117
3	SF100 46	P25 P60 P146
2	Qualitas Corpus 6	P44 P121

De los 113 estudios que reportaron información acerca de los criterios específicos, los predominantes fueron: específicos del caso de estudio (41%), el período de actividad en el proyecto (35%), la disponibilidad de los proyectos y meta-datos (33%), el tamaño del proyecto (28%) que tiene en cuenta diferentes dimensiones desde la cantidad de líneas de código hasta medidas de punto función, la popularidad (27%), y el dominio de aplicación (23%). Esto evidencia la diversidad de criterios existentes para recolectar las muestras y la ausencia de un procedimiento estándar para la construcción de colecciones de proyectos.

Un análisis que puede extenderse de los datos recolectados para esta RQ es la estrategia de muestreo. Aquellos estudios primarios que utilizaron lineamientos propios

y no informaron criterios de selección para recolectar los proyectos podrían estar relacionados con el muestreo por conveniencia (15 estudios). En el muestreo por conveniencia los sujetos se seleccionan arbitrariamente o en función de su cercanía, disponibilidad o facilidad de estudio 47. El principal problema con este enfoque es que amenaza la generalización de los resultados.

Los casos en los que los autores establecieron claramente los criterios que deben tener los proyectos para ser seleccionados y recopilados de forma cuidadosa pero no aleatoria, o utilizaron colecciones construidas por otros autores (129 estudios), podrían estar asociados con el muestreo con propósito, en el sentido de que los elementos se seleccionan de acuerdo con alguna lógica o estrategia 47. Seis estudios emplearon técnicas de muestreo probabilístico o utilizaron colecciones que las aplicaron [P4, P25, P60, P134, P139, P146].

5.2 RQ2: Características de los proyectos

Para caracterizar la muestra de los proyectos software recolectados por los grupos de investigación se describieron el lenguaje de programación principal y el tipo de actividad o función que desempeña. De 138 artículos que reportan el lenguaje de programación principal de los proyectos, la mayoría (80%) experimentan con Java y en menor grado (28%) los lenguajes C o C++. Esto puede tener una relación directa con la cantidad de proyectos de fuente abierta en estos lenguajes en repositorios de software como Github 48, Apache Software Foundation 49 and SourceForge 50. Otra razón es la gran cantidad de herramientas de análisis estático del código fuente compatibles disponibles para el caso del lenguaje Java.

Finalmente, de los 117 artículos que fue posible determinar el tipo de software empleado la mayoría era para el diseño y construcción de sistemas (88%) y, en segundo lugar, aplicaciones de uso general (71%).

5.3 RQ3: Métricas del código

Para responder esta pregunta se buscaron las métricas extraídas del código de los proyectos software. Así se obtuvieron 68 estudios primarios, y de este conjunto se encontraron métricas que cuantifican el tamaño (67%), diseño (63%) y los code smells (32%). Esto evidencia el uso no generalizado de métricas de código en estudios ISBE.

El resto de los trabajos se valieron de otras fuentes de información. Por ejemplo, los reportes de defectos para calcular su tiempo de resolución [P1, P46, P56], clasificarlos [P7, P20, P33, P52] o detectar reportes duplicados [P47]; entre otros estudios. También la información de ejecución del programa (análisis dinámico) para localizar funciones específicas [P2], clasificar los procesos [P18], estudiar la asignación de memoria [P40], analizar cómo trabajan las excepciones [P64] o las dependencias del sistema en ejecución [P104].

Otra fuente han sido los datos contenidos en el repositorio del proyecto, para medir el acoplamiento y dependencias en la evolución del sistema [P5, P6, P32], clasificar commits [P9, P24], predecir defectos en el software [P29]; estudiar el uso de patrones en la historia del proyecto [P42] o estudiar el aporte de los desarrolladores [P51], entre

otros. Para estos casos es necesario que los proyectos tengan un conjunto de meta-datos registrados a lo largo de su tiempo de desarrollo a través de un sistema para la gestión de versiones.

5.4 RQ4: Herramientas

Es común en estos estudios hallar diferentes herramientas para dar soporte a tareas específicas para lograr precisión y repetitividad 51. Dada la variedad de herramientas encontradas, se realizó una clasificación en seis categorías en base a los datos o artefactos extraídos del código: métricas, code smells y vulnerabilidades, estructura y dependencias, código refactorizado, conjunto de pruebas y patrones de diseño. En la **¡Error! No se encuentra el origen de la referencia.** se puede observar un gráfico de burbujas con la clasificación antes mencionada (eje vertical), donde están descripta la cantidad de herramientas encontradas, cuántas de ellas siguen vigentes, es decir, si la última versión estable sigue en funcionamiento y cuantas han sido actualizadas después del 1º de enero del 2021 (eje horizontal).

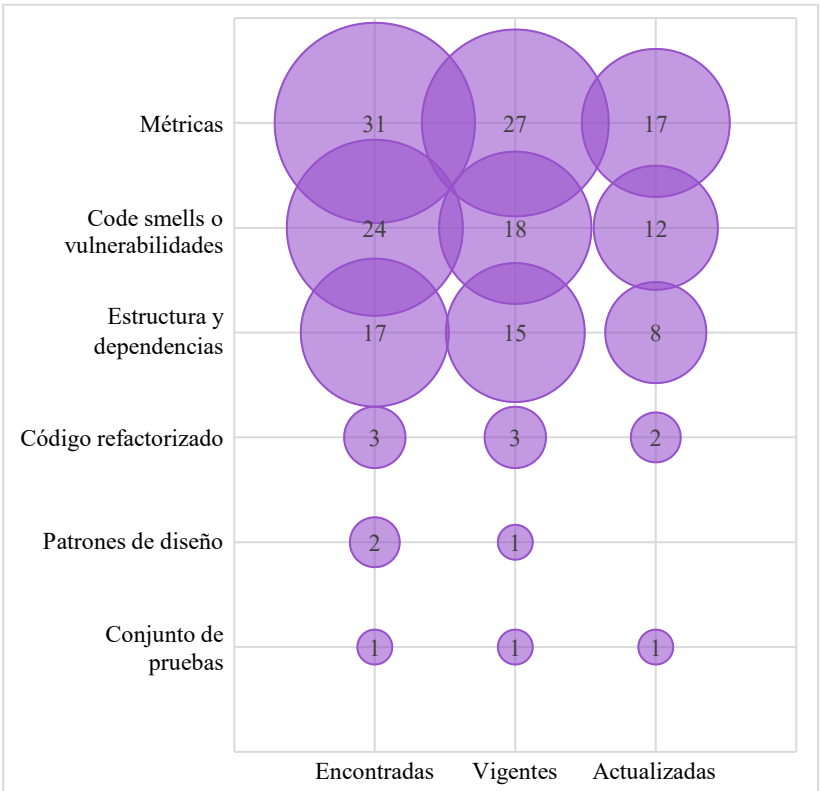

Fig. 5. Cantidad de herramientas encontradas, vigentes y actualizadas por tipo.

En muchas ocasiones se informa el modelo, técnica, procedimiento o algoritmo utilizado, pero no así la herramienta utilizada. Por ejemplo, Dit et al. [P2] utilizan un modelo de fusión de datos compuesto por técnicas de recuperación de información, análisis dinámicos y minado Web para la localización de métodos que desarrollan funciones específicas del sistema. Corazza et al. [P3] implementan un algoritmo de búsqueda meta-heurístico en un modelo de aprendizaje automatizado para estimar el esfuerzo de desarrollo. Chen et al. [P18] declararon el uso de un "método propio" para la refactorización de artefactos software. Dallal y Morasca [P14] trabajaron con una herramienta con la que calculan 29 métricas de código fuente, pero no reportan su nombre, ni como acceder a la misma.

Table 13. Herramientas más utilizadas.

#	Herramientas	Artículos
8	Understand	P26 P50 P51 P74 P76 P89 P102 P104
5	Evo Suite	P4 P25 P60 P62 P146
4	CKJM	P26 P67 P107 P132
3	PMD	P11 P22 P44
3	Refactoring Miner	P74 P76 P149
3	CCFinder	P90 P116 P130
3	DECOR	P42 P54 P129

5.5 RQ5: Análisis estadísticos

Esta pregunta pretende determinar qué técnicas o procedimientos estadísticos son elegidos por los investigadores para validar los resultados de los experimentos realizados. En este sentido, la selección y aplicación apropiada de los métodos de análisis es una de las recomendaciones de Wohlin y Rainer 52 para garantizar que la evidencia generada se presente correctamente y evitar que existan interpretaciones erróneas de los resultados.

De los 109 artículos que registraron procedimientos estadísticos el 77% son inferenciales, de los cuales el 29% contiene tests estadísticos paramétricos para el análisis de los datos recopilados. Esto implica supuestos, como que los datos obtenidos sigan una distribución normal, lo que podría no coincidir con la distribución subyacente de la población donde se extrae los proyectos o los mecanismos de extracción de los datos. Así, por ejemplo, muchos estudios utilizan reglas basadas en la disponibilidad del proyecto o su popularidad, pero no en la representatividad de los datos con respecto a la población.

Finalmente, en 68 estudios incorporaron un análisis de tamaño del efecto. En Kitchenham et al. 53 remarcan su utilidad porque proporcionan una medida objetiva de la importancia que tiene un fenómeno en un experimento, independientemente de la significación estadística de la prueba de hipótesis conducida. Además, le afecta menos el tamaño de la muestra que a la significación estadística, siendo una reco-mendación para los casos en que las muestras sean poco representativas.

5.6 Implicancias

En los estudios en ISBE los resultados deben ser replicables y generalizables. La replicación es importante para validar los resultados de un estudio científico y para ello se debe proporcionar las colecciones de proyectos, herramientas y scripts de la investigación realizada. Sin un paquete de replicación, no se pueden comparar los resultados anteriores con los nuevos, y no se puede garantizar la confianza de los resultados 54.

Las colecciones de proyectos podrían tener un problema de generalización debido a problemas metodológicos al seleccionar los proyectos. La falta de representatividad en una muestra puede sesgar la generalización de los hallazgos 55. El caso ideal sería seleccionar aleatoriamente una muestra estadísticamente significativa de proyectos. Este enfoque se conoce como muestreo probabilístico, donde se emplea la aleatoriedad en el sentido de que cada elemento de la población tiene una probabilidad de selección. Dado que los elementos no se seleccionan arbitrariamente, estas técnicas facilitan la generalización estadística 56 pero no siempre son aplicables. Otra forma de producir muestras representativas es mediante el muestreo por propósito. En este sentido, enfoques como el propuesto por Nagappan et al. 56, para construir muestras diversas y representativas, parecen prometedoras.

Finalmente, para reducir la amenaza de producir resultados no generalizables, los profesionales e investigadores deben evitar consumir colecciones de proyectos en los que su estudio fundacional no informe claramente el procedimiento de selección. Al construir una colección de proyectos no es suficiente compartir solo los proyectos (producto), sino que es necesario proporcionar el procedimiento para recolectarlos (proceso).

6 Amenazas a la validez

A continuación, se discuten las amenazas a la validez del estudio siguiendo el enfoque propuesto por Runeson y Höst 57. Se consideraron cinco aspectos de validez.

- Validez de constructo. La validez del constructo refleja hasta qué punto la metodología de la investigación representa a la estrategia del investigador y lo que se busca estudiar en las preguntas de investigación. Esta amenaza está presente al diseñar el instrumento de extracción de datos. La misma disminuyó implementando una prueba piloto de la tabla, tomando artículos al azar para completar una primera versión, y modificándola iterativamente según sea necesario hasta lograr la versión final.
- Validez interna. Este aspecto de validez analiza los riesgos cuando se estudian relaciones causales 58. Elegir los artículos y el período de tiempo adecuados son factores importantes que afectan la validez interna. La cantidad de artículos relacionados con la ISBE ha crecido mucho recientemente y las revistas de Ingeniería del Software tienen diferentes grados de aceptación para investigaciones empíricas. Para mitigar esta amenaza, se seleccionaron artículos de revistas y conferencias específicos en ISBE, que son ampliamente aceptados en el ámbito de la Ingeniería del Software en

términos de alcance y reputación. La subjetividad en la selección disminuyó mediante el proceso descrito en la Sección 3.1, realizando tantas iteraciones como fueron necesarias hasta obtener un grado de acuerdo alto.

- Validez externa. La validez externa se refiere hasta qué punto es posible generalizar los resultados más allá del estudio. Los hallazgos aquí presentados se basan en las publicaciones pertenecientes a EMSE, EASE y ESEM. Si bien se desconoce si los resultados se pueden generalizar a artículos de otras revistas o conferencias, la investigación se basó en el análisis de 150 artículos y por tanto, puede considerarse representativa.
- Fiabilidad. La fiabilidad evidencia hasta qué punto los resultados de la investigación son independientes de los investigadores. Es decir, si otro autor llevase a cabo el mismo estudio, los resultados deberían ser iguales o similares 57. En este artículo, los métodos y procesos de investigación se describen en detalle para garantizar su reproducibilidad y en el anexo se incluyen las planillas originales con los datos extraídos.
- Sesgo de publicación. El sesgo de publicación se refiere al "problema de que es más probable que se publiquen los resultados positivos de la investigación que los negativos" 59. Este problema ocurre en cualquier revisión de literatura o estudio de mapeo. Sin embargo, en este caso, su efecto fue moderado porque nuestro estudio no pretende comparar resultados de investigación.

7 Conclusiones

En este mapeo sistemático se identificaron artículos en los que se realizaron estudios empíricos con colecciones de proyectos. Se abordaron cinco preguntas de investigación de estos estudios, para determinar los criterios de selección y características de los proyectos, como también las métricas de código obtenidas, las herramientas empleadas para ello y los análisis estadísticos desarrollados. Por medio de una búsqueda manual inicial en la revista EMSE y las conferencias ESEM y EASE se obtuvieron 1709 artículos entre el 1 de enero del 2013 al 31 de diciembre del 2021. De los cuales 150 estudios fueron seleccionados después de aplicar los criterios de inclusión y exclusión definidos.

De acuerdo a nuestros hallazgos, las estrategias generales más comunes para la construcción de colecciones de proyectos son: utilizar lineamientos propios y emplear una colección existente. La mayoría de los artículos reportaron criterios específicos de proyectos, y estos fueron: específicos del caso de estudio, la actividad en el tiempo, la disponibilidad de los datos, el tamaño y la popularidad. El lenguaje de programación principal de los proyectos software fue Java (80%), y C/C++ (28%); y el tipo de software utilizado fueron: software para la construcción de sistemas (88%), aplicaciones de uso general (71%) y servidores (54%). Menos de la mitad de los estudios informaron métricas de código extraídas, de las cuales en su mayoría fueron de tamaño y diseño. Un tercio de los artículos reportaron herramientas para generar métricas y otros artefactos del código. Con respecto a los análisis estadísticos, los procedimientos inferenciales más frecuentes fueron pruebas estadísticas no paramétricas, y también se aplicaron medidas de tamaño del efecto.

Considerando los resultados del mapeo sistemático, es oportuno realizar las siguientes observaciones. Con respecto a la repetición de los estudios en ISBE, es indispensable que los autores faciliten los proyectos software y las herramientas empleadas para generar los resultados, ofreciendo guías y asistencia a los investigadores de ser necesario. Con respecto a la construcción de muestras representativas, se recomienda emplear técnicas de muestreo probabilístico para seleccionar los proyectos. Creemos esencial aumentar los esfuerzos para recopilar los proyectos de software utilizando un marco claro y sistemático.

Finalmente, como aporte de este trabajo se identificaron algunas pautas que ayudan a sistematizar la selección de proyectos en la construcción de colecciones con fines de investigación. En líneas generales, las reglas que se pueden reunir son: código fuente libre tanto su acceso como distribución, proyectos vigentes con historia de desarrollo, que sean repositorios populares y estén construidos en lenguaje Java. La colección resultante debería conservar el código fuente de los proyectos, sus métricas obtenidas del análisis estático del código y las herramientas empleadas. Como trabajos futuros se analizarán las colecciones creadas con fines de investigación presentes en la **¡Error! No se encuentra el origen de la referencia.** y en artículos relacionados, con el objetivo de reconocer cuales fueron los criterios considerados en cada caso, el propósito de su construcción, y la vigencia de estos.

Referencias

1. Parnas, D.L.: Some software engineering principles. In: Software fundamentals: collected papers by David L. Parnas. pp. 257–266 (2001).
2. Lehman, M.M.: Laws of software evolution revisited. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 108–124. Springer Verlag (1996). https://doi.org/10.1007/BFb0017737.
3. Kitchenham, B., Pfleeger, S.L.: Software quality: the elusive target. IEEE Softw. 13, 12–21 (1996). https://doi.org/10.1109/52.476281.
4. Garvin, D.A.: What Does “Product Quality” Really Mean?, https://sloanreview.mit.edu/article/what-does-product-quality-really-mean/, (1984).
5. Kitchenham, B.A., Dybå, T., Jørgensen, M.: Evidence-based Software Engineering. In: Proceedings of the 26th International Conference on Software Engineering. pp. 273–281 (2004).
6. Tempero, E., Anslow, C., Dietrich, J., Han, T., Li, J., Lumpe, M., Melton, H., Noble, J.: The Qualitas Corpus: A curated collection of Java code for empirical studies. In: Proceedings - Asia-Pacific Software Engineering Conference, APSEC. pp. 336–345 (2010). https://doi.org/10.1109/APSEC.2010.46.
7. Barone, A.V.M., Sennrich, R.: A parallel corpus of Python functions and documentation strings for automated code documentation and code generation. (2017).
8. Allamanis, M., Sutton, C.: Mining Source Code Repositories at Massive Scale using Language Modeling. In: 2013 10th Working Conference on Mining Software Repositories (MSR). pp. 207–216 (2013).
9. Keivanloo, I., Rilling, J., Zou, Y.: Spotting working code examples. In: Proceedings - International Conference on Software Engineering. pp. 664–675. IEEE Computer Society (2014). https://doi.org/10.1145/2568225.2568292.

10. Zerouali, A., Mens, T.: Analyzing the Evolution of Testing Library Usage in Open Source Java Projects. In: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). pp. 417–421 (2017).
11. Goeminne, M., Mens, T.: Towards a Survival Analysis of Database Framework Usage in Java Projects. In: 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 551–555 (2015).
12. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic Mapping Studies in Software Engineering. In: Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering. pp. 68–77 (2008).
13. Knuth, D.E.: An Empirical Study of FORTRAN Programs. Softw. Pract. Exp. 1, 105–133 (1971). https://doi.org/10.1002/spe.4380010203.
14. Chidamber, S.R., Kemerer, C.F.: A Metrics Suite for Object Oriented Design. (1994).
15. Chidamber, S.R., Kemerer, C.F.: Towards a metrics suite for object oriented design. In: Conference proceedings on Object-oriented programming systems, languages, and applications - OOPSLA '91. pp. 197–211. Association for Computing Machinery (ACM), New York, New York, USA (1991). https://doi.org/10.1145/117954.117970.
16. Harrison, R., Counsell, S., Nithi, R.: Coupling Metrics for Object-Oriented Design. In: Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262) (1998). https://doi.org/10.1109/METRIC.1998.731240.
17. Terra, R., Miranda, L.F., Valente, M.T., Bigonha, R.S.: Qualitas.class corpus. ACM SIGSOFT Softw. Eng. Notes. 38, 1–4 (2013). https://doi.org/10.1145/2507288.2507314.
18. Howison, J., Conklin, M., Crowston, K.: FLOSSmole: A collaborative repository for FLOSS research data and analyses. Int. J. Inf. Technol. Web Eng. 1, 17–26 (2006). https://doi.org/10.4018/jitwe.2006070102.
19. Herraiz, I., Izquierdo-Cortazar, D., Rivas-Hernandez, F., Gonzalez-Barahona, J., Robles, G., Dueñas-Domínguez, S., Garcia-Campos, C., Gato, J.F., Tovar, L.: Flossmetrics: Free / libre / open source software metrics. In: Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR. pp. 281–284. IEEE Computer Society (2009). https://doi.org/10.1109/CSMR.2009.43.
20. Gyimesi, P., Gyimesi, G., Tóth, Z., Ferenc, R.: Characterization of Source Code Defects by Data Mining Conducted on GitHub. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 47–62. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21413-9_4.
21. Chávez, A., Ferreira, I., Fernandes, E., Cedrim, D., Garcia, A.: How does refactoring affect internal quality attributes?: A multi-project study. In: ACM International Conference Proceeding Series. pp. 74–83. Association for Computing Machinery (2017). https://doi.org/10.1145/3131151.3131171.
22. Petersen, K., Vakkalanka, S., Kuzniarz, L.: Guidelines for conducting systematic mapping studies in software engineering: An update. In: Information and Software Technology. pp. 1–18. Elsevier (2015). https://doi.org/10.1016/j.infsof.2015.03.007.
23. Kitchenham, B.A., Budgen, D., Brereton, P.O.: Using mapping studies as the basis for further research - A participant-observer case study. Inf. Softw. Technol. 53, 638–651 (2011). https://doi.org/10.1016/j.infsof.2010.12.011.
24. Zhang, L., Tian, J.H., Jiang, J., Liu, Y.J., Pu, M.Y., Yue, T.: Empirical Research in Software Engineering — A Literature Survey. J. Comput. Sci. Technol. 33, 876–899 (2018). https://doi.org/10.1007/s11390-018-1864-x.
25. Molléri, J.S., Petersen, K., Mendes, E.: Survey Guidelines in Software Engineering: An Annotated Review. In: International Symposium on Empirical Software Engineering and Measurement. IEEE Computer Society (2016). https://doi.org/10.1145/2961111.2962619.

26. Storey, M.A., Ernst, N.A., Williams, C., Kalliamvakou, E.: The who, what, how of software engineering research: a socio-technical framework. Empir. Softw. Eng. 25, 4097–4129 (2020). https://doi.org/10.1007/s10664-020-09858-z.
27. Ali, N.B., Petersen, K.: Evaluating strategies for study selection in systematic literature studies. In: International Symposium on Empirical Software Engineering and Measurement. pp. 1–4. IEEE Computer Society, New York, New York, USA (2014). https://doi.org/10.1145/2652524.2652557.
28. Kitchenham, B., Brereton, P.: A systematic review of systematic review process research in software engineering, https://linkinghub.elsevier.com/retrieve/pii/S0950584913001560, (2013). https://doi.org/10.1016/j.infsof.2013.07.010.
29. Kitchenham, B., Charters, S., Budgen, D., Brereton, P., Turner, M., Linkman, S., Jørgensen, M., Mendes, E., Visaggio, G.: Guidelines for performing Systematic Literature Reviews in Software Engineering. (2007).
30. Higgins, J.P.T., Thomas, J., Chandler, J., Cumpston, M., Li, T., Page, M.J., Welch, V.A.: Cochrane Handbook for Systematic Reviews of Interventions. Wiley (2019). https://doi.org/10.1002/9781119536604.
31. Mendeley, https://www.mendeley.com/, last accessed 2022/05/22
32. Framendeley, https://chrome.google.com/webstore/detail/framendeley/decpeaebklmmgfhnnhggeikfhhlbcjpf, last accessed 2022/05/22.
33. Saldaña, J.: The Coding Manual for Qualitative Researchers. SAGE Publications Ltd (2015).
34. Janssen, M., van der Voort, H.: Agile and adaptive governance in crisis response: Lessons from the COVID-19 pandemic. Int. J. Inf. Manage. 55, 102180 (2020). https://doi.org/10.1016/j.ijinfomgt.2020.102180.
35. Crabtree, B.F., Miller, W.L.: Doing Qualitative Research. SAGE Publications, Inc (1999).
36. Anexo, http://bit.ly/AnexoTablaCompleta, last accessed 2022/05/22
37. Forward, A., Lethbridge, T.C.: A Taxonomy of Software Types to Facilitate Search and Evidence-Based Software Engineering. (2008).
38. Elmidaoui, S., Cheikhi, L., Idri, A.: Towards a Taxonomy of Software Maintainability Predictors. In: Advances in Intelligent Systems and Computing. pp. 823–832. Springer Verlag (2019). https://doi.org/10.1007/978-3-030-16181-1_77.
39. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the Design of Existing Code. (2002).
40. Sheskin, D.J.: PARAMETRIC and NONPARAMETRIC STATISTICAL PROCEDURES SECOND EDITION. (2000).
41. Jureczko, M., Madeyski, L.: Towards identifying software project clusters with regard to defect prediction. In: ACM International Conference Proceeding Series. p. 1. ACM Press, New York, New York, USA (2010). https://doi.org/10.1145/1868328.1868342.
42. D'Ambros, M., Lanza, M., Robbes, R.: Evaluating defect prediction approaches: A benchmark and an extensive comparison. In: Empirical Software Engineering. pp. 531–577. Springer (2012). https://doi.org/10.1007/s10664-011-9173-9.
43. Shepperd, M., Song, Q., Sun, Z., Mair, C.: Data quality: Some comments on the NASA software defect datasets. IEEE Trans. Softw. Eng. 39, 1208–1215 (2013). https://doi.org/10.1109/TSE.2013.11.
44. Maxwell, K.: Applied Statistics for Software Managers. (2002).
45. Miyazaki, Y., Terakado, M., Ozaki, K., Nozaki, H.: Robust regression for developing software estimation models. J. Syst. Softw. 27, 3–16 (1994). https://doi.org/10.1016/0164-1212(94)90110-4.

46. Fraser, G., Arcuri, A.: Sound empirical evidence in software testing. In: Proceedings - International Conference on Software Engineering. pp. 178–188. IEEE (2012). https://doi.org/10.1109/ICSE.2012.6227195.
47. Baltes, S., Ralph, P.: Sampling in Software Engineering Research: A Critical Review and Guidelines. (2020).
48. GitHut 2.0, https://madnight.github.io/githut, last accessed 2022/05/22.
49. Apache Software Foundation, https://projects.apache.org, last accessed 2022/05/22.
50. Source Forge, https://sourceforge.net/directory, last accessed 2022/05/22.
51. Maibaum, T., Wassyng, A.: A product-focused approach to software certification. Computer (Long. Beach. Calif). 41, 91–93 (2008). https://doi.org/10.1109/MC.2008.37.
52. Wohlin, C., Rainer, A.: Challenges and recommendations to publishing and using credible evidence in software engineering. Inf. Softw. Technol. 134, 106555 (2021). https://doi.org/10.1016/j.infsof.2021.106555.
53. Kitchenham, B., Madeyski, L., Budgen, D., Keung, J., Brereton, P., Charters, S., Gibbs, S., Pohthong, A.: Robust Statistical Methods for Empirical Software Engineering. Empir. Softw. Eng. 22, 579–630 (2017). https://doi.org/10.1007/s10664-016-9437-5.
54. Cosentino, V., Luis, J., Izquierdo, C., Cabot, J.: Findings from GitHub: Methods, datasets and limitations. In: Proceedings - 13th Working Conference on Mining Software Repositories, MSR 2016. pp. 137–141. Association for Computing Machinery, Inc (2016). https://doi.org/10.1145/2901739.2901776.
55. Munaiah, N., Kroh, S., Cabrey, C., Nagappan, M.: Curating GitHub for engineered software projects. Empir. Softw. Eng. 22, 3219–3253 (2017). https://doi.org/10.1007/s10664-017-9512-6.
56. Nagappan, M., Zimmermann, T., Bird, C.: Diversity in Software Engineering Research. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2013. ACM Press, New York, New York, USA (2013). https://doi.org/10.1145/2491411.
57. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14, 131–164 (2009). https://doi.org/10.1007/s10664-008-9102-8.
58. Siegmund, J., Siegmund, N., Apel, S.: Views on internal and external validity in empirical software engineering. In: Proceedings - International Conference on Software Engineering. pp. 9–19. IEEE Computer Society (2015). https://doi.org/10.1109/ICSE.2015.24.
59. Unterkalmsteiner, M., Gorschek, T., Islam, A.K.M.M., Cheng, C.K., Permadi, R.B., Feldt, R.: Evaluation and measurement of software process improvement-A systematic literature review, (2012). https://doi.org/10.1109/TSE.2011.26.