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

Juan Andrés Carruthers[1], Jorge Andrés Díaz-Pace[2] y Emanuel Agustín Irrazábal[1]

[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

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.

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 relacionada por medio de mediciones vinculadas 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, una colección ad hoc de proyectos software a veces no es suficiente para lograr generalización, ni replicación de los resultados.

De esta manera surgen los “datasets” o colecciones de proyectos software, que reducen el costo de recopilar los proyectos y facilitan la replicación de los estudios [6]. Por ejemplo, las colecciones construidas por Barone y Sennrich [7], Allamanis y Sutton [8] o Keivanloo [9]. Estas 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, las reglas de selección de los proyectos no siempre son explícitas. La definición de un modelo de procedimientos a partir de reglas estándar es fundamental para garantizar la preservación sistemática de la colección en el tiempo. Un ejemplo es el Qualitas [6], pero su última versión es del año 2013. De sus 112 proyectos, 51 están descontinuados y el resto ha tenido un promedio de 58 cambios de versión.

El objetivo de este trabajo es determinar los criterios de selección y las características de los proyectos software que son insumos de estudios empíricos, las métricas extraídas del código, 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 del 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 que comprometen su replicación y la representatividad de las muestras.

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 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. Esto hizo más necesaria la construcción de colecciones de proyectos software que facilite la replicación de los experimentos, 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 para analizar 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 origi-

nal y descartaron aquellos que no estuvieran disponibles y los que no utilizaron la herramienta Maven. De igual manera, Goeminne y Mens [11] realizaron cambios al Github Java Corpus para estudiar cinco frameworks de base de datos. Por su parte, Tempero et al [6] construyeron la colección Qualitas Corpus donde incluyeron sistemas Java con sus archivos fuente y binarios disponibles en formato ".jar".

Otras colecciones, además de la documentación, código fuente y binarios del proyecto, también proporcionan meta-datos. Por ejemplo, Gyimesi et al. [13] y Chavez et al. [14] reunieron proyectos Java con sus métricas orientadas a objetos (falta de cohesión entre métodos, acoplamiento entre objetos, peso de métodos por clase, etc).

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 empíricos. A su vez también reportar las métricas de código extraídas de los proyectos, las herramientas utilizadas y los estudios estadísticos practicados. De esta manera, evaluaremos los factores que consideramos perjudiciales en la representatividad de las muestras y la replicación de los estudios empíricos, y realizaremos recomendaciones para la selección de proyectos.

3 Metodología

Se llevó a cabo un mapeo sistemático siguiendo las pautas para obtener una visión general del uso de colecciones de proyectos en la ISBE [15]. Se seleccionó esta técnica por centrarse en la "clasificación y análisis temático de un tema de la Ingeniería del Software" [16]. Aunque la recolección de proyectos sea una práctica estándar para estudios empíricos, los criterios de selección, sus características, las métricas del código extraídas, las herramientas utilizadas para ello y los análisis estadísticos ejecutados no han sido abordados ampliamente por otros estudios secundarios.

Las tablas y gráfico que detallan las tareas de cada etapa están en el material suplementario [17]. Las preguntas de investigación se presentan en la Tabla 1.

Tabla 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 Fig. A1.

- Búsqueda manual: se realizó una búsqueda manual en los foros Empirical Software Engineering (EMSE), Empirical Software Engineering and Measurement (ESEM) e International Conference on Evaluation and Assessment in Software Engineering (EASE) por ser representativos del área y haber sido utilizados en otros estudios [18, 19]. De acuerdo a estrategias como las de Zhang et al. [18] y Storey et al. [20] se seleccionó el período de tiempo comprendido entre el 1 de enero del 2013 hasta el 31 de diciembre del 2021 de EMSE, ESEM y EASE.
- 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 no de acuerdo con los criterios presentes en la Tabla 2. 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 junto con el CI o CE en que se basó la decisión. Para medir el acuerdo se utilizó el estadístico Kappa de Cohen [21] y se registró un valor de 0.77, demostrando un nivel de acuerdo alto.
- Muestreo "bola de nieve": se complementó la búsqueda manual con el método de bola de nieve hacia atrás, donde las referencias proporcionaron artículos relacionados. Los artículos recopilados durante esta etapa también se agregaron a la lista de candidatos y se seleccionaron de acuerdo a los criterios descritos anteriormente.

Tabla 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 definida se realizó una búsqueda manual en EMSE, ESEM y EASE donde fueron recolectados 1709 artículos; de los cuales se excluyeron 99 según los criterios de exclusión. De los 1610 incluidos en la fase 2, 1480 fueron rechazados por no cumplir algún criterio de inclusión. A los 130 artículos aceptados se sumaron 20 en la fase 4, quedando 150. Los resultados se observan en la Tabla A1.

3.3 Extracción de datos

Se utilizó un cuestionario de extracción de datos basado en las preguntas de investigación para recopilar información relevante. La extracción fue realizada por dos investi-

gadores y revisada por un tercero, para verificar la consistencia de los artículos con los datos del formulario. Los datos recolectados incluyeron información general (título, autores, año de publicación y fuente) e información relativa a las preguntas de investigación, como se ilustra en la Tabla A2.

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

3.4 Análisis de los datos

Las respuestas fueron analizadas de forma cualitativa mediante la codificación abierta [22] y cerrada [23] de los textos encontrados en los estudios recolectados. La codificación abierta se basó en la realizada por Janssen y Van der Voort [24]. La cerrada se utilizó para identificar y resignificar las respuestas a las taxonomías encontradas.

En ambos casos, dos autores realizaron el proceso de forma separada e identificaron los desacuerdos, que se resolvieron con el tercero que visitaba nuevamente la fuente y evaluaba las justificaciones. En la sección 4 están las clasificaciones.

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. Algunas de las tablas, al igual que las referencias de los artículos seleccionados se encuentran en el material suplementario [17]. 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 teniendo en cuenta: la estrategia general y los criterios específicos. En la primera clasificación (ver Fig. 1), el 67% de los artículos optaron por un método de selección propio, el 31% se basaron en otra colección y el 2% utilizaron aleatorización.

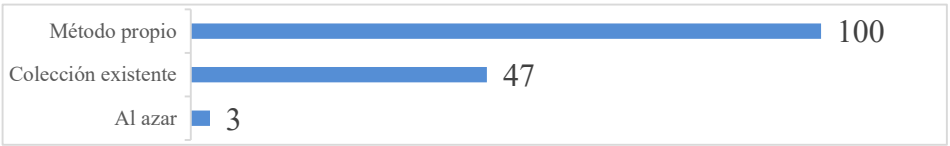

Fig. 1. Método utilizado para elegir los proyectos.

Para la segunda clasificación (ver Tabla A3) se definieron 13 categorías. En total se extrajeron datos de 113 estudios. 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 fuera una migración [P8], proyectos que incluyan una matriz que indique las fallas que cubren los tests [P12], entre otros. El 35% consideraron la actividad del proyecto. La actividad fue medida a través de los años de desarrollo o cantidad de commits. Así hay casos en que se selecciona proyectos con: 3 o más años [P1], con más de 10.000 commits [P71], o más de 32 commits y un año de desarrollo [P50].

El 33% de los estudios seleccionaron proyectos que estuvieran disponibles públicamente con sus meta-datos. El 28% tuvieron en cuenta su tamaño, en función de sus clases [P6, P14, P42], módulos [P17], líneas de código [P75, P99, P118], puntos de función IFPUG [P15] o puntos de función FiSMA [P58]. La popularidad en repositorios públicos fue considerada en el 27% de los estudios, y se tomaron enfoques diferentes para cuantificarla. Por ejemplo, a través de: el número de descargas [P6], la cantidad de usuarios [P7, P45, P56], el número de visitas [P36], entre otras.

En el 23% consideraron el dominio, es decir, la funcionalidad o ámbito de aplicación. Se recolectaron de dominios específicos, como bases de datos [P9], aplicaciones de propósito general [P45], y sistemas de un banco [P18]. En el 20% la condición fue el uso de herramientas para dar soporte a procesos, como: gestión de dependencias [P27, P88], gestión de versiones [P63, P97] o revisión de código [P51, P59].

El 13% buscaron que se haya registrado actividad reciente al momento de recolección, es decir, que estos sean mantenidos. La calidad del proyecto fue considerada en el 10% de los casos. Para establecerla se utilizó: el diseño, la metodología de desarrollo [P48], herramientas [P73, P76, P109], o escalas de calidad como ISBSG [P15].

4.2 RQ2: Características de los proyectos

Los proyectos software en las colecciones fueron clasificados según el lenguaje de programación principal (Fig. 2) y el tipo de software (Fig. 3). En total se registraron 138 artículos que describieron los lenguajes de programación principales. En un 80% fueron construidos en Java, seguido por C con un 25% y C++ con un 20%.

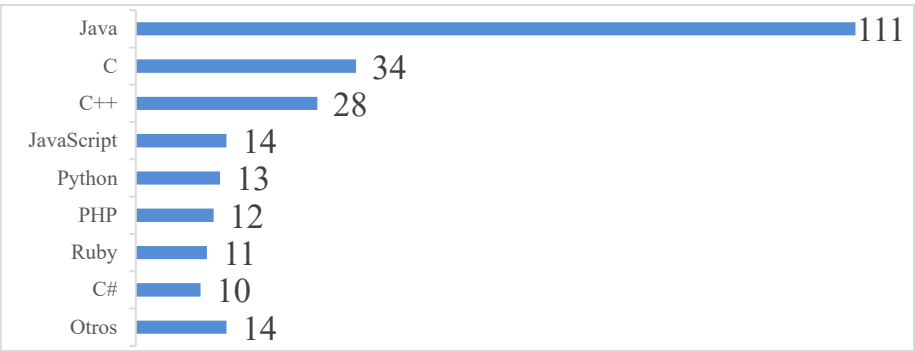

Fig. 2. Lenguajes de programación de los proyectos.

Para identificar los tipos de software se buscaron sus nombres y descripciones. Cada proyecto fue clasificado según la taxonomía de Forward y Lethbridge [25], de donde se emplearon seis categorías (ver Fig. 3). Se usó el término "aplicación de uso general" para agrupar las categorías: software guiado por datos, orientado al consumidor y de propósito general. De los estudios 117 aportaron esta información.

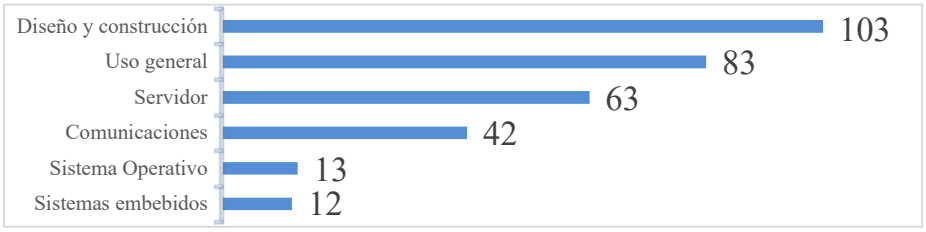

Fig. 3. Tipos de software.

El 88% de los artículos trabajaron con software para el diseño y construcción de sistemas. Por ejemplo, compiladores [P41], frameworks de desarrollo [P8], entornos de desarrollo integrado [P2], entre otros. El 71% emplearon aplicaciones de uso general, como: videojuegos [P44], navegadores [P20], o sitios web [P1], 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]. 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 en dispositivos mecánicos o eléctricos para realizar una función específica. Por ejemplo, software armamentístico [P23], de navegación aérea [P29], para motores de combustión [P34].

4.3 RQ3: Métricas del código

De acuerdo al objeto de estudio del artículo se usaron diferentes métricas del código de los proyectos, en la Tabla 3 se agruparon siguiendo la taxonomía de Elmidaoui et al. [26]. De los 68 estudios que extrajeron métricas, el 66% son métricas de tamaño del código, por ejemplo, líneas de código [P14], o líneas de comentarios [P16].

En el 65% de artículos se utilizan métricas de diseño. Estas miden propiedades inherentes del software y se puede disgregar en otras 7 subcategorías (ver Tabla A4). Las métricas de complejidad calculan la cantidad de caminos existentes en el flujo de control del programa; como la complejidad ciclomática [P26], o el peso de métodos por clase [P11]. Las métricas de acoplamiento cuantifican la 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, por ejemplo: el número de métodos [P14], clases [P44] o atributos [P14]. Las métricas de herencia miden los atributos y métodos compartidos entre clases jerárquicamente, de las cuales están: número de hijos [P11], profundidad de árbol de herencia [P26].

Tabla 3. Categorías de métricas de código.

#	Tipo de Métricas	Artículos
45	Tamaño Código Fuente	P8 P11 P14 P16 P23 P24 P26 P34 P36 P44 P50 P51 P53 P56 P67 P69 P71 P73-P76 P80 P82 P87 P92 P93 P96 P100 P107 P109 P112-P114 P119 P120 P122 P124 P128 P132 P135 P138 P141 P143 P146 P148
44	Diseño	P5 P6 P11 P14 P16 P19 P26 P34-P36 P44 P50 P51 P53-P55 P65 P67 P74-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

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 [P11], la cohesión de clases ajustada [P55] o la cohesión de clases suelta [P14]. Las medidas de encapsulamiento calculan los datos y funciones empaquetadas en una unidad, por ejemplo, el número de accessors [P44], la métrica de acceso a datos [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 [27]. Si bien esta categoría no se encuentra presente en la taxonomía original, 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. En total se reportaron 59 herramientas, las cuales fueron clasificadas de acuerdo a los datos extraídos del código en seis categorías (ver Tabla 4). De los estudios seleccionados, 50 informaron herramientas para generar métricas y artefactos del código, donde el 58% calcularon métricas de código, el 42% extrajeron code smells, el 40% obtuvieron la estructura y dependencias, el 8% refactorizaron código y generaron conjuntos de prueba y el 4% detectaron patrones de diseño.

Tabla 4. 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 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 agruparon en las dos áreas generales de la estadística (ver Tabla 5), tomando como referencia las clasificaciones de Sheskin [28]. En total 109 estudios informaron los procedimientos estadísticos realizados.

Tabla 5. 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. El 77% de los artículos seleccionaron procedimientos inferenciales que se clasificaron en 2 categorías, test no paramétrico y test paramétrico (ver Tabla A5). Una de las diferencias entre estos es que los test paramétricos hacen suposiciones específicas con respecto a uno o más parámetros de la distribución de la población subyacente. Ejemplos de test paramétricos son: test T [P10], test chi cuadrado de Wald [P51], y test de análisis de varianza [P1]. Ejemplos de no paramétricos: test Mann-Whitney [P2], test de rangos de Wilcoxon [P3], y test Shapiro-Wilk [P28].

Por otra parte, el 74% de los artículos trabajó con procedimientos descriptivos, utilizados para presentar y resumir los datos. Estos se clasificaron en cinco categorías (ver Tabla A6). El tamaño de efecto mide la magnitud de fuerza de un fenómeno en un experimento, por ejemplo, el tamaño de efecto de Cliff [P28], el coeficiente de correlación Spearman [P8], o Pearson [P36]. Las medidas de variabilidad y las de tendencia central son estadísticos básicos. Ejemplos de medidas de variabilidad son: la desviación estándar [P46], los cuartiles [P14] y la varianza [P108]. En el caso de medidas de tendencia central: 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.

5.1 RQ1: Criterios de selección

Para abordar esta pregunta se buscaron evidencias de la selección de proyectos software desde la estrategia general y los criterios específicos. En la mayoría (67%) de los casos los investigadores siguen lineamientos propios y, en la segunda clasificación se identificaron 13 categorías. Esto evidencia la diversidad de criterios y la ausencia de un procedimiento estándar para la construcción de colecciones de proyectos.

Un análisis que puede extenderse de los datos recolectados es la estrategia de muestreo. Aquellos estudios primarios que utilizaron lineamientos propios y no informaron criterios de selección para recolectar los proyectos podrían relacionarse con el muestreo por conveniencia (15 estudios). Donde los sujetos se eligen arbitrariamente o en función de su cercanía, disponibilidad o facilidad de estudio [29]. El principal problema con este enfoque es que amenaza la generalización de los resultados.

Los casos en que los autores establecieron criterios de selección, y los recolectaron de forma cuidadosa pero no aleatoria, o usaron colecciones construidas por otros autores (129 estudios), podrían asociarse con el muestreo con propósito. Donde los elementos se eligen de acuerdo con alguna lógica o estrategia [29]. Seis estudios emplearon técnicas de muestreo probabilístico o utilizaron colecciones que las aplicaron.

5.2 RQ2: Características de los proyectos

Para caracterizar la muestra de los proyectos recolectados por los grupos de investigación se identificó el lenguaje de programación. De 138 artículos que reportaron el lenguaje de programación principal de los proyectos, la mayoría (80%) trabajaron con Java. Esto puede tener una relación directa con la cantidad de proyectos de fuente abierta en estos lenguajes en repositorios de software como Github [30]. Otra razón es la gran cantidad de herramientas de análisis estático del código compatibles con Java.

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 (45%), evidenciando el uso no generalizado de métricas de código en estudios empíricos.

El resto de los trabajos se valieron de otras fuentes. Por ejemplo, los reportes de defectos para calcular su tiempo de resolución [P1, P46, P56], clasificarlos [P7, P33, P52] o detectar duplicados [P47]. También la información de ejecución del programa para localizar funciones específicas [P2], clasificar procesos [P18], o estudiar la asignación de memoria [P40]. Otras fuentes 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 [P29], entre otros.

5.4 RQ4: Herramientas

El objetivo de esta pregunta es conocer las herramientas utilizadas en estudios empíricos para la recolección de artefactos relacionados con el código fuente de proyectos.

Es notable que solamente 50 artículos (33%) mencionan de manera explícita los nombres de las herramientas utilizadas. Es un requisito necesario para la replicación de los estudios empíricos el reporte claro de las herramientas.

En muchas ocasiones informan el modelo, técnica, procedimiento o algoritmo, pero no la herramienta. Por ejemplo, Corazza et al. [P3] implementan un modelo de aprendizaje automatizado para estimar el esfuerzo de desarrollo. Chen et al. [P18] declararon el uso de un "método propio" para refactorización. Dallal y Morasca [P14] usaron una herramienta con 29 métricas de código, pero no reportaron su nombre.

5.5 RQ5: Análisis estadísticos

Esta pregunta pretende determinar qué procedimientos estadísticos son elegidos para validar los resultados de los estudios empíricos. La aplicación apropiada de los métodos de análisis permite garantizar que la evidencia se presente correctamente y así evitar las interpretaciones erróneas de los resultados [31].

El 29% de los estudios emplearon tests estadísticos paramétricos, esto implica supuestos como que los datos obtenidos provengan de una distribución normal, lo que podría no coincidir con la distribución subyacente de la población. 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 (45%) analizaron el tamaño del efecto. Esta funciona como medida objetiva de la importancia que tiene un fenómeno en un experimento [32], y se ve menos influenciada por el tamaño de la muestra que la significación estadística, siendo una recomendación en los casos que las muestras sean pequeñas.

5.6 Implicancias

En los estudios empíricos en la 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 [33].

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 [34]. 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 [35]. Otra forma de producir muestras representativas es mediante el muestreo con propósito.

Finalmente, para reducir la amenaza de producir resultados no generalizables, los profesionales e investigadores deben evitar consumir colecciones en los que no se informe el procedimiento de selección. Al construir una colección no es suficiente compartir solo los proyectos, sino también el procedimiento para recolectarlos.

6 Amenazas a la validez

A continuación, se discuten las amenazas a la validez del estudio siguiendo el enfoque propuesto por Runeson y Höst [35]. Se consideraron cuatro aspectos de validez.

- Validez de constructo. Refleja hasta qué punto la metodología de la investigación representa 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 se disminuyó implementando una prueba piloto de la tabla, tomando artículos al azar para completar una primera versión, y modificándola iterativamente según fue necesario hasta lograr la versión final.
- Validez interna. Analiza los riesgos cuando se estudian relaciones causales [36]. Elegir los artículos y el período de tiempo adecuados son factores importantes que afectan la validez interna. Para mitigar esta amenaza, se seleccionaron artículos de foros específicos en la ISBE, que son ampliamente aceptados en el ámbito de la Ingeniería del Software en términos de alcance y reputación. La subjetividad en la selección se disminuyó mediante el proceso de la Sección 3.1, realizando tantas iteraciones hasta obtener un alto grado de acuerdo.
- 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. 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 [35]. En este artículo, los métodos y procesos de investigación se describen para garantizar su reproducibilidad y en el material suplementario se incluye la lista de artículos seleccionados.

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 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. Es posible que no se hayan localizado todos los estudios relevantes, por esa razón el proceso fue desarrollado siguiendo un protocolo bien definido. Dicho esto, por medio de una búsqueda manual en EMSE, 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.

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 y C/C++; y el tipo de software utilizado fue: para construcción de sistemas, de uso general y servidores. 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. Los análisis estadísticos más frecuentes fueron pruebas estadísticas no paramétricas, y 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 empíricos, es indispensable que los autores faciliten los proyectos software y las herramientas empleadas para generar los resultados, ofreciendo guías y asistencia a los investigadores. Con respecto a la construcción de muestras representativas, se recomienda emplear técnicas de muestreo probabilístico para seleccionar los proyectos. Es esencial recopilar los proyectos de software utilizando un marco claro y sistemático.

Finalmente, como aporte de este trabajo se identificaron pautas que ayudan a sistematizar la selección de proyectos en la construcción de colecciones. En líneas generales, las reglas que se pueden reunir son: código fuente libre de acceso y distribución libre, proyectos vigentes con historia de desarrollo, que sean populares y estén construidos en Java. La colección resultante debería conservar el código, las métricas obtenidas del análisis estático y las herramientas empleadas. Como trabajos futuros se analizarán las colecciones más populares, con el objetivo de reconocer cuales fueron los criterios considerados, 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: LNCS. pp. 108–124. (1996).
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 ICSE. pp. 273–281 (2004).
6. Tempero, E., Anslow, C., Dietrich, J., Han, T., Li, J., Lumpe, M.,: The Qualitas Corpus: A curated collection of Java code for empirical studies. In APSEC. pp. 336–345 (2010).
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: MSR 2013. pp. 207–216 (2013).
9. Keivanloo, I., Rilling, J., Zou, Y.: Spotting working code examples. In: Proceedings - ICSE. pp. 664–675. IEEE Computer Society (2014).
10. Zerouali, A., Mens, T.: Analyzing the Evolution of Testing Library Usage in Open Source Java Projects. In: 2017 IEEE 24th SANER. pp. 417–421 (2017).
11. Goeminne, M., Mens, T.: Towards a Survival Analysis of Database Framework Usage in Java Projects. In: ICSME 2015. pp. 551–555 (2015).

12. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic Mapping Studies in Software Engineering. In: Proceedings of the 12th EASE. pp. 68–77 (2008).
13. Gyimesi, P., Gyimesi, G., Tóth, Z., Ferenc, R.: Characterization of Source Code Defects by Data Mining Conducted on GitHub. In: Lecture Notes in Comp. Sc. Springer, (2015).
14. Chávez, A., Ferreira, I., Fernandes, E., Cedrim, D., Garcia, A.: How does refactoring affect internal quality attributes?: A multi-project study. In: ACM ICPS. pp. 74–83. (2017).
15. Petersen, K., Vakkalanka, S., Kuzniarz, L.: Guidelines for conducting systematic mapping studies in software engineering: An update. In: Infor. and Soft. Tech.. pp. 1–18. (2015).
16. 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).
17. Material suplementario, https://bit.ly/Mapeo-JAIIO-Material-Supl, last accessed 2022/05/22.
18. 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).
19. Molléri, J.S., Petersen, K., Mendes, E.: Survey Guidelines in Software Engineering: An Annotated Review. In: ESEM. IEEE Computer Society (2016).
20. Storey, M.A., Ernst, N.A., Williams, C., Kalliamvakou, E.: The who, what, how of software engineering research: a socio-technical framework. EMSE. 25, 4097–4129 (2020).
21. Kitchenham, B., Brereton, P.: A systematic review of systematic review process research in software engineering, (2013). https://doi.org/10.1016/j.infsof.2013.07.010.
22. Saldaña, J.: The Coding Manual for Qualitative Researchers. SAGE Publications (2015).
23. Crabtree, B.F., Miller, W.L.: Doing Qualitative Research. SAGE Publications, Inc (1999).
24. 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).
25. Forward, A., Lethbridge, T.C.: A Taxonomy of Software Types to Facilitate Search and Evidence-Based Software Engineering. (2008).
26. Elmidaoui, S., Cheikhi, L., Idri, A.: Towards a Taxonomy of Software Maintainability Predictors. In: Advances in Intelligent Systems and Comp. pp. 823–832. Springer (2019).
27. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the Design of Existing Code. (2002).
28. Sheskin, D.J.: Parametric and nonparametric statistical procedures second edition. (2000).
29. Baltes, S., Ralph, P.: Sampling in Software Engineering Research: A Critical Review and Guidelines. (2020).
30. GitHut 2.0, https://madnight.github.io/githut, last accessed 2022/05/22.
31. Wohlin, C., Rainer, A.: Challenges and recommendations to publishing and using credible evidence in software engineering. Inf. Softw. Technol. 134, 106555 (2021).
32. Kitchenham, B., Madeyski, L., Budgen, D., Keung, J., Brereton, P., Charters, S.,: Robust Statistical Methods for Empirical Software Engineering. EMSE 22, 579–630 (2017).
33. Cosentino, V., Luis, J., Izquierdo, C., Cabot, J.: Findings from GitHub: Methods, datasets and limitations. In: Proceedings MSR 2016. pp. 137–141. ACM, Inc (2016).
34. Munaiah, N., Kroh, S., Cabrey, C., Nagappan, M.: Curating GitHub for engineered software projects. Empir. Softw. Eng. 22, 3219–3253 (2017).
35. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14, 131–164 (2009).
36. Siegmund, J., Siegmund, N., Apel, S.: Views on internal and external validity in empirical software engineering. In: Proceedings - ICSE. pp. 9–19. IEEE Computer Society (2015).