MODELO PARA CURADURÍA DE PROYECTOS SOFTWARE DE FUENTE ABIERTA PARA ESTUDIOS EMPÍRICOS EN INGENIERÍA DE SOFTWARE

Emanuel Irrazabal; Juan Andrés Carruthers; Juan Alberto Pinto Oppido

Facultad de Ciencias Exactas y Naturales y Agrimensura.
Universidad Nacional del Nordeste
{eirrazabal, jacarruthers}@exa.unne.edu.ar, juan_al_pinto@hotmail.com

RESUMEN

Este es el cuarto año del proyecto F018-2017; una continuación de los proyectos F07-2009 y F10-2013, ambos enfocados en modelos, métodos y herramientas para la calidad del software. Este proyecto se enfoca en el estudio de la calidad del Software desde el punto de vista del código fuente por medio de métricas e indicadores cuantificables en este. Esta línea de investigación financiada por CONICET a través de una beca interna doctoral, busca generar métodos empíricos que contribuyan a identificar aquellos factores o atributos vinculados directamente con la calidad del producto software.

En particular, se está trabajando la construcción de un catálogo de proyectos Software de calidad. Esto ha incluido el desarrollo de un mapeo sistemático para reconocer las características de los proyectos utilizados en estudios. Se atiende la necesidad de los grupos de investigación de obtener muestras curadas de proyectos imprescindibles para la generación de resultados confiables y generalizables en la experimentación de estudios empíricos de la calidad de Software, proporcionando los insumos y procedimientos necesarios para conseguirlo de manera efectiva.

Palabras clave: calidad de software, proyecto fuente abierta, catálogo de proyectos.

CONTEXTO

Las líneas de Investigación y Desarrollo presentada en este trabajo corresponden al proyecto PI-17F018 “Metodologías y herramientas emergentes para contribuir con la calidad del software”, acreditado por la Secretaría de Ciencia y Técnica de la Universidad Nacional del Nordeste (UNNE) para el periodo 2018-2021, y a la beca interna doctoral de CONICET otorgada por RESOL-2021-154-APN-DIR#CONICET para el período 2021-2025.

1. INTRODUCCIÓN

En la Ingeniería de Software se trabaja mayoritariamente con la construcción de aplicaciones software multi-versión [1]. Por lo tanto, muchas de las actividades asociadas con una aplicación software provocan revisiones para mejorar la funcionalidad o para corregir errores, especialmente en las metodologías ágiles [2].

En el desarrollo software, la calidad puede estudiarse desde el punto de vista de: i) la calidad del proceso de desarrollo software, y ii) la calidad del código fuente [3]. En este último caso es necesario obtener métodos empíricos para demostrar la calidad del software [4] y utilizar evidencia directamente relacionada con el producto software resultante a partir de métricas e indicadores que se vinculen directamente con la calidad [5]. Sin embargo, esto no siempre es sencillo de realizar, ya que las conclusiones generales de los estudios

empíricos en Ingeniería de Software a menudo dependen de una gran cantidad de variables [6].

El uso masivo de repositorios para el código fuente (por ej., SourceForge, GitHub o Maven) le ha otorgado a los investigadores e ingenieros de software el acceso a millones de proyectos y, por lo tanto, datos para el desarrollo de estudios empíricos [7] [8] [9]. No obstante, la proporción de ruido en una muestra aleatoria tomada de repositorios podría sesgar el estudio, y puede llevar a los investigadores a conclusiones poco realistas, potencialmente inexactas [10].

Esto se contrapone con la condición de reproductibilidad y generalidad de los resultados empíricos, tal y como lo indica el énfasis actual en la Ingeniería de Software basada en evidencia. En particular, la reproductibilidad es una condición necesaria, no solo en publicaciones en revistas o conferencias de prestigio de la disciplina, sino también para las empresas de desarrollo de software que a menudo desean analizar la evolución de sus propios proyectos o como patrón comparativo en auditorías de software. En este contexto, una práctica para demostrar la efectividad de las métricas como predictores de las características de calidad del software es la construcción de los denominados corpus o catálogos de proyectos [11].

Los catálogos de proyectos son un insumo para los grupos de trabajo y sirven como mecanismo de comparación para distintos tipos de experimentos. Existen varios ejemplos en la literatura, como los realizados por Barone y R. Sennrich [12], Allamanis y Sutton [13] o Keivanloo [14] y se diferencian por la cantidad, calidad de proyectos que lo componen; como también los criterios y métodos para agruparlos.

Un catálogo popular en Ingeniería de Software es el Qualitas corpus [11], cuya última versión es de 2013. En general, se observa que no existe información actualizada en la mayoría de los catálogos, lo que hace necesario revisar los proyectos y sus versiones, y generar nuevas métricas sobre ellos. Una alternativa interesante para ello es la utilización de herramientas de código abierto, que han demostrado ser de gran utilidad en entornos de trabajo profesionales, tanto tradicionales [15] como aplicadas a nuevas técnicas de desarrollo [16].

Por lo tanto, la línea de trabajo del proyecto tiene que ver con las características de calidad del código fuente del software.

2. LÍNEAS DE INVESTIGACIÓN Y DESARROLLO

A continuación, se describe la línea de investigación y desarrollo:

Curaduría de proyectos

Se están estudiando los aspectos metodológicos y criterios considerados por la comunidad científica para conformar los catálogos de proyectos. Para esta etapa de documentación se utilizarán dos métodos de investigación. Por un lado, el método de revisiones sistemáticas [17] para identificar, evaluar, e interpretar toda la información relativa a un tema de investigación en particular, de un modo sistemático y replicable. Como segundo método de documentación se utilizarán las encuestas [18], para recopilar información de los grupos de investigación y los equipos de trabajo de las empresas privadas.

El siguiente paso consiste en la creación de un modelo de procedimientos para la construcción, mantenimiento y curaduría de un cuerpo de proyectos software y sus métricas de calidad de producto. Teniendo como fin proveer una estrategia para la construcción de estudios empíricos en Ingeniería del Software que brinde una mejor reproducibilidad, consistencia experimental, y flexibilidad para una evolución gestionada del cuerpo de proyectos en el tiempo. Y, finalmente, se

propone implementar el modelo en un ambiente real de trabajo.

3. RESULTADOS OBTENIDOS/ESPERADOS

En el marco de este proyecto y respecto de la línea de curaduría de proyectos se lograron los siguientes objetivos:

1. Relevamiento de información a considerar en la selección de proyectos Software y metadatos utilizados en el campo de estudio.
   En primer lugar, se realizó un mapeo sistemático (SMS) con el objetivo de identificar las características de los proyectos Software, tipos de metadatos consumidos, distintas herramientas usadas para recolectar los metadatos y análisis posteriores realizados a los mismos tenidos en cuenta por la literatura de la ingeniería del Software.

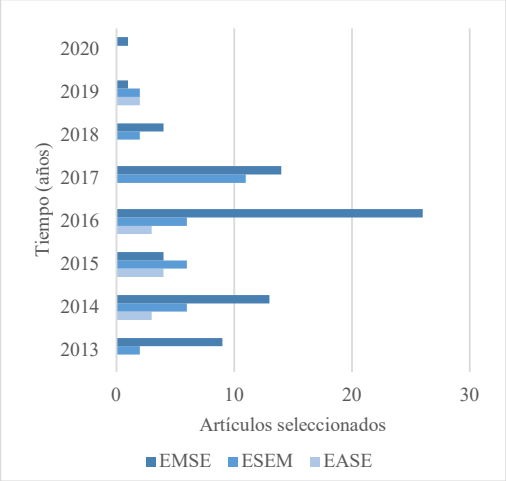

Fig. I: Distribución de los artículos seleccionados de cada revista por año.

   Las fuentes seleccionadas fueron la revista Empirical Software Engineering (ESE) 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. En Fig. I se puede observar la distribución de artículos recolectados por revista. El estudio fue realizado en el marco de una beca de pregrado de la UNNE.
2. Tesis de grado y desarrollo de herramienta para extracción de metadatos de plataforma de análisis estático de código fuente.

Fig. II: Página de Inicio SET.

Se construyó la aplicación web “Sonar Exporting Tool” (SET) para extraer las métricas de la plataforma Sonar Cloud (SC) en formatos de datos consumibles y difundir el código fuente de los proyectos Software analizados. Todos los proyectos seleccionados forman parte del Qualitas Corpus [11]. Las Fig. II y Fig. III son capturas de la aplicación.

SET permite exportar medidas de métricas de SC en los formatos de datos csv, json y xml; la actualización automática y manual de la base de datos. También se creó un procedimiento para el análisis automatizado de colecciones de proyectos por medio del cliente SonarScanner.
Además, se busca mantener actualizado cada uno de los proyectos dentro de la aplicación en su versión estable más reciente.

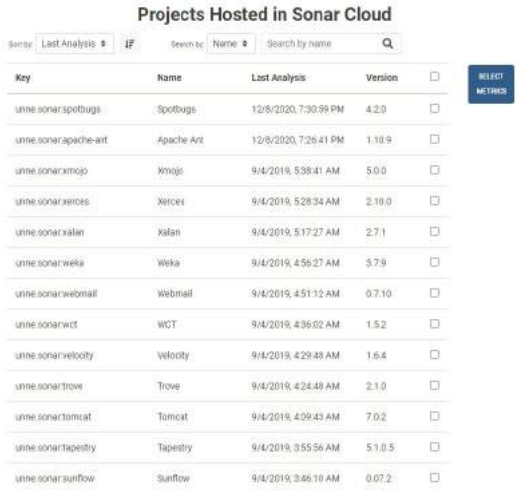

Fig. III: Listado de proyectos en SET.

4. FORMACIÓN DE RECURSOS HUMANOS

En esta línea de trabajo del Grupo de Investigación sobre Calidad de Software (GICS) están involucrados 3 docentes investigadores, un becario interno doctoral de CONICET y un becario de investigación de pregrado.

5. REFERENCIAS

[1] Parnas, D. L. (2001). Some software engineering principles. Hoffman, D. M. y Weiss, D. M. (Ed), Software fundamentals: collected papers by David L. Parnas (pp. 257–266). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

[2] Irrazabal, E., Vásquez, F., Díaz, R., & Garzás, J. (2001) Applying ISO/IEC 12207: 2008 with SCRUM and Agile methods. In International Conference on Software Process Improvement and Capability Determination (pp. 169-180). Springer, Berlin, Heidelberg.

[3] Lehman, M. M. (1996). Laws of software evolution revisited. En Montangero C. (Ed), Laws of Software Evolution Revisited (pp. 108-124). Berlin, Germany: Springer-Verlag.

[4] Kitchenham, B. y Pfleeger, S. L. (1996). Software quality: the elusive target. IEEE Software, 13(1), 12-21.

[5] Garvin, D. A. (1984). What Does "Product Quality" Really Mean?. MIT Sloan Management Review, 25-43.

[6] Basili, V.R., Shull, F., and Lanubile, F. Building knowledge through families of experiments. Software Engineering, IEEE Transactions on, 25 (1999), 456--473.

[7] Vidal, S, Bergel, A., Marcos, C., Díaz Pace, J. A.: Understanding and addressing exhibitionism in Java empirical research about method accessibility. Empirical Software Engineering 21(2): 483-516 (2016)

[8] Vidal, S., Bergel., A., Díaz Pace, J. A., Marcos, C.: Over-exposed classes in Java: An empirical study. Comput. Lang. Syst. Struct. 46: 1-19 (2016)

[9] Vázquez, H., Bergel, A., Vidal, S., Díaz Pace, J. A., Marcos, C.: Slimming javascript applications: An approach for removing unused functions from javascript libraries. Inf. Softw. Technol. 107: 18-29 (2019)

[10] Munaiah, N.. Curating github for engineered software projects. Empirical Software Engineering, 2017, vol. 22, no 6, p. 3219-3253.

[11] Tempero, E., Anslow, C., Dietrich, J., Han, T., Li, J., Lumpe, M., Melton, H y Noble, J. (2010). The Qualitas Corpus: A curated collection of Java code for empirical studies. IEEE, 336-345.

[12] Barone, A. V. M., y Sennrich, R. (2017). A parallel corpus of Python functions and documentation strings for automated code documentation

and code generation. arXiv preprint arXiv:1707.02275.

[13] Allamanis, M., y Sutton, C. (2013). Mining source code repositories at massive scale using language modeling. IEEE Press, 207-216.

[14] Keivanloo, I., Rilling, J., y Zou, Y. (2014). Spotting working code examples. ACM, 664-675.

[15] Irrazábal, E., Garzás, J., & Marcos, E. (2011). Alignment of Open Source Tools with the New ISO 25010 Standard-Focus on Maintainability. In International Conference on Software and Data Technologies (Vol. 2, pp. 111-116).

[16] Mascheroni, M. A., & Irrazábal, E. (2018). Continuous testing and solutions for testing problems in continuous delivery: A systematic literature review. Computación y Sistemas, 22(3), 1009-1038.

[17] Kitchenham, B. (2004). Procedures for performing systematic reviews. Keele University, 33(2004), 1-26.

[18] Kitchenham, Barbara, and Shari Lawrence Pfleeger. "Principles of survey research." ACM SIGSOFT Software Engineering Notes 27.5 (2002): 17-20.