DESARROLLO DE ESTUDIOS EMPÍRICOS EN INGENIERÍA DE SOFTWARE

Emanuel Irrazabal; Rubén Alfredo Bernal; Juan Andrés Carruthers; Andrea Lezcano Airaldi; Yanina Medina; Iván Sambrana; Celeste Ojeda Rodríguez

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

RESUMEN

Este es el primer año del proyecto F01-2021; una derivación de los proyectos F07-2009, F10-2013 y F018-2017, que desarrollaron modelos, métodos y herramientas para la calidad del software. Este proyecto se enfoca en el diseño y desarrollo de estudios empíricos de calidad de software, como un insumo para la Ingeniería de Software basada en Evidencia.

La línea principal de investigación es la construcción de un catálogo de proyectos software de calidad. Se atiende la necesidad de los grupos de investigación para 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.

Otras líneas secundarias de investigación son el desarrollo de estudios empíricos para la toma de decisiones en grupo y el desarrollo de estimaciones por juicio experto mediadas a partir de tecnologías móviles.

Palabras clave: calidad de software, estudios empíricos, proyecto fuente abierta.

CONTEXTO

Las líneas de Investigación y Desarrollo presentada en este trabajo corresponden al proyecto PI-21F01 “Desarrollo de estudios empíricos en Ingeniería del Software”, acreditado por la Secretaría de Ciencia y Técnica de la Universidad Nacional del Nordeste (UNNE) para el periodo 2022-2025, 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 del Software (IS) se trabaja 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 IS 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]-[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 IS 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 IS 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 IS 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 y de acuerdo con lo desarrollado en el año 2.021 en el marco del proyecto de

investigación anterior 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.
   Se realizaron dos mapeos sistemáticos, uno enfocado en reunir evidencias sobre estudios empíricos en IS realizados con colecciones de proyectos, y el otro en el uso colecciones existentes en la literatura. El objetivo en ambos casos es identificar las características de los proyectos Software, tipos de metadatos consumidos y las distintas herramientas usadas para recolectar los metadatos de las colecciones. En total se revisaron 5.000 artículos científicos de las principales revistas y congresos sobre la temática.

2. Desarrollo de herramientas para dar soporte en la generación de métricas obtenidas del análisis estático del código fuente.
   Sonar Exporting Tool (SET), aplicación web 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. 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.
   Sonar Juploader, aplicación java que permite analizar automáticamente un lote de proyectos con SC por medio del cliente SonarScanner. La aplicación da soporte a la gestión y uso de organizaciones de SC, pre-configuración de proyectos, análisis de proyectos, y visualización de reportes [19].

3. Se realizó un experimento en donde se evaluaron tres modelos predictivos para la detección temprana de defectos en proyectos software [20]. Los tres clasificadores fueron entrenados con tres conjuntos de datos conformados por valores de métricas orientadas a objetos obtenidas del análisis estático del código de cinco sistemas Java. Para el entrenamiento de cada modelo se utiliza el método de ensamble validación y votación.
   En este sentido, también se está trabajando en la construcción de un curso que facilite a alumnos de posgrado conceptos esenciales para la experimentación en IS, como también otras áreas del conocimiento. Desde la planificación y diseño de un experimento, hasta la utilización de herramientas estadísticas y visuales para reportarlos de manera efectiva.

4. FORMACIÓN DE RECURSOS HUMANOS

En esta línea de trabajo del Grupo de Investigación sobre Calidad de Software (GICS) están involucrados 3 docentes investigadores, dos becarios internos doctorales de CONICET, un becario de investigación de pregrado y un profesional licenciado en sistemas de información.

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. (2011). 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., & Lanubile, F. (1999). Building knowledge through families of experiments. IEEE Transactions on Software Engineering, 25(4), 456-473.

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

[8] Vidal, S., Bergel, A., Díaz-Pace, J. A., & Marcos, C. (2016). Over-exposed classes in Java: An empirical study. Computer Languages, Systems & Structures, 46, 1-19.

[9] Vázquez, H. C., Bergel, A., Vidal, S., Pace, J. D., & Marcos, C. (2019). Slimming javascript applications: An approach for removing unused functions from javascript libraries. Information and software technology, 107, 18-29.

[10] Munaiah, N., Kroh, S., Cabrey, C., & Nagappan, M. (2017). Curating github for engineered software projects. Empirical Software Engineering, 22(6), 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, B., & Pfleeger, S. L. (2002). Principles of survey research. ACM SIGSOFT Software Engineering Notes.

[19] Carruthers, J. A., & Ojeda Rodríguez, C. (2021). Evaluación de conjuntos de datos utilizados

en la construcción de modelos para la predicción de defectos en clases de proyectos software. In IV Jornadas de Calidad de Software y Agilidad
(pp. 7-16).

[20] Pinto Oppido, J. A., Carruthers, J. A., & Irrazábal, E. A. (2021). Sonar JUploader, aplicación para el análisis, sincronización y actualización automática de proyectos a Sonar Cloud. In IV Jornadas de Calidad de Software y Agilidad
(pp. 27-36).