ARTÍCULOS REFACTORING

Universidad Carlos III

Grado Ingeniería Informática 2022-23

Desarrollo Software (Grupo 81)

Práctica realizada por:

- **Jaime Vaquero Rabahieh** (NIA: 100472248, Email: 100472248@alumnos.uc3m.es)
- **Alejandro Díaz Cuéllar** (NIA: 100472173, Email: 100472173@alumnos.uc3m.es)

ÍNDICE

1) ARTÍCULOS JAIME

1.1) PRIMERO

Referencia: H. K Khosravi, A. Rassoulzadegan, “A Meta-Learning Approach for Software Refactoring”, Cornell University, 19/01/2023, https://arxiv.org/abs/2301.08061

Resumen: Es un texto en el que sus autores indican las desventajas del refactoring realizado por personas y por máquinas y proponen como solución un algoritmo metalingüístico (MAML) que ha pasado de forma exitosa las pruebas experimentales (91% de acierto).

Opinión: Considero este artículo relevante debido a que representa lo que va a convertirse el futuro del refactoring: el uso de métodos y algoritmos con un pequeño margen de error que, en caso de fallar en una situación, serán sustituidos por otros con mayor eficiencia.

1.2) SEGUNDO

Referencia: H. Muneer Yahya, D. B. Taha, “Software Code Refactoring: A Comprehensive Review”, Journal of Education and Science, 01/03/2023, https://www.iasj.net/iasj/download/dfc9ea6567d57e1a

Resumen: Para analizar el uso del refactoring estos días, se han elegido 17 proyectos de software de entre 2014 y 2021 de los que sacaron los métodos de refactoring utilizados. Los resultados indicaron que cada autor realizó un algoritmo distinto que engloba no solo limpiar el código sino también el diseño y los requerimientos. Además, parte de esos métodos fueron publicados recientemente (entre 2014 y 2021 lanzaron 6 nuevos algoritmos, casi 1 por año). Por lo tanto, llegaron a la conclusión de que el refactoring hoy en día se basa en crear algoritmos que ayuden a limpiar los proyectos.

Opinión: Lo que más destaco de este artículo es como representan lo que empezó siendo el refactoring y como ha cambiado hoy día: de simplificar código a utilizar algoritmos de limpieza. Desde siempre, la forma más útil de resolver problemas ha sido con la creación de fórmulas, procesos y algoritmos de resolución. Tan eficiente que incluso algo que parece tan simple como es la limpieza de código se prefiere resolver mediante algoritmos.

2) ARTÍCULOS ALEJANDRO

2.1) PRIMERO

Referencia: Hieke Keuning, Bastiaan Heeren, Johan Jeuring, “A Tutoring System to Learn Code Refactoring”.
Publicado durante el SIGCSE '21: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education el día 05/03/2021.

La referencia a la publicación es: A Tutoring System to Learn Code Refactoring | Proceedings of the 52nd ACM Technical Symposium on Computer Science Education

Resumen: Aunque las clases, cursos y tutoriales de programación están a la orden del día, es bastante infrecuente que se le inculquen a los alumnos nociones de refactoring. En este articulo se explica el concepto de refactoring y mantener la buena calidad del código, ejemplificando las herramientas disponibles para el mismo así como plataformas para aprender refactoring. Pero aparte de esto, el artículo tiene como objetivo proporcionar un nuevo sistema para aprender a enseñar refactoring de manera adecuada, algo que consideran necesario para avanzar en esta cuestión.

Opinión: Me parece un artículo muy interesante y, desgraciadamente, necesario. Es cierto que apenas se nos ha mencionado el refactoring o técnicas relacionadas con el mismo y no se le ha dado importancia en las asignaturas de programación que hemos tenido en la carrera así que sí que creo que puede ser un tema que se suele pasar un poco por encima. Es por ello que opino que cuanta más importancia se le de mejor, porque es muy importante el buen mantenimiento del código y también aumentar su legibilidad y optimización.

2.2) SEGUNDO

Referencia: Abdullah Almogaheda, Mazni Omarb, Nur Haryani Zakaria, “Refactoring Codes to Improve Software Security Requirements”.
Publicado en Procedia Computer Science, International Conference on Industry Sciences and Computer Science Innovation en Agosto de 2023.
Link al paper: (PDF) Refactoring Codes to Improve Software Security Requirements

Resumen: El refactoring es una de las técnicas más extendidas en programación para el buen mantenimiento, legibilidad y testing del código. Sin embargo, no hay demasiada información de su impacto en seguridad. Para solventar esto, se ha llevado a cabo un estudio en el que se comprueba el impacto de una serie de técnicas de refactoring: Extract Method, Incline Method, Encapsulate Field, Remove Setting Method y Hide Method en una serie de entornos como un sistema de manejo de un banco, midiendo una serie de parámetros concretos.
Los resultados muestran que no siempre el uso de una técnica de refactoring puede ayudar en cuanto a la seguridad, y que, por tanto hay que estudiar bien las técnicas de refactoring y aplicarlas con cabeza, midiendo su posible impacto en todas las facetas del software.

Opinión: Me parece un artículo muy interesante puesto que no se me había ocurrido que el uso del refactoring (o de ciertas técnicas) pudiera ser negativo. Por tanto está claro que no se trata de aplicar categóricamente técnicas de refactoring, sino hacerlo con cabeza y sabiendo qué estás haciendo y por qué lo haces. En ese sentido me ha parecido un artículo bastante enriquecedor.