uc3m | Universidad Carlos III de Madrid


# *Ejercicio Guiado 4*

## *Desarrollo de Software*

*Ingeniería informática - Grupo 81*

| **Fecha** | **02/05/2022** |
|---|---|
| **Autores** | Francisco Antonio Gallardo Fuentes - 100451146<br>100451146@alumnos.uc3m.es |
| | Ángel del Pozo Manglano - 100442123<br>angel.pozo@alumnos.uc3m.es |

# Índice:

## Energy Efficiency Analysis of Code Refactoring Techniques for Green and Sustainable Software in Portable Devices

En este artículo se nos habla del mantenimiento general que nos supone usar técnicas de refactoring en nuestro código, ya sea el mantener durante más tiempo aplicaciones, dispositivos electrónicos como laptops o dispositivos móviles e incluso programas informáticos.
También se nos habla de diferentes técnicas como Encapsulate Field, Data Collection entre otras, muchas de ellas vistas en clase y otras que no.
El artículo es interesante porque hace un experimento en el que va haciendo combinaciones de diferentes técnicas y su consecuencia energética en las aplicaciones móviles quedando como conclusión que las combinaciones de las técnicas reducen el consumo energético de las aplicaciones y su rendimiento a futuro.

## Analysis of Code Refactoring Impact on Software Quality: A Scientific Explanation

Para enlazar el artículo anterior en el cual se hacía un experimento de las técnicas de refactoring, en este artículo nos hace una explicación científica de lo que es el refactoring en el código y su impacto.
Este artículo es interesante porque te explica formalmente las consecuencias del uso del refactoring en el código, ya sea su complejidad, limpieza, eficiencia, costes.
Y todo ello habla en base científica, por lo que podemos afirmar que lo que se dice es cierto.

## Refactoring for Parameterizing Java Classes

Aunque estamos estudiando Python, siempre es interesante ver cómo funciona el refactoring en otras clases de lenguaje como lo es Java.
En este artículo se nos habla del problema de la parametrización en las librerías de Java.
La parametrización consiste en añadir TIPOS de parámetros cuyo significado pueda concordar en varios contextos.
La solución que proponen es agregar parámetros de tipo a las declaraciones de clase no genéricas existentes.
Para demostrar que esto funciona, los autores han hecho un estudio en la plataforma Eclipse y comprobaron que su efectividad, precisión y corrección. Además indican que los resultados son más precisos que los usados por bibliotecas.

## Making Program Refactoring Safer

Para finalizar, en este último artículo se comenta una forma de hacer una refactorización segura.
Según se comenta, hay muchos casos en los que la refactorización se ha hecho mal y no alberga todos los posibles casos, entonces al hacer un mínimo cambio, esto puede generar un error en el código.
Es por ello que los autores desarrollaron un complemento de Eclipse llamado SafeRefactor que recibe un código fuente y una refactorización definida por el usuario como valores de entrada y se informa por salida si se puede hacer o no la refactorización.
Obviando toda la parte del desarrollo del complemento, los autores realizaron diferentes pruebas (duplicación del código, extracciones…) y técnicas como revisión por parejas o pruebas unitarias para ver si este complemento funciona, cuyos resultados fueron beneficiosos en la mayoría de los casos.

# Conclusiones

El refactoring en la gran mayoría de los artículos se da a entender como una técnica de desarrollo muy eficaz.
Al principio de la práctica cuando estábamos realizando técnicas de refactoring, pensábamos que esto lo único que hacía era ensuciar el código añadiendo muchas clases, métodos y demás, aunque ahora pensamos que tiene su utilidad (exceptuando algunos detalles que seguimos creyendo que no tiene sentido como la creación de clases exclusivamente para un solo método).
El desarrollo de la práctica fue positivo y estamos seguros de que en prácticas futuras aplicaremos indirectamente muchas de las técnicas y gastaremos un parte tiempo en limpiar el código