Análisis de la Mantenibilidad: Caso práctico

Ejercicio sesión seminario 21 de febrero ACG

Curso 2022-2023

Se han propuesto cientos de métricas para la evaluación de calidad de software, pero no todas proporcionan un soporte práctico para el desarrollador de software[1]. Algunas demandan mediciones que son demasiado complejas, otras son tan esotéricas que pocos profesionales tienen la esperanza de entenderlas y otras violan las nociones básicas intuitivas de lo que realmente es el software de alta calidad.

Las métricas de mantenibilidad no pueden medir el coste de realizar un cambio particular al sistema software, sino que miden aspectos de la complejidad y la calidad de los programas, ya que existe una alta correlación entre la complejidad y la mantenibilidad (a mayor complejidad menor mantenibilidad) y entre la calidad y la mantenibilidad (a mayor calidad mayor mantenibilidad, y viceversa). Existen maneras de medir la mantenibilidad para todos los elementos software que están o estarán sometidos a mantenimiento: código, documentos de usuario, documentos de análisis o diseño, etc. Las métricas del software se pueden clasificar en tres categorías[2]:

- Métricas de producto.
- Métricas del proceso.
- Métricas de proyecto.

Algunas métricas pueden pertenecer a múltiples categorías.

Centrándonos en las Métricas de Mantenibilidad Orientadas al Producto, estas describen las características del producto que de alguna forma determinan la mantenibilidad, por ejemplo, el tamaño, complejidad o características del diseño. Una de las métricas orientadas al producto son las Métricas en Orientación a Objetos. Estas métricas se centran en métricas que se pueden aplicar a las características de encapsulamiento, ocultamiento de información, herencia y técnicas de abstracción de objetos.

Chidamber & Kemerer[3] proponen una familia de medidas para desarrollos orientados a objetos:

- Métodos ponderados por clase (MPC): Tamaño y complejidad
- Profundidad árbol de herencia (PAH): Tamaño
- Número de descendientes (NDD): Tamaño, acoplamiento y cohesión
- Acoplamiento entre clases (ACO): Acoplamiento
- Respuesta para una clase (RPC): Comunicación y complejidad
- Carencia de cohesión en los métodos (CCM): Cohesión interna

Estas métricas, en líneas generales, permiten averiguar cuán bien están definidas las clases y el sistema, lo cual tiene un impacto directo en la mantenibilidad del mismo, tanto por la comprensión de lo desarrollado como por la dificultad de modificarlo con éxito.

---

[1] https://cnx.org/exports/a0484c7c-5b86-4ef2-9f7e-fb6bb11dfe59@9.1.pdf/m%C3%A9tricas-del-mantenimiento-de-software-9.1.pdf

[2] https://cnx.org/contents/iMh-5ld0@2/M%C3%A9tricas-de-Mantenibilidad-del-Software

[3] Chidamber, S.R., D.P. y C.F.Kemerer, "Management Use of Metrics for Object-Oriented Software: An Exploratory Analysis", IEEE Trans. Software Engineering.

Ejercicio propuesto

El ejercicio consiste en analizar la mantenibilidad de un software y modificarlo para mejorar este factor de calidad.

En PoliformaT->Recursos->Seminarios->Seminario 1, tenéis un fichero comprimido `src.zip` con varias clases Java (`Book`, `Customer`, `Movie` y `Rental`, y `Main y Consola`). Crear un proyecto Java con dos paquetes: un paquete `igu` con la clase `Main y Consola`, y un paquete `logic` con el resto de clases.

A partir de estas clases se pide:

1. Dibujar el diagrama de clases de diseño (utilizar cualquier herramienta software para editar diagramas de clases o hacerlo a mano).
2. Analizar la **cohesión** y **acoplamiento** de las clases, y los paquetes (con una escala de valores sencilla, por ejemplo: MALO, REGULAR, BUENO).
3. Reflexionar acerca de la facilidad de realizar los siguientes cambios:
    a. añadir un nuevo código de precio: TopSeller
    b. añadir un nuevo elemento para alquilar: por ejemplo, dispositivo electrónico
4. ¿Se te ocurre cómo puedes modificar el diseño y la implementación para mejorar el mantenimiento?