Análisis y desarrollo de la implementación del desafío 2

Informática II

Jose Andrés Henao Alzate
Departamento de Ingeniería Electrónica
Universidad de Antioquia
Medellín, Colombia
andres.henao9@udea.edu.co

***Resumen*—Este proyecto consiste en el desarrollo de un sistema de gestión de combustible para la red nacional de estaciones de servicio TerMax en Colombia. Utilizando programación orientada a objetos (POO) en C++, el sistema gestiona eficientemente las ventas de combustible y realiza el seguimiento de transacciones en múltiples estaciones. El sistema simula operaciones como el procesamiento de ventas, la gestión de inventarios y la detección de fugas. Al modelar estos procesos de manera precisa, el sistema proporciona una simulación de las operaciones de una estación de servicio, optimizando la gestión y distribución de recursos.**

***Index Terms*—Lenguaje C++, Eficiencia, Análisis, Clases, Abstracción, Encapsulamiento, Atributos, Métodos.**

I. Introducción

Las estaciones de servicio son infraestructuras fundamentales para el suministro de combustible. En Colombia, la gestión eficiente de estas estaciones es clave para garantizar la continuidad del transporte y el abastecimiento de bienes. El proyecto TerMax tiene como objetivo mejorar la administración de estaciones de servicio mediante un sistema automatizado, que optimiza operaciones como la venta de combustible, gestión de inventarios y detección de fugas. Este informe detalla el diseño e implementación del sistema utilizando programación orientada a objetos en C++. Además, se describen los problemas encontrados y los resultados obtenidos durante la implementación.

II. Análisis del problema y procedimiento de solución

El primer paso en la solución fue comprender el problema y llevarlo a un enfoque orientado a objetos. Se identificaron tres clases clave: *Estación de servicio*, *Surtidor*, y *Tanque*, representadas en la figura 1. La clase principal, *Estación de servicio*, contiene los atributos principales del problema: **nombre, código, gerente, región** y **ubicación geográfica**. Estos atributos son extraídos del archivo principal, por lo que no se profundiza en ellos aquí.

Los métodos asociados a esta clase son esenciales para simular el comportamiento del sistema. Entre ellos destacan:

- **void fugas():** Verifica posibles fugas en el tanque principal. Es una función amiga de la clase *Tanque*, ya que accede a sus atributos privados.
- **void vender():** Simula una venta de combustible en un surtidor específico.

Para la clase *Surtidor*, sus atributos describen características estándar del surtidor, mientras que su método principal es:

- **void consultar_venta():** Busca en el archivo de ventas las transacciones realizadas por ese surtidor y calcula su producción.

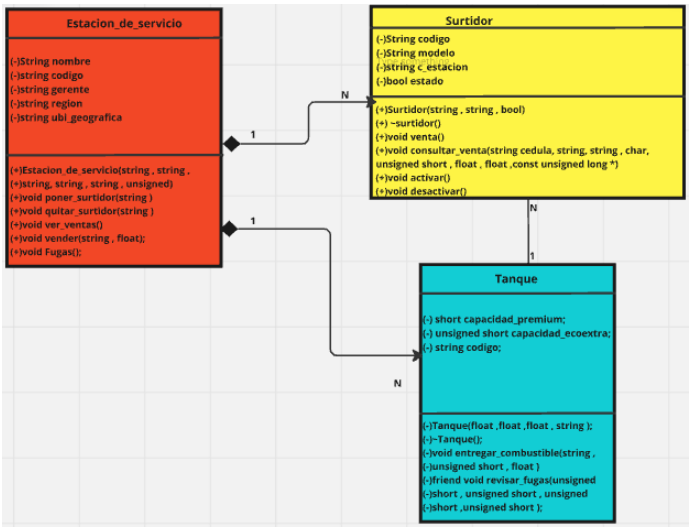

Figura 1. Diagrama UML simplificado.

III. Problemas encontrados y evolución de la implementación

Aunque la implementación del sistema no presentaba una alta complejidad técnica, el tiempo disponible fue un factor determinante. Debido a problemas externos, la carga de trabajo recayó en una sola persona, lo que afectó el avance del proyecto. Aunque se logró implementar todas las clases, no

fue posible integrar completamente el menú del sistema ni instanciar todas las clases de manera óptima.

No obstante, las clases fueron probadas de forma individual, y las pruebas arrojaron resultados satisfactorios. Las operaciones de escritura en los archivos también funcionaron como se esperaba, lo cual es un indicador de que los métodos y atributos fueron correctamente implementados y que con un tiempo adicional, se da una garantía de una entrega mas óptima y de mayor calidad. Aun con lo anterior, y en contraste con el argumento final la entrega en general deja mucho que desear

IV. Discusión de resultados

A nivel técnico, las clases y sus funcionalidades arrojaron resultados positivos en las pruebas individuales. Sin embargo, debido a limitaciones de tiempo, no se completó la implementación del menú y la integración total del sistema, lo que afectó el resultado general del proyecto. Aun así, los módulos probados confirmaron el funcionamiento correcto de las operaciones críticas del sistema.

V. Conclusiones

- **Implementación parcial:** Aunque las clases fueron correctamente implementadas y probadas de forma individual, la falta de tiempo impidió la integración completa de todas las funcionalidades, como el menú principal del sistema.
- **Resultados técnicos:** Las pruebas realizadas sobre las clases y las operaciones de escritura en archivos fueron satisfactorias y cumplieron con lo esperado.
- **Impacto del tiempo:** La principal limitación fue la falta de tiempo, debido a la concentración de la carga de trabajo en una sola persona. Este proyecto resalta la importancia de una gestión de tiempo eficiente y una distribución equitativa de tareas para completar proyectos de mayor envergadura.