Cohesión en el Diseño de Software

Edilson Avalos Condori

June 13, 2024

1 Definición

La cohesión mide el grado en que los elementos de un módulo están relacionados funcionalmente. Alta cohesión facilita el mantenimiento y la reutilización del código.

2 Tipos de Cohesión

- **Funcional**: Todos los elementos contribuyen a una sola función bien definida.
- **Secuencial**: La salida de un elemento es la entrada de otro.
- **Comunicacional**: Los elementos operan sobre los mismos datos.
- **Procedural**: Los elementos son parte de un mismo procedimiento.
- **Temporal**: Los elementos se ejecutan en el mismo período de tiempo.
- **Lógica**: Los elementos realizan tareas lógicamente similares.
- **Coincidental**: Los elementos están agrupados arbitrariamente.

3 Importancia de la Alta Cohesión

- **Mantenimiento**: Facilita la localización y corrección de errores.
- **Reutilización**: Los módulos cohesivos son más fáciles de reutilizar en otros contextos.
- **Comprensibilidad**: Código más fácil de entender y modificar.

4 Ejemplo de Código

A continuación, se muestra un ejemplo de código en Python para ilustrar cómo una clase con alta cohesión tendría todos sus métodos y atributos relacionados con una única tarea o conjunto de tareas estrechamente relacionadas:

```python
class Calculadora:
    def __init__(self):
        self.resultado = 0

    def sumar(self, valor):
        self.resultado += valor
        return self.resultado

    def restar(self, valor):
        self.resultado -= valor
        return self.resultado

    def multiplicar(self, valor):
        self.resultado *= valor
        return self.resultado

    def dividir(self, valor):
        if valor != 0:
            self.resultado /= valor
        else:
            raise ValueError("No se puede dividir por cero")
        return self.resultado

# Ejemplo de uso de la clase Calculadora
calculadora = Calculadora()
print(calculadora.sumar(10))        # Output: 10
print(calculadora.restar(2))        # Output: 8
print(calculadora.multiplicar(3))   # Output: 24
print(calculadora.dividir(4))       # Output: 6
```

5 Referencias

- Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.
- Repositorio de GitHub con ejemplos relacionados: `https://github.com/EdilsonAvalosCondori/Ingenier-a-de-software`