1 Complejidad de Métodos

La complejidad de métodos en software se refiere a la medida de la dificultad para entender, modificar y mantener un método o una función dentro de un programa. Varias métricas se utilizan para cuantificar esta complejidad, entre ellas la Complejidad Ciclomática, las métricas de Halstead y la longitud del código.

1.1 Definición y Tipos

La complejidad de métodos en software se refiere a la medida de la dificultad para entender, modificar y mantener un método o una función dentro de un programa. Esta complejidad se puede medir utilizando diversas métricas.

- **Complejidad Ciclomática:** Desarrollada por Thomas McCabe en 1976, esta métrica mide el número de caminos independientes a través del código de un programa. Se calcula usando la fórmula $M = E - N + 2P$, donde E es el número de aristas en el grafo de flujo de control, N es el número de nodos y P es el número de componentes conectados [1].
- **Métricas de Halstead:** Introducidas por Maurice Halstead, estas métricas analizan los operadores y operandos en el código para medir la complejidad. Algunas de las métricas de Halstead incluyen el Volumen (V), el Esfuerzo (E) y la Longitud (N). El Volumen, por ejemplo, se calcula como $V = N \cdot \log_2(n)$, donde N es la longitud del programa y n es el tamaño del vocabulario del programa [2].
- **Líneas de Código (LOC):** Una métrica simple que cuenta el número total de líneas de código en un método. Aunque no mide directamente la complejidad lógica, un alto número de LOC puede indicar que el código es difícil de manejar y entender [3].

1.2 Aplicaciones y Limitaciones

1.2.1 Aplicaciones

- **Evaluación de Calidad:** Las métricas de complejidad ayudan a los desarrolladores a identificar partes del código que pueden ser propensas a errores y difíciles de mantener [1].
- **Mejora del Código:** Ayudan a decidir cuándo modularizar el código para mejorar su mantenibilidad y facilidad de prueba [4].
- **Planificación de Pruebas:** La complejidad ciclómatica, en particular, se usa para determinar el número mínimo de casos de prueba necesarios para cubrir todos los caminos posibles en el código [3].

1.2.2 Limitaciones

- **Interpretación Variable:** La interpretación de las métricas puede variar dependiendo de las herramientas utilizadas y del contexto del proyecto. Diferentes herramientas de análisis estático pueden proporcionar resultados diferentes para las mismas métricas [2].
- **Limitación en Métricas Individuales:** Ninguna métrica individual puede capturar completamente la complejidad de un sistema. Es importante utilizar múltiples métricas en conjunto para obtener una visión más completa .
- **No Correlación Directa con Defectos:** Algunas investigaciones han mostrado que métricas como la complejidad ciclómatica no siempre están correlacionadas con el número de defectos en el código [3].

A continuación, se presenta un ejemplo de cómo calcular la Complejidad Ciclomática utilizando Python y la biblioteca `radon`:

Listing 1: Ejemplo de código en Python para calcular la Complejidad Ciclomática

```
import subprocess
import os
```

```
def calcular_complejidad_ciclomatica(nombre_archivo):
    try:
        resultado = subprocess.check_output(['radon', 'cc', '-s', nombre_archivo])
        resultado = resultado.decode('utf-8')
        print(f"Complejidad ciclomatica para {nombre_archivo}:\n{resultado}")
    except subprocess.CalledProcessError as e:
        print(f"Error al calcular la complejidad ciclomatica: {e}")

def main():
    nombre_archivo = input("Ingrese el nombre del archivo de codigo: ")
    if not os.path.isfile(nombre_archivo):
        print("El archivo no existe.")
        return

    calcular_complejidad_ciclomatica(nombre_archivo)

if __name__ == "__main__":
    main()
```

La complejidad de métodos es una métrica esencial para evaluar la calidad del software y mejorar los procesos de desarrollo. Aunque cada métrica tiene sus limitaciones, el uso combinado de varias métricas proporciona una visión más completa del estado del software y las áreas potenciales de mejora.

References

[1] C. Ebert, J. Cain, G. Antoniol, S. Counsell, and P. Laplante, "Cyclomatic complexity," *IEEE Software*, vol. 33, no. 6, pp. 27–29, 2016. DOI: 10.1109/MS.2016.147.

[2] D. R. Wijendra and K. Hewagamage, "Analysis of cognitive complexity with cyclomatic complexity metric of software," *International Journal of Computer Applications*, vol. 174, pp. 14–19, 2021.

[3] A. Odeh, M. Odeh, N. Odeh, and H. Odeh, "Machine learning model for measuring cyclomatic complexity of source code," in *2023 International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS)*, 2023, pp. 149–153. DOI: 10.1109/ICCNS58795.2023.10193630.

[4] S. G. MacDonell, "Comparative review of functional complexity assessment methods for effort estimation," English, *Software Engineering Journal*, vol. 9, 107–116(9), 3 May 1994, ISSN: 0268-6961. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/sej.1994.0014.