Métricas de Mantenibilidad de Software

Lindell Dennis Vilca Mamani

1 Métricas de Mantenibilidad de Software

La mantenibilidad de software se refiere a la facilidad con la que se puede modificar un sistema de software después de su entrega inicial. Esta capacidad de modificación incluye corrección de defectos, mejora de rendimiento, adaptación a un entorno modificado y adición de nuevas funcionalidades. Aquí se describen las métricas más relevantes para evaluar la mantenibilidad de un software [2].

1.1 Métricas de Proceso

Las métricas de proceso se refieren a atributos relacionados con las actividades de mantenimiento del software. Estas incluyen:

- **Número de peticiones para mantenimiento correctivo**: Un incremento en los reportes de errores puede indicar una disminución en la mantenibilidad del software, sugiriendo que se introdujeron más errores de los que se corrigieron.
- **Tiempo promedio para el análisis del impacto**: Este tiempo refleja cuántos componentes del programa se ven afectados por una solicitud de cambio. Un incremento en este tiempo sugiere una mayor complejidad en el mantenimiento.
- **Tiempo promedio para implementar una petición de cambio**: Diferente del tiempo para el análisis del impacto, mide el tiempo necesario para modificar el sistema y su documentación. Un aumento en este tiempo puede indicar una menor mantenibilidad.
- **Número de peticiones de cambio pendientes**: Un aumento en el número de solicitudes de cambio pendientes puede indicar dificultades en el mantenimiento del software.

1.2 Métricas de Producto

Las métricas de producto evalúan atributos específicos del software mismo, incluyendo:

- **Métricas dinámicas**: Se recopilan mediante mediciones realizadas en un programa en ejecución, como el número de reportes de errores o el tiempo necesario para completar un cálculo.
- **Métricas estáticas**: Se obtienen mediante mediciones de representaciones del sistema, como el código fuente o la documentación. Ejemplos incluyen el tamaño del código y la complejidad ciclomática.

1.3 Recomendaciones para Mejorar la Mantenibilidad

Para mejorar la mantenibilidad del software, se recomienda:

- **Especificación precisa y uso de desarrollo orientado a objetos**: Ayuda a reducir los costos de mantenimiento.
- **Administración de la configuración**: Facilita el seguimiento de los cambios y las versiones del software.
- **Documentación de calidad**: Mantener la documentación actualizada es crucial para la analizabilidad y comprensibilidad del software [2].

2 Aplicaciones y Limitaciones de las Métricas de Mantenibilidad de Software

2.1 Aplicaciones

1. **Evaluación de la Calidad del Código**:
   - **Testabilidad**: Mide cuán efectivamente se pueden realizar pruebas en el software. Incluye la cobertura de pruebas y la calidad de los casos de prueba [1].
   - **Comprensibilidad**: Evalúa la legibilidad del código, considerando si sigue convenciones de nombrado, si está bien comentado y si la intención de los métodos es clara [1].
2. **Gestión de Proyectos**:
   - **Estimación y Planificación**: Facilitan la planificación y asignación de recursos en proyectos de mantenimiento al proporcionar datos sobre la complejidad y el esfuerzo requerido .
   - **Riesgo y Priorización**: Ayudan a priorizar cambios y evaluaciones de riesgo basadas en la estructura del código y su susceptibilidad a errores .
3. **Optimización del Rendimiento**:
   - **Modificabilidad**: Analiza la simplicidad estructural y de diseño, evaluando la facilidad para realizar cambios en el software sin introducir errores [1].

2.2 Limitaciones

1. **Dificultad en la Medición Directa**:
   - **Subjetividad**: Muchos aspectos de la mantenibilidad, como la comprensibilidad y la simplicidad del diseño, son subjetivos y difíciles de medir directamente [1].
   - **Automatización Completa**: La evaluación automatizada puede ser incompleta y a menudo requiere intervención manual para verificar resultados .
2. **Dependencia del Contexto**:
   - **Entorno y Herramientas**: La relevancia de ciertas métricas puede variar dependiendo de la tecnología y el entorno de desarrollo utilizado .
   - **Variabilidad en la Interpretación**: Diferentes equipos pueden interpretar y priorizar las métricas de manera distinta, afectando su aplicación y utilidad .
3. **Recursos y Costos**:
   - **Implementación y Mantenimiento**: Requiere inversión significativa en herramientas y capacitación para recolectar y analizar métricas de manera efectiva .
   - **Análisis e Interpretación**: Necesita personal especializado para interpretar las métricas correctamente y tomar decisiones informadas, lo cual puede no estar disponible en todas las organizaciones [1].

3 Ejemplo en Python

Listing 1: Script para calcular métricas de mantenibilidad de software

```
import radon
from radon.complexity import cc_visit
from radon.metrics import mi_visit
from radon.raw import analyze

# Leer el contenido del archivo de codigo fuente
def read_file(file_path):
    with open(file_path, 'r') as file:
        return file.read()

# Calcular la complejidad ciclomatica
def calculate_cyclomatic_complexity(code):
    complexities = cc_visit(code)
```

```
    total_complexity = sum(c.complexity for c in complexities)
    return total_complexity

# Calcular las metricas crudas (SLOC, comentarios, etc.)
def calculate_raw_metrics(code):
    metrics = analyze(code)
    return metrics

# Calcular el indice de mantenibilidad
def calculate_maintainability_index(code, multi=False):
    mi = mi_visit(code, multi)
    return mi

if __name__ == "__main__":
    file_path = 'ruta/al/archivo_de_codigo.py'
    code = read_file(file_path)

    # Calcular metricas
    cyclomatic_complexity = calculate_cyclomatic_complexity(code)
    raw_metrics = calculate_raw_metrics(code)
    maintainability_index = calculate_maintainability_index(code)

    # Imprimir resultados
    print(f"Complejidad ciclomatica total: {cyclomatic_complexity}")
    print(f"Metricas crudas: {raw_metrics}")
    print(f"Indice de mantenibilidad: {maintainability_index}")
```

En resumen, las métricas de mantenibilidad son herramientas cruciales para evaluar y mejorar la calidad del software, pero es esencial considerar sus limitaciones y el contexto específico del proyecto para maximizar su efectividad.

References

[1] Quandary Peak Research, "Measuring Software Maintainability," `https://quandarypeak.com/2020/11/measuring-software-maintainability/`, accessed June 13, 2024.

[2] Gómez, O., Henríquez, J., Garrido, J., & Vidal, P. (2020). Métricas para la mantenibilidad del software. *Ingeniare. Revista chilena de ingeniería*, 28(4), 654-663.