Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

IC6831 Aseguramiento De La Calidad Del Software

Grupo 40

Actividad 05 - Informe Grupal de Métricas

Profesora:

Ericka Solano Fernández

Equipo 1:

Valery Mishel Carvajal Oreamuno – 2022314299

Luis Felipe Calderon Perez – 2021048663

Sergio Chavarría Avilés – 2021077879

Kevin Yadir Calvo Rodriguez – 2023367224

Andrew Denilson Lopez Herrera – 2021062132

Enero, 2025

Índice

Métricas asociadas

Modularidad: Se evaluó utilizando la métrica Número de Módulos (NM), que mide la división del código en componentes independientes. Se evaluó también utilizando la métrica de adecuación de la complejidad ciclomática, que representa cuántos módulos de software tienen una complejidad ciclomática aceptable.

Reusabilidad: Se midió mediante la métrica Reutilización de Funciones (RF), que calcula la frecuencia con la que las funciones son reutilizadas dentro del proyecto.

Analizabilidad: Fue evaluado por la métrica de complejidad ciclomática (CC), la cual evalúa la complejidad de la lógica del programa.

Capacidad de ser probado: Las métricas establecidas fueron promedio del tamaño de instrucciones (PI), acoplamiento entre objetos (AO) y complejidad ciclomática (CC).

Herramientas

Herramientas exploradas para cada uno de los lenguajes de los proyectos analizados:

Understand - SciTools: herramienta utilizada para analizar modularidad.

SonarQube: herramienta utilizada para medir reutilización de funciones y duplicación de código.

SonarCloud: herramienta web usada para medir la mantenibilidad de un proyecto, midiendo la reutilización de funciones y la seguridad del código.

Codacy: Es una herramienta automatizada de análisis y calidad de código que ayuda a los desarrolladores a entregar un mejor software, más rápido . Con Codacy, obtienes análisis estático, complejidad ciclomática, duplicación y cambios en la cobertura de pruebas unitarias de código en cada confirmación y solicitud de incorporación de cambios. (Python, pero se pueden usar otros lenguajes).

(Python) pylint: Pylint analiza el código, busca errores, aplica un estándar de codificación, busca errores en el código y puede hacer sugerencias sobre cómo refactorizar.

(Python) radon: Radon es una herramienta de Python que calcula varias métricas a partir del código fuente, como, complejidad ciclomática, métricas sin procesar (entre ellas, SLOC, líneas de comentarios, líneas en blanco, etc.), métricas de Halstead (todas ellas) e índices de mantenibilidad (el que se utiliza en Visual Studio).

Otras herramientas

Algunas otras herramientas asociadas que no fueron exploradas.

- CodeMR
- Ndepend
- Jarchitect
- Junit
- JaCoCo
- LinearB
- Swarmia
- Selenium
- BrowserStack Test Management
- PractiTest
- Apache Jmeter
- CodeScene
- Crucero de dependencias (JS)

Criterios de aceptación/rechazo encontrados en la literatura y experiencias de la industria.

- **Modularidad:**

  Métrica 1:

  ○ 1-5 módulos: Óptimo

  ○ 6-10 módulos: Aceptable

  ○ 11+ módulos: Requiere optimización

  Métrica 2:

  X = 1– A/B

  Donde:

  A = Número de módulos de software que tienen un puntaje de complejidad ciclomática que excede el umbral especificado

  B = Número de módulos de software implementados

  Sí X >= 0.8, entonces es aceptable

  Métrica 3:

  Cantidad de líneas de código por módulo (LOC)

  ○ LOC > 200 : Rechazo

  ○ 100 < LOC < 200 : Aceptable

- **Reusabilidad:**

  ○ 0-30%: Baja reutilización

  ○ 31-60%: Aceptable

  ○ 61%+: Alta reutilización

- **Analizabilidad:**

  ○ La complejidad ciclomática debe ser inferior a 10.

  ○ Debe tener un valor igual o inferior a 0,67 de acoplamiento entre los objetos.

- **Capacidad de ser probado:**

  Métrica 1:

  Complejidad ciclomática, calculada sumando dependiendo de cada palabra clave (o su equivalente) dentro del código:

  De 1 - 5, riesgo bajo - bloque simple - aceptable

  De 6 - 10, riesgo bajo - bloque bien estructurado y estable - aceptable

  De 11 - 20, riesgo moderado - bloque ligeramente complejo - mejorable

  De 21 - 30, riego más que moderado - bloque más complejo - mejorable

  De 31 - 40, riego alto - bloque complejo, alarmante - inaceptable

  41+, riesgo muy alto - bloque inestable y propenso a errores - inaceptable

- **Modificabilidad:**

  Métrica 1:

  Porcentaje de líneas duplicadas

  ○ Porcentaje de líneas duplicadas < 5% : Bueno

  ○ 5% <= Porcentaje de líneas duplicadas < 10% : Aceptable

  ○ 10% <= Porcentaje de líneas duplicadas: Rechazo

  Métrica 2:

  Relación de deuda técnica

  ○ Porcentaje de líneas duplicadas < 10% : Bueno

  ○ 10% <= Porcentaje de líneas duplicadas < 20% : Aceptable

  ○ 20% <= Porcentaje de líneas duplicadas : Rechazo

Tabla comparativa

Aporte tablas comparativas donde a través de algún criterio se puedan evaluar los aspectos que dan una valoración positiva a algunas de las herramientas exploradas.
Espacio

Herramienta	**Understand**	**SonarQube**	**SonarCloud**	**Codacy**	**pylint**	**radon**
Instalación	Página web	Página web	Página web	Página web	Comandos	Comandos
Método Pago	Licencia comercial paga o suscripción anual. Hay prueba gratuita.	Versión Community gratuita y planes comerciales para versiones Developer, Enterprise y Data Center.	Plan básico gratuito y otros dos más completos de paga.	Plan gratuito y otros métodos de pago.	Gratuito	Gratuito y de código abierto.
Tecnología	C, C++, Java y Python.	Compatible con muchos lenguajes de programación, scripting, desarrollo móvil, big data, infraestructura y demás.	Java	Múltiple	Python	Python
Forma de cargar el proyecto	Se importa el código fuente desde repositorios y se configura un proyecto seleccionando el lenguaje y las opciones de análisis.	Se configura un proyecto en el servidor SonarQube y se analiza el código desde el CI/CD o localmente con el scanner de SonarQube.	Se conecta a una github. Se debe configurar un archivo build.yml, agregar properties al pom.xml y agregar un secret de github. Se hará un estudio al proyecto cada vez que haya	Se conecta a una cuenta de GitHub y con ella se carga un repositorio que va a ser puesto a prueba.	Acceder al archivo por medio de una terminal y ejecutar el comando “pylint nombreArchivo.py”	Se ejecuta “radon cc <ruta_del_proyecto>” desde terminal.

			un commit.			
Funciones	Análisis de métricas (complejidad, líneas de código, duplicaciones). Visualización de dependencias. Generación de informes detallados. Identificación de problemas en el código para mejorar mantenibilidad.	Detecta bugs, vulnerabilidades y problemas de calidad. Mide métricas como mantenibilidad, cobertura de pruebas y complejidad. Soporte para integración con herramientas de desarrollo.	Mide mantenibilidad y seguridad del proyecto. Crea lista de correcciones pendientes. Compara cada revisión con las revisiones pasadas para ver el progreso.		Mide mantenibilidad	Identifica funciones/métodos complejos. Calcula índices de calidad del código. Linters para análisis estático.

Referencias bibliográficas consultadas

Admin. (2024, 26 marzo). *Cyclomatic complexity: understanding an essential metric*. Metridev.https://www.metridev.com/metrics/cyclomatic-complexity-understanding-an-essential-metric/#:~:text=It%20depends%20on%20the%20specific,below%2010%20to%20be%20acceptable.

Codacy - Code Quality and Security for Developers. (s. f.). https://www.codacy.com/

International Organization for Standardization. (2011). ISO/IEC 25010: Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – System and software quality models. ISO. Recuperado de https://iso25000.com/index.php/normas-iso-25000/iso-25010

SciTools. (n.d.). Understand – Static analysis and metrics for code. Recuperado de https://www.scitools.com

SonarCloud. (n.d.). *SonarCloud: Continuous Code Quality & Security*. Recuperadode https://sonarcloud.io

SonarSource. (n.d.). SonarQube Documentation. Recuperado de https://docs.sonarqube.org

Rojas, R., Pérez, M., & Martínez, J. (2020). Mantenibilidad del Software: Consideraciones para su especificación y validación. Revista Ingeniare. Revista chilena de ingeniería, 28(4), 654-667.

Sde-Coursepack. (s. f.). *SDE*. Analyzability. https://sde-coursepack.github.io/modules/refactoring/Analyzability/

Heitlager, Ilja & Kuipers, Tobias & Visser, Joost. (2007). A Practical Model for Measuring Maintainability. 30 - 39. 10.1109/QUATIC.2007.8. (https://www.researchgate.net/publication/4276839_A_Practical_Model_for_Measuring_Maintainability)

Lacchia, M. (s.f.). Introduction to Code Metrics: Cyclomatic Complexity. Radon 4.1.0 documentation. Recuperado de: https://radon.readthedocs.io/en/latest/intro.html#cyclomatic-complexity

Organización Internacional de Normalización. (2010). Calidad se Software y Datos (ISO 25010). https://iso25000.com/index.php/normas-iso-25000/iso-25010

Organización Internacional de Normalización. (2016). ISO 25023 : SQuARE (System and Software Quality Requirements and Evaluation) Measurement of system and software product quality. ISO/IEC 25023:2016(E)

Pressman, Roger S. Ph.D. (1982). Software Engineering - A Practitioner's Approach - Fourth Edition. 0-07-052182-4

pylint. (2024, 24 diciembre). PyPI. https://pypi.org/project/pylint/

radon. (2023, 26 marzo). PyPI. https://pypi.org/project/radon/