**Exploring the Cyclomatic Complexity’s Relevance Today – Unit 5**

**Question**

The Cyclomatic Complexity is commonly considered in modules on testing the validity of code design today. However, in your opinion, should it be? Does it remain relevant today? Specific to the focus of this module, is it relevant in our quest to develop secure software?

**Answer**

The Cyclomatic or McCabe’s Complexity is a common metric used in software development to quantify the complexity of a software by measuring the linearly independent paths in it (Ebert *et al.*, 2016). Optimal values of cyclomatic complexity lie between 5 and 10 (Kumar *et al*., 2021), and this metric can be leveraged to quantify such a complexity at various scales, i.e., from modules to functions and classes in them, along with their methods (Butler & McCabe, 2021). Therefore, this measure enables to estimate the modularity and maintainability of the software, thus enabling to highlight software-related components that need to be simplified or further decoupled (Butler & McCabe, 2021). As a result, software’s quality can be improved, along with its testability.

The cyclomatic complexity measure can lead to improve the software’s maintainability, quality, and testability, thus also enabling to identify potential security vulnerabilities, assessing their impact, and mitigating it by enhancing the software design and streamlining business logic via appropriate refactoring (Caldeira *et al*., 2022) to boost software security (Kumar *et al*., 2021). Furthermore, the cyclomatic complexity is faster to calculate than the Halstead’s measures and it relatively straightforward to leverage, thus enabling to optimise the software and enhance its security from design to development and support, i.e., throughout the software development lifecycle (Liu *et al*., 2018).

**References**

Butler, C. W., & McCabe, T. J. (2021) Cyclomatic Complexity-Based Encapsulation, Data Hiding, and Separation of Concerns. *Journal of Software Engineering and Applications*, 14(1): 44-66.

Caldeira, J., e Abreu, F. B., Cardoso, J., & dos Reis, J. P. (2022) Unveiling process insights from refactoring practices. *Computer Standards & Interfaces*, 81: 103587.

Ebert, C., Cain, J., Antoniol, G., Counsell, S., & Laplante, P. (2016) Cyclomatic complexity. *IEEE software*, 33(6): 27-29.

Kumar, S. K. S., Kulyadi, S. P., Mohandas, P., Raman, M. S., & Vasan, V. S. (2021) Computation of Cyclomatic Complexity and Detection of Malware Executable Files. In *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)* (pp. 1-5). IEEE.

Liu, H., Gong, X., Liao, L., & Li, B. (2018) Evaluate how cyclomatic complexity changes in the context of software evolution. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* (Vol. 2, pp. 756-761). IEEE.