**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? Justify all opinions which support an argument and share your responses with your team.**

## The Cyclomatic Complexity

According to Ebert et al, (2016) Cyclomatic Complexity is a software metric used to indicate the complexity of a program and the number of linear independent paths through a program's source code.

Abran etal (2004) argued that higher Cyclomatic Complexity can result in more complex code which can then have security issues and require constant debugging. It can be counter argued that Cyclomatic Complexity is relevant for code maintainability, as high complexity suggests intricate code structures that hinder understanding and modification. By identifying complex areas, developers can simplify code, improve readability, and reduce the likelihood of introducing bugs during maintenance.

It also aids testability by providing insights into the number of test cases required for comprehensive code coverage (Phogat, et al 2011). Higher complexity indicates more potential execution paths, necessitating increased testing efforts. Considering Cyclomatic Complexity helps prioritize testing and ensures critical paths are adequately covered.

Mccabe (1996), advocated it can help with risk identification, for instance with the Millennium bug. High complexity indicates error-prone areas that may contain defects. Developers and reviewers can pay closer attention to complex sections during code reviews and testing, reducing potential vulnerabilities.

Madi et al (2013) contend that Cyclomatic Complexity has limited focus since it solely measures control flow complexity, disregarding factors such as data flow, input validation, and domain-specific security vulnerabilities. Relying solely on Cyclomatic Complexity may overlook critical security issues.

The interpretation of Cyclomatic Complexity thresholds can be subjective and lead to context dependency. Lopez and Habra (2005) concluded that acceptable complexity varies between projects and domains and applying robust standards without considering specific requirements may lead to suboptimal code design decisions.

Tiwari and Kumar (2014) maintained that in developing secure software, Cyclomatic Complexity can be used as a base method of testing. This can help with maintainability as well as security. It should be also be used in conjunction with secure coding practices, threat modelling, and vulnerability analysis.

To conclude, Cyclomatic Complexity is relevant for code maintainability, testability, and risk identification. However, it can be debated whether it is useful in developing secure software. Cyclomatic Complexity should not be used in isolation for measuring security. A more robust way would be to utilise secure coding practices, threat modeling, and domain-specific security considerations.  As such, Cyclomatic Complexity is a valuable tool, however, using a range of other security measures to develop secure software would be good practice.

Abran, A., Lopez, M. and Habra, N., 2004. An analysis of the McCabe Cyclomatic complexity number. In Proceedings of the 14th International Workshop on Software Measurement (IWSM) IWSM-Metrikon (pp. 391-405).

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

Lopez, M. and Habra, N., 2005. Relevance of the cyclomatic complexity threshold for the java programming language. SMEF 2005, p.195.

Madi, A., Zein, O.K. and Kadry, S., 2013. On the improvement of cyclomatic complexity metric. International Journal of Software Engineering and Its Applications, 7(2), pp.67-82.

Mccabe, T., 1996. Cyclomatic complexity and the year 2000. IEEE Software, 13(3), pp.115-117.

Phogat, M., Kumar, D. and Murthal, D.C.R.U.S.T., 2011. Testability of software system. IJCEM International Journal of Computational Engineering & Management, 14, p.10.

Tiwari, U. and Kumar, S., 2014. Cyclomatic complexity metric for component based software. ACM SIGSOFT Software Engineering Notes, 39(1), pp.1-6.