**The Role of Cyclomatic Complexity and Faceted Data in Secure Software Development**

**Hamdan Almheiri**

**Table of Contents**

[1 Portfolio 1: The Relevance of Cyclomatic Complexity in Modern Secure Software Development 3](#_Toc177557089)

[1.1 The Evolution of Software Development Methodologies 3](#_Toc177557090)

[1.2 Is Cyclomatic Complexity Still Relevant? 4](#_Toc177557091)

[1.3 Modern Tools and Techniques 4](#_Toc177557092)

[1.4 The Role of Cyclomatic Complexity in Secure Software Development 5](#_Toc177557093)

[1.5 Conclusion 5](#_Toc177557094)

[2 Portfolio 2: Ontology and Faceted Data as Tools for Secure System Design 6](#_Toc177557095)

[2.1 Faceted Data as a Solution for Data Leakage 6](#_Toc177557096)

[2.2 Advantages of Faceted Data 7](#_Toc177557097)

[2.3 Challenges and Drawbacks 8](#_Toc177557098)

[2.4 Designing a Python System for Faceted Data Security 8](#_Toc177557099)

[2.5 Conclusion 8](#_Toc177557100)

[References 10](#_Toc177557101)

**Table of Figures**

[Figure 1: Cyclomatic Complexity 3](#_Toc177557084)

[Figure 2: Waterfall SDLC 4](#_Toc177557085)

[Figure 3: Role of Cyclomatic Complexity in Secure Software Development 5](#_Toc177557086)

[Figure 4: Components of Ontology 6](#_Toc177557087)

[Figure 5: Advantages of Faceted Data 7](#_Toc177557088)

# Portfolio 1: The Relevance of Cyclomatic Complexity in Modern Secure Software Development

Cyclic Complexity, developed by Thomas McCabe in 1976, is a measure of program complexity that uses its control flow, as shown in Figure 1 (McCabe, 1976). It calculates the number of independent paths through a program. This metric can point out how maintainable the code is and how failures could occur. CC has been traditionally a cornerstone in software development, but its applicability in modern security-conscious environments is highly debated.

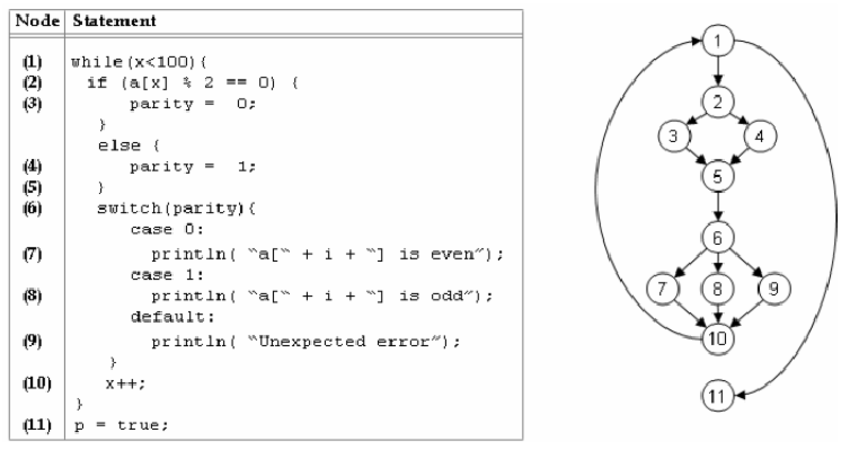


Figure 1: Cyclomatic Complexity

## The Evolution of Software Development Methodologies

According to Chandran and Das (2022), in older decades, the waterfall was the dominating model in software development; see Figure 2. This waterfall approach consisted of rigid, sequential stages where CC played an essential role in determining code complexity and pinpointing an area where an error or a vulnerability might occur. Since agile development and continuous integration/continuous deployment started to set in, the focus has shifted to iterative development, rapid delivery, and security. In this new context, CC is still useful as a tool to determine code complexity but would have to be taken in addition to other tools and metrics, which would provide a great deal more detail about where possible weaknesses regarding security may occur.

Figure 2: Waterfall SDLC

## Is Cyclomatic Complexity Still Relevant?

A study by Ogunrinde and Akinola (2020) highlighted that CC today is still relevant to contemporary development, especially for code-maintenance purposes. High CC values indicate that a program may be hard to test and render secure because of its complexity. Indeed, reducing complexity helps improve the general quality of the code so that it gets easier to maintain and secure. However, as per the analysis of Stroud, Ertas and Mengel (2019), CC does not directly measure critical security risks in secure software development, such as SQL injection or cross-site scripting. More complicated code means an excellent attack surface where vulnerabilities may remain undiscovered. While CC is helpful in understanding where potential issues may lie, it needs to be used along with security-specific metrics.

## Modern Tools and Techniques

Tools such as SonarQube and Python's Bandit provide real-time information regarding code complexity and security risks (Kuszczyński and Walkowski, 2023). These tools go beyond the reach of CC in detecting vulnerabilities in encryption, injection flaws, or improper input validations. However, even today, most modern tools are built on or keep CC as a primary metric, indicating its continued utility in analysing code structure and complexity.

## The Role of Cyclomatic Complexity in Secure Software Development

CC can point out pieces of the code where complex problems are more likely to introduce vulnerabilities. Hence, simplification can reduce the risk of hard-to-find security-related bugs. As mentioned in the study of Odeh et al. (2024), the complexity does not cause security problems; it makes the detection of vulnerabilities much harder. Minimising the number of control paths can also reduce the attack surface. However, CC in itself is not enough. It should be used with penetration testing, security scanning, and other forms of analysis that provide deeper insight into how vulnerabilities might arise in natural systems.

Figure 3: Role of Cyclomatic Complexity in Secure Software Development

## Conclusion

Conclusion Cyclomatic complexity remains interesting for code quality evaluation, indications of risks, and related factors. Nowadays, security and agility cannot be abandoned in development; CC must be configured with other tools and practices to assure coverage over its deep security analysis. Though CC is not a self-sufficient solution, it is an integral part of developing secure and maintainable software.

# Portfolio 2: Ontology and Faceted Data as Tools for Secure System Design

An ontology provides a structured framework that organises information and clarifies the relationships within a piece of information in a specified domain (Arogundade, Abayomi-Alli and Misra, 2022). Ontology thus becomes an essential phenomenon in secure software development when considering sensitive data management. For example, in a medical record or financial information-handling system, ontology can categorise data as public, confidential, or highly sensitive and describe permissions and security protocols related to each. An ontology would include the following essential components for the system I am designing.

Figure 4: Components of Ontology

This ontology thus provides a proper structure that aligns with data access management and enforces security policies. This structured approach will help to ensure that any unauthorized individual cannot access sensitive information or leak any breach.

## Faceted Data as a Solution for Data Leakage

In the opinion of Schmitz et al. (2016), faceted data is a crucial dynamic method for implementing enforcement of mechanisms in the information flow policies in the software systems. Such faceted values allow one program to present different views of data depending on the privilege level of the data consumer. The end product is a faceted value of a credit card number, which is a full-out number, billed to an authorised user only, but appears to a less privileged user only with the last four forgotten numbers. Thus, it is possible to keep the yet sensitive data safe while granting the respective access levels at the same time.

## Advantages of Faceted Data

According to Abdel-Fatao et al. (2021), this realises one of the greatest advantages of faceted data; it can impose a fine-grained access control model. In a similar manner, by defining several facets in every data chunks, a developer can be certain that the users with varying privileges are only capable of viewing what is allowed. This is easier when working in the system where data sensitivity varies depending on the role of the users or the environment in which the data will be used. In addition, extra benefits can be gained in terms of system efficiency: Running multiple executions for various user roles is highly unnecessary if a single execution can handle various data views dynamically; this reduces the resulting overhead and thus helps to increase performance against other various techniques like secure multi-execution (Liu et al., 2018). Another added advantage is that the faceted data enhances dynamic security. Because the security policies are enforced at runtime, the system can adapt to changing conditions or the discovery of new vulnerabilities, providing a more adaptable way of protecting sensitive information.

Figure 5: Advantages of Faceted Data

## Challenges and Drawbacks

On the other hand, there are also some cons to faceted data. One big con could be the barriers to getting it done. As per Polikarpova et al. (2020), many facets of each piece of data add layers to the system, making the development and debugging quite tricky. Performance may suffer in the case of a system with faceted data, especially when a large number of facets are involved, or complex security policies need to be enforced. Another concern is misconfiguration, for example if facets are unseen and poorly demarcated, the sensitive data remains open to the unauthorised users. This contemplates that strict observance while designing and testing this model must be put into account so that system works accordingly.

## Designing a Python System for Faceted Data Security

The four possible steps of a fundamental system to perform the faceted data security in python are as follows:

The above showed the basis of applying faceted data in this system as a way of protecting sensitive information and at the same time retaining the flexibility and efficiency in the process.

# References

Abdel-Fatao, H., Nofong, V.M., Agbeyeye, L.K., Umaru, Y.M. and Alese, B.K., 2021. A Robust Approach for Data Leakage Assessment. *Ghana Journal of Technology*, *5*(2), pp.48-59.

Arogundade, O.T., Abayomi-Alli, A. and Misra, S., 2020. An ontology-based security risk management model for information systems. *Arabian Journal for Science and Engineering*, *45*(8), pp.6183-6198.

Chandran, K. and Das Aundhe, M., 2022. Agile or waterfall development: The Clementon Company dilemma. *Journal of Information Technology Teaching Cases*, *12*(1), pp.8-15.

Kuszczyński, K. and Walkowski, M., 2023. Comparative Analysis of Open-Source Tools for Conducting Static Code Analysis. *Sensors*, *23*(18), p.7978.

Liu, M., Peng, X., Jiang, Q., Marcus, A., Yang, J. and Zhao, W., 2018, September. Searching stackoverflow questions with multi-faceted categorisation. In *Proceedings of the 10th Asia-Pacific Symposium on Internetware* (pp. 1-10).

McCabe, T.J., 1976. A complexity measure. *IEEE Transactions on Software Engineering*, (4), pp.308-320.

Odeh, A.H., Odeh, M., Odeh, H. and Odeh, N., 2024. Measuring Cyclomatic Complexity of Source Code Using Machine Learning. *Revue d'Intelligence Artificielle*, *38*(1), p.183.

Ogunrinde, M.A. and Akinola, S.O., 2020. Performance Evaluation of a Code Complexity Measurement Tool: An Empirical Approach. *African Journal of Computing & ICT*, *13*(3).

Polikarpova, N., Stefan, D., Yang, J., Itzhaky, S., Hance, T. and Solar-Lezama, A., 2020. Liquid information flow control. *Proceedings of the ACM on Programming Languages*, *4*(ICFP), pp.1-30.

Schmitz, T., Rhodes, D., Austin, T.H., Knowles, K. and Flanagan, C., 2016, April. Faceted dynamic information flow via control and data monads. In *International Conference on Principles of Security and Trust* (pp. 3-23). Berlin, Heidelberg: Springer Berlin Heidelberg.

Stroud, R.O., Ertas, A. and Mengel, S., 2019. Application of cyclomatic complexity in enterprise architecture frameworks. *IEEE Systems Journal*, *13*(3), pp.2166-2176.