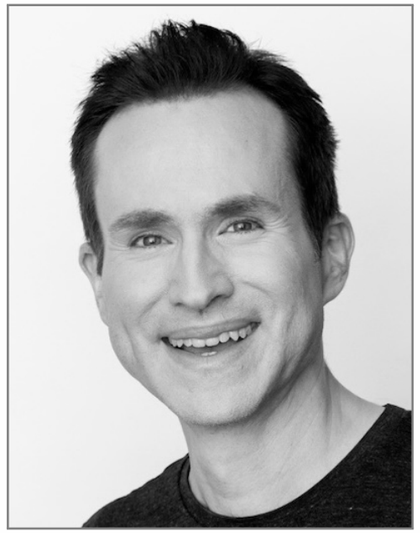
Cyclomatic Complexity

Yegor Bugayenko

Lecture #2 out of 24
80 minutes

The slidedeck was presented by the author in this YouTube Video

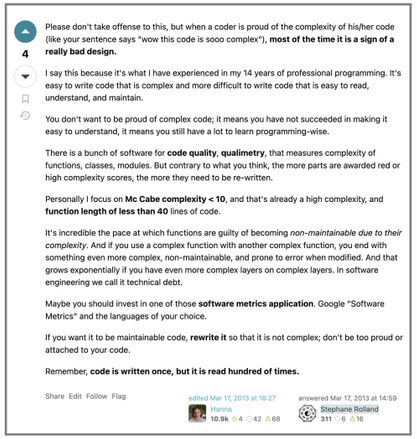


Trevor Foucher

"Instead of minimizing the number of lines, a better metric is to minimize the time needed for someone to understand it."

— Dustin Boswell and Trevor Foucher. The Art of Readable Code, 2011

1. Some programmers mistakenly feel proud of higher complexity of their code [Bugayenko, 2015, 2014].



4

Please don't take offense to this, but when a coder is proud of the complexity of his/her code (like your sentence says "wow this code is sooo complex"), **most of the time it is a sign of a really bad design.**

I say this because it's what I have experienced in my 14 years of professional programming. It's easy to write code that is complex and more difficult to write code that is easy to read, understand, and maintain.

You don't want to be proud of complex code; it means you have not succeeded in making it easy to understand, it means you still have a lot to learn programming-wise.

There is a bunch of software for **code quality**, **qualimetry**, that measures complexity of functions, classes, modules. But contrary to what you think, the more parts are awarded red or high complexity scores, the more they need to be re-written.

Personally I focus on **Mc Cabe complexity < 10**, and that's already a high complexity, and **function length of less than 40** lines of code.

It's incredible the pace at which functions are guilty of becoming *non-maintainable due to their complexity*. And if you use a complex function with another complex function, you end with something even more complex, non-maintainable, and prone to error when modified. And that grows exponentially if you have even more complex layers on complex layers. In software engineering we call it technical debt.

Maybe you should invest in one of those **software metrics application**. Google "Software Metrics" and the languages of your choice.

If you want it to be maintainable code, **rewrite it** so that it is not complex; don't be too proud or attached to your code.

Remember, **code is written once, but it is read hundred of times.**

Share Edit Follow Flag

edited Mar 17, 2013 at 18:27 — Hanna — 10.9k ◇4 ○42 △68

answered Mar 17, 2013 at 14:59 — Stephane Rolland — 311 ○6 △16

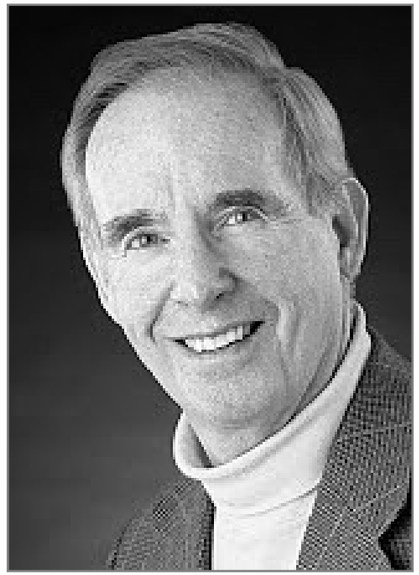
"Please don't take offense to this, but when a coder is proud of the complexity of his/her code (like your sentence says 'wow this code is sooo complex'), most of the time it is a sign of a really bad design."

— Stephane Rolland. How Can I Convey Complexity of Source Code Without Showing Code? https://graphicdesign.stackexchange.com/a/16706/81312, mar 2013. [Online; accessed 22-09-2024]

“The simpler your solution is, the better you are as a software developer. Most software developers can write code that works. Creating code that works and is as simple as possible — that is the true challenge.”

— Vladimir Khorikov. KISS Revisited. https://enterprisecraftsmanship.com/posts/kiss-revisited/, jun 2015. [Online; accessed 22-09-2024]

2. Thomas McCabe suggested to measure Cyclomatic Complexity (CC) of source code.

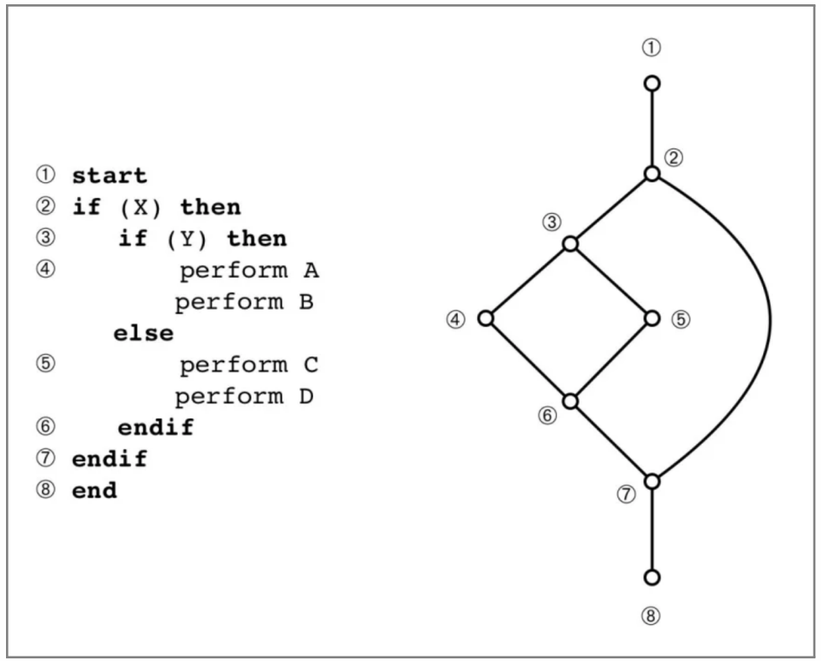


Thomas J. McCabe

"Cyclomatic Complexity (CC) is a count of the number of decisions in the source code. The higher the count, the more complex the code. The formula is simple: $C = E - N + 2$."

— Thomas J. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering*, (4):308–320, 1976. doi:10.1109/tse.1976.233837

What is the complexity of this program?



I found this picture here.

3. CC may correlate with other metrics.

Gregory Seront

"From the results of the experiments we conducted, we observed no significant correlation between the depth of inheritance of a class and its weighted method complexity (WMC)."

— Grégory Seront, Miguel Lopez, Valérie Paulus, and Naji Habra. On the Relationship Between Cyclomatic Complexity and the Degree of Object Orientation. In *Proceedings of the 9th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE)*, pages 109–117, 2005

Meine van der Meulen

> “There is a very strong correlation between Lines of Code and Halstead Volume and there is an even stronger correlation between Lines of Code and McCabe’s Cyclomatic Complexity.”
>
> — Meine J. P. van der Meulen and Miguel A. Revilla. Correlations Between Internal Software Metrics and Software Dependability in a Large Population of Small C/C++ Programs. In *Proceedings of the 18th International Symposium on Software Reliability (ISSRE’07)*, pages 203–208. IEEE, 2007

Yonghee Shin

"The results of our study show weak evidence that software complexity is the enemy of software security for the nine complexity metrics we collected. However, vulnerable code seems to be more complex than faulty code."

— Yonghee Shin and Laurie Williams. Is Complexity Really the Enemy of Software Security? In *Proceedings of the 4th Workshop on Quality of Protection*, pages 47–50, 2008. doi:10.1145/1456362.1456372

Abram Hindle

"Our results strongly suggest that measuring indentation is a cheap and accurate proxy for code complexity of revisions."

— Abram Hindle, Michael W. Godfrey, and Richard C. Holt. Reading Beside the Lines: Indentation as a Proxy for Complexity Metrics. In *Proceedings of the 16th International Conference on Program Comprehension*, pages 133–142. IEEE, 2008. doi:10.1109/ICPC.2008.13

Joanne E. Hale

"The models developed are found to successfully predict roughly 90% of CC's variance by LOC alone. This suggest not only that the linear relationship between LOC and CC is stable, but the aspects of code complexity that CC measures, such as the size of the test case space, grow linearly with source code size across languages and programming paradigms."

— Jay Graylin, Joanne E. Hale, Randy K. Smith, Hale David, Nicholas A. Kraft, and Ward Charles. Cyclomatic Complexity and Lines of Code: Empirical Evidence of a Stable Linear Relationship. *Journal of Software Engineering and Applications*, 2(3):137, 2009. doi:10.4236/jsea.2009.23020

Md Abdullah Al Mamun

"We also found that complexity and documentation domains are more correlated with size domain than themselves."

— Md Abdullah Al Mamun, Christian Berger, and Jörgen Hansson. Correlations of Software Code Metrics: An Empirical Study. In *Proceedings of the 27th International Workshop on Software Measurement and the 12th International Conference on Software Process and Product Measurement*, pages 255–266, 2017. doi:10.1145/3143434.3143445

Adnan Muslija

“There is a low to moderate correlation between the effort needed to test a program and its complexity.”

— Adnan Muslija and Eduard P. Enoiu. On the Correlation Between Testing Effort and Software Complexity Metrics, 2018

Abd Jader

"As the complexity of the software increases, the probability to introduce new errors also increases."

— Abd Jader, Marwa Najm, and Riyadh Zaghlool Mahmood. Calculating McCabe's Cyclomatic Complexity Metric and Its Effecct on the Quality Aspects of Software. *International Journal of Innovative Research and Creative Technology*, (5), 2018

4. In his presentation "*Software Quality Metrics to Identify Risk*", Tom McCabe introduced the following categorisation to interpret cyclomatic complexity: 1–10 little risk, 11–20 moderate risk, 21–50 high risk, 50+ very high risk.

Geoffrey K. Gill

“The complexity density ratio is demonstrated to be a useful predictor of software maintenance productivity on a small pilot sample of actual maintenance project.”

— Geoffrey K. Gill and Chris F. Kemerer. Cyclomatic Complexity Density and Software Maintenance Productivity. *IEEE Transactions on Software Engineering*, 17(12):1284–1288, 1991. doi:10.1109/32.106988

5. Tom McCabe suggested to prohibit functions where CC is larger than ten. Modern static analyzers may help you control this automatically.

CC can be calculated by a few tools:

- Checkstyle for Java
- PMD for Java
- mccabe for Python
- Rubocop for Ruby
- JSHint for JavaScript
- Lizard for C++, and 20 other languages

References

Dustin Boswell and Trevor Foucher. The Art of Readable Code, 2011.

Yegor Bugayenko. Are You a Hacker or a Designer? `https://www.yegor256.com/141026.html`, oct 2014. [Online; accessed 22-09-2024].

Yegor Bugayenko. The Better Architect You Are, the Simpler Your Diagrams. `https://www.yegor256.com/150629.html`, jun 2015. [Online; accessed 22-09-2024].

Geoffrey K. Gill and Chris F. Kemerer. Cyclomatic Complexity Density and Software Maintenance Productivity. *IEEE Transactions on Software Engineering*, 17(12):1284–1288, 1991. doi:10.1109/32.106988.

Jay Graylin, Joanne E. Hale, Randy K. Smith, Hale David, Nicholas A. Kraft, and Ward Charles. Cyclomatic Complexity and Lines of Code: Empirical Evidence of a Stable Linear Relationship. *Journal of Software Engineering and Applications*, 2 (3):137, 2009. doi:10.4236/jsea.2009.23020.

Abram Hindle, Michael W. Godfrey, and Richard C. Holt. Reading Beside the Lines: Indentation as a Proxy for Complexity Metrics. In *Proceedings of the 16th International Conference on Program Comprehension*, pages 133–142. IEEE, 2008. doi:10.1109/ICPC.2008.13.

Abd Jader, Marwa Najm, and Riyadh Zaghlool Mahmood. Calculating McCabe's Cyclomatic Complexity Metric and Its Effecct on the Quality Aspects of Software. *International Journal of Innovative Research and Creative Technology*, (5), 2018.

Vladimir Khorikov. KISS Revisited. `https://enterprisecraftsmanship.com/posts/kiss-revisited/`, jun 2015. [Online; accessed 22-09-2024].

Md Abdullah Al Mamun, Christian Berger, and Jörgen Hansson. Correlations of Software Code Metrics: An Empirical Study. In *Proceedings of the 27th International Workshop on Software Measurement and the 12th International Conference*

on Software Process and Product Measurement, pages 255–266, 2017. doi:10.1145/3143434.3143445.

Thomas J. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering*, (4):308–320, 1976. doi:10.1109/tse.1976.233837.

Adnan Muslija and Eduard P. Enoiu. On the Correlation Between Testing Effort and Software Complexity Metrics, 2018.

Stephane Rolland. How Can I Convey Complexity of Source Code Without Showing Code? `https://graphicdesign.stackexchange.com/a/16706/81312`, mar 2013. [Online; accessed 22-09-2024].

Grégory Seront, Miguel Lopez, Valérie Paulus, and Naji Habra. On the Relationship Between Cyclomatic Complexity and the Degree of Object Orientation. In *Proceedings of the 9th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE)*, pages 109–117, 2005.

Yonghee Shin and Laurie Williams. Is Complexity Really the Enemy of Software Security? In *Proceedings of the 4th Workshop on Quality of Protection*, pages 47–50, 2008. doi:10.1145/1456362.1456372.

Meine J. P. van der Meulen and Miguel A. Revilla. Correlations Between Internal Software Metrics and Software Dependability in a Large Population of Small C/C++ Programs. In *Proceedings of the 18th International Symposium on Software Reliability (ISSRE'07)*, pages 203–208. IEEE, 2007.